Guía del usuario de

# AWS CodePipeline



Versión de API 2015-07-09

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

# AWS CodePipeline: Guía del usuario de

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

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

# Table of Contents

¿Qué es CodePipeline? ..... 1
Integración y entrega continuas ..... 1
¿Qué puedo hacer con CodePipeline? ..... 2
Un vistazo rápido a CodePipeline ..... 3
¿Cómo puedo empezar CodePipeline? ..... 4
Conceptos ..... 4
Canalizaciones ..... 5
Ejecuciones de canalización ..... 7
Operaciones de etapas ..... 8
Ejecuciones de acción ..... 9
Tipos de ejecución ..... 9
Tipos de acción ..... 9
Artefactos ..... 10
Revisiones de origen ..... 10
Desencadenadores ..... 10
Variables ..... 11
Condiciones ..... 11
Reglas ..... 12
DevOps ejemplo de canalización ..... 12
Cómo funcionan las ejecuciones de canalización ..... 14
Cómo se inician las ejecuciones de canalización ..... 15
Cómo se procesan las revisiones de origen en ejecuciones de canalización ..... 15
Cómo funcionan las anulaciones de código fuente con el transformador de entrada EventBridge ..... 16
Cómo se detienen las ejecuciones de canalización ..... 18
Procesamiento de ejecuciones en el modo SUPERSEDED ..... 21
Procesamiento de ejecuciones en modo EN COLA ..... 22
Procesamiento de ejecuciones en modo PARALELO ..... 24
Gestión del flujo de canalización ..... 24
Artefactos de entrada y salida ..... 27
Funcionamiento de las condiciones de las etapas ..... 30
Reglas para condiciones de etapas ..... 33
Tipos de canalización ..... 34
¿Qué tipo de canalización es el adecuado para mí? ..... 35

Introducción ..... 40
Inscríbase en un Cuenta de AWS ..... 40
Aplique una política gestionada para el acceso administrativo a CodePipeline ..... 41
Instala la AWS CLI ..... 42
Abre la consola para CodePipeline ..... 43
Siguientes pasos ..... 43
Integraciones de productos y servicios ..... 44
Integraciones con CodePipeline tipos de acciones ..... 44
Integraciones de acciones de código fuente ..... 44
Integraciones de acciones de compilación ..... 52
Integraciones de acciones de prueba ..... 54
Integraciones de acciones de implementación ..... 56
Integración de la acción de aprobación con Amazon Simple Notification Service ..... 63
Integraciones de acciones de invocación ..... 63
Integraciones generales con CodePipeline ..... 65
Ejemplos de la comunidad ..... 68
Publicaciones de blog ..... 68
Tutoriales ..... 73
Tutorial: Implemente en instancias de Amazon EC2 con CodePipeline ..... 74
Requisitos previos ..... 75
Paso 1: creación de instancias de Amazon EC2 en Linux ..... 75
Paso 2: agregación de los permisos del bucket de artefactos al rol de instancia de EC2 ..... 77
Paso 3: agregación del archivo de script al repositorio ..... 78
Paso 4: creación de una canalización ..... 79
Paso 5: Pon a prueba tu canalización ..... 83
Tutorial: Cree y envíe una imagen de Docker a Amazon ECR con CodePipeline (tipo V2) ..... 84
Requisitos previos ..... 85
Paso 1: agregación de un Dockerfile al repositorio de origen ..... 85
Paso 2: agregación del archivo imagedefinitions.json a su repositorio de origen ..... 87
Paso 3: creación de una canalización ..... 88
Paso 4: Probar la canalización ..... 94
Tutorial: Implemente en Amazon EKS con CodePipeline ..... 95
Requisitos previos ..... 95
Paso 1: creación de un clúster en Amazon EKS (opcional) ..... 96
Paso 2: configuración de un clúster privado en Amazon EKS ..... 98
Paso 3: Actualiza la política de roles de CodePipeline servicio en IAM ..... 99

Paso 4: Cree una entrada de acceso para la función CodePipeline de servicio ..... 102
Paso 5: Crea un repositorio de código fuente y agrega los archivos de configuración del gráfico de `Helm` ..... 103
Paso 6: creación de una canalización ..... 104
Tutorial: creación de una canalización que ejecute comandos mediante computación (tipo V2) ..... 106
Requisitos previos ..... 107
Paso 1: Crea los archivos fuente y envíalos a tu GitHub repositorio ..... 107
Paso 2: Crear la canalización ..... 108
Paso 3: ejecución de una canalización y verificación de los comandos de compilación ..... 112
Tutorial: Usar etiquetas de Git para iniciar la canalización ..... 113
Requisitos previos ..... 114
Paso 1: Abre CloudShell y clona tu repositorio ..... 114
Paso 2: Crear una canalización para activarla en las etiquetas de Git ..... 115
Paso 3: Etiquetar las confirmaciones para publicarlas ..... 119
Paso 4: Publicar los cambios y ver los registros ..... 120
Tutorial: filtrado de solicitudes de extracción por nombres de ramificación para iniciar una canalización (tipo V2) ..... 121
Requisitos previos ..... 121
Paso 1: creación de una canalización que iniciar con una solicitud de extracción de ramificaciones específicas ..... 122
Paso 2: Crea y fusiona una pull request GitHub.com para iniciar la ejecución de tu canalización ..... 124
Tutorial: Uso de variables a nivel de canalización ..... 126
Requisitos previos ..... 126
Paso 1: Crear la canalización y compilar el proyecto ..... 126
Paso 2: Publicar los cambios y ver los registros ..... 130
Tutorial: Crear una canalización simple (bucket de S3) ..... 130
Creación de un bucket de S3 ..... 132
Cree instancias Amazon EC2 de Windows Server e instale el agente CodeDeploy ..... 134
Crea una aplicación en CodeDeploy ..... 136
Crear la primera canalización ..... 138
Agregar otra etapa ..... 141
Activar y desactivar transiciones entre etapas ..... 149
Eliminar recursos ..... 150
Tutorial: Crear una canalización simple (CodeCommit repositorio) ..... 151

Crea un CodeCommit repositorio ..... 152
Descargar, confirmar e insertar el código ..... 153
Cree una instancia Linux de Amazon EC2 e instale el agente CodeDeploy ..... 156
Crea una aplicación en CodeDeploy ..... 158
Crear la primera canalización ..... 159
Actualiza el código de tu CodeCommit repositorio ..... 163
Eliminar recursos ..... 165
Documentación adicional ..... 165
Tutorial: Crear una canalización de cuatro etapas ..... 166
Cómo completar los requisitos previos de ..... 167
Creación de una canalización ..... 172
Agregar más etapas ..... 174
Eliminar recursos ..... 177
Tutorial: Configura una regla de CloudWatch eventos para recibir notificaciones por correo electrónico cuando se produzcan cambios en el estado del oleoducto ..... 178
Configurar una notificación de correo electrónico mediante Amazon SNS ..... 179
Cree una regla de notificación de CloudWatch eventos para CodePipeline ..... 180
Eliminar recursos ..... 182
Tutorial: Crea y prueba una aplicación para Android con AWS Device Farm ..... 182
Configure CodePipeline para usar sus pruebas de Device Farm ..... 183
Tutorial: Prueba una aplicación para iOS con AWS Device Farm ..... 188
Configure CodePipeline para usar sus pruebas de Device Farm (ejemplo de Amazon S3) .. 190
Tutorial: Crear una canalización que se implemente en Service Catalog ..... 195
Opción 1: Realizar la implementación en Service Catalog sin un archivo de configuración ... 196
Opción 2: Realizar la implementación en Service Catalog con un archivo de configuración . 201
Tutorial: Crea una canalización con AWS CloudFormation ..... 206
Ejemplo 1: Crear un AWS CodeCommit canalización con AWS CloudFormation ..... 206
Ejemplo 2: Cree una canalización de Amazon S3 con AWS CloudFormation ..... 209
Tutorial: cree una canalización que utilice variables de las acciones de AWS CloudFormation despliegue ..... 213
Requisitos previos: crear un AWS CloudFormation un rol de servicio y un repositorio CodeCommit ..... 214
Paso 1: Descargue, edite y cargue la muestra AWS CloudFormation plantilla ..... 214
Paso 2: Crear la canalización ..... 215
Paso 3: Añade un CloudFormation acción de implementación para crear el conjunto de cambios ..... 218

Paso 4: Agregar una acción de aprobación manual ..... 219
Paso 5: Agregue una acción de CloudFormation implementación para ejecutar el conjunto de cambios ..... 220
Paso 6: Añade una acción CloudFormation de despliegue para eliminar la pila ..... 221
Tutorial: implementación estándar de Amazon ECS con CodePipeline ..... 221
Requisitos previos ..... 222
Paso 1: Añadir un archivo de especificación de compilación a su repositorio de código fuente ..... 225
Paso 2: Crear la canalización de implementación continua ..... 227
Paso 3: Añada los permisos de Amazon ECR al rol CodeBuild ..... 229
Paso 4: Probar la canalización ..... 230
Tutorial: Cree una canalización con una fuente e implementación de Amazon ECR ECS-to-CodeDeploy ..... 230
Requisitos previos ..... 233
Paso 1: Crear una imagen y enviarla al repositorio de Amazon ECR ..... 233
Paso 2: Cree los archivos AppSpec fuente y de definición de tareas y envíelos a un repositorio CodeCommit ..... 234
Paso 3: Crear el equilibrador de carga de aplicaciones y los grupos de destino ..... 238
Paso 4: Crear un clúster y un servicio de Amazon ECS ..... 241
Paso 5: Cree su grupo de CodeDeploy aplicaciones e implementaciones (plataforma informática ECS) ..... 244
Paso 6: Crear la canalización ..... 245
Paso 7: Realizar un cambio en la canalización y verificar la implementación ..... 250
Tutorial: Crear una canalización que implemente una habilidad de Amazon Alexa ..... 250
Requisitos previos ..... 251
Paso 1: Crear un perfil de seguridad de LWA para los servicios de desarrolladores de Alexa ..... 251
Paso 2: Crea los archivos fuente de habilidades de Alexa y envíalos a tu CodeCommit repositorio ..... 251
Paso 3: Utilizar los comandos de la CLI de ASK para crear un token de actualización ..... 253
Paso 4: Crear la canalización ..... 254
Paso 5: Realizar un cambio en cualquier archivo de origen y verificar la implementación .... 257
Tutorial: Crear una canalización que utilice Amazon S3 como proveedor de implementación ... 257
Opción 1: Implementar los archivos de un sitio web estático en Amazon S3 ..... 258
Opción 2: Implementar archivos compilados archivados en Amazon S3 desde un bucket de origen de S3 ..... 263

Tutorial: Publica aplicaciones en AWS Serverless Application Repository ..... 269
Antes de empezar ..... 270
Paso 1: Crear un archivo buildspec.yml ..... 271
Paso 2: Crear y configurar la canalización ..... 271
Paso 3: Implementar la aplicación de publicación ..... 273
Paso 4: Crear la acción de publicación ..... 274
Tutorial: Implementaciones de funciones Lambda con CodePipeline ..... 275
Requisitos previos ..... 275
Paso 1: creación de la función de Lambda de ejemplo ..... 276
Paso 2: carga del archivo de la función en el repositorio ..... 277
Paso 3: creación de una canalización ..... 277
Paso 4: Probar la canalización ..... 280
Más información ..... 281
Tutorial: Uso de variables con acciones de invocación de Lambda ..... 281
Requisitos previos ..... 282
Paso 1: Crear una función de Lambda ..... 282
Paso 2: Añadir una acción de invocación de Lambda y una acción de aprobación manual a la canalización ..... 285
Tutorial: Usa una acción de AWS Step Functions invocación ..... 287
Requisito previo: cree o elija una canalización sencilla ..... 288
Paso 1: Crear la máquina de estados de ejemplo ..... 288
Paso 2: Agregar una acción de invocación de Step Functions a la canalización ..... 288
Tutorial: Cree una canalización que se utilice AppConfig como proveedor de despliegue ..... 290
Requisitos previos ..... 290
Paso 1: Crea tus recursos AWS AppConfig ..... 291
Paso 2: Cargar los archivos en un bucket de origen de S3 ..... 291
Paso 3: Crear la canalización ..... 292
Paso 4: Realizar un cambio en cualquier archivo de origen y verificar la implementación ..... 294
Tutorial: Utilice un clon completo con una fuente de GitHub canalización ..... 294
Requisitos previos ..... 295
Paso 1: Crear un archivo README ..... 296
Paso 2: Crear la canalización y compilar el proyecto ..... 296
Paso 3: actualice la política CodeBuild de roles de servicio para usar las conexiones ..... 300
Paso 4: Ver comandos del repositorio en el resultado de la compilación ..... 300
Tutorial: Usa un clon completo con una fuente de CodeCommit canalización ..... 301
Requisitos previos ..... 302

Paso 1: Crear un archivo README ..... 302
Paso 2: Crear la canalización y compilar el proyecto ..... 302
Paso 3: Actualiza la política CodeBuild de roles de servicio para clonar el repositorio ..... 305
Paso 4: Ver comandos del repositorio en el resultado de la compilación ..... 306
Tutorial: crear una canalización con acciones AWS CloudFormation StackSets de despliegue ..... 306
Requisitos previos ..... 307
Paso 1: Cargue la muestra AWS CloudFormation archivo de plantilla y parámetros ..... 308
Paso 2: Crear la canalización ..... 310
Paso 3: Ver la implementación inicial ..... 312
Paso 4: Añade una CloudFormationStackInstances acción ..... 313
Paso 5: Ver los recursos del conjunto de pilas para su implementación ..... 314
Paso 6: Actualizar el conjunto de pilas ..... 315
Cómo crear una regla de verificación de variables para una canalización como una condición de entrada ..... 315
Requisitos previos ..... 316
Paso 1: Cree un archivo fuente de muestra y agréguelo a su GitHub repositorio ..... 316
Paso 2: Crear la canalización ..... 317
Paso 2: edición de la etapa de compilación para agregar la condición y la regla ..... 320
Paso 3: ejecución de la canalización y visualización de las variables resueltas ..... 322
Prácticas recomendadas y casos de uso ..... 325
Ejemplos de cómo usar CodePipeline ..... 325
Úselo CodePipeline con Amazon S3, AWS CodeCommit, y AWS CodeDeploy ..... 325
Úselo CodePipeline con proveedores de acciones de terceros (GitHub y Jenkins) ..... 326
Se usa CodePipeline para compilar, compilar y probar código con CodeBuild ..... 327
Úselo CodePipeline con Amazon ECS para la entrega continua de aplicaciones basadas en contenedores a la nube ..... 327
Úselo CodePipeline con Elastic Beanstalk para la entrega continua de aplicaciones web a la nube ..... 327
Úselo con CodePipeline AWS Lambda para la entrega continua Lambda-based de aplicaciones sin servidor ..... 327
Úselo CodePipeline con CloudFormation plantillas para la entrega continua a la nube ..... 328
Uso CodePipeline con Amazon VPC ..... 329
Disponibilidad. ..... 329
Cree un punto final de VPC para CodePipeline ..... 330
Solución de problemas con la configuración de la VPC ..... 331

Defina CI/CD las canalizaciones con etapas y acciones ..... 332
Creación de una canalización, etapas y acciones ..... 333
Creación de una canalización personalizada (consola) ..... 334
Crear una canalización (CLI) ..... 348
Creación de una canalización a partir de plantillas estáticas ..... 354
Editar una canalización ..... 361
Editar una canalización (consola) ..... 362
Editar una canalización (AWS CLI) ..... 366
Ver las canalizaciones y los detalles ..... 370
Ver canalizaciones (consola) ..... 370
Ver los detalles de las acciones en una canalización (consola) ..... 375
Ver el ARN de la canalización y el ARN del rol de servicio (consola) ..... 378
Visualización de los detalles y el historial de la canalización (CLI) ..... 379
Visualización de resultados de reglas para condiciones de etapa en el historial de ejecución ..... 380
Eliminar una canalización ..... 383
Eliminar una canalización (consola) ..... 383
Eliminar una canalización (CLI) ..... 384
Crear una canalización que use recursos de otra cuenta ..... 385
Requisito previo: crear un AWS KMS clave de cifrado ..... 387
Paso 1: Configurar roles y políticas de cuenta ..... 388
Paso 2: Editar la canalización ..... 395
Migrar los canales de sondeo para utilizar la detección de cambios basada en eventos ..... 398
Cómo migrar canalizaciones de sondeo ..... 399
Cómo ver las canalizaciones de sondeo de su cuenta ..... 400
Migre las canalizaciones de sondeo con una fuente CodeCommit ..... 405
Migración de canalizaciones de sondeo con un origen de S3 habilitado para los eventos ..... 428
Migre las canalizaciones de sondeo con una fuente y CloudTrail un registro de S3 ..... 448
Migre las canalizaciones de sondeo de una GitHub acción fuente (mediante una aplicación OAuth) a las conexiones ..... 484
Migra los canales de sondeo de una acción fuente GitHub (a través de la aplicación OAuth) a webhooks ..... 488
Cree el rol CodePipeline de servicio ..... 505
Crea el rol CodePipeline de servicio (consola) ..... 506
Cree el rol CodePipeline de servicio (CLI) ..... 507
Etiquetado de recursos ..... 509

Etiquetado de una canalización ..... 510
Etiquetado de canalizaciones (consola) ..... 510
Etiquetado de canalizaciones (CLI) ..... 512
Creación de una regla de notificación ..... 514
..... 519
Conexión a proveedores de origen propios mediante acciones de origen ..... 522
Acciones de origen de Amazon ECR y EventBridge ..... 522
Cree una EventBridge regla para una fuente de Amazon ECR (consola) ..... 523
Cree una EventBridge regla para una fuente (CLI) de Amazon ECR ..... 526
Cree una EventBridge regla para una fuente de Amazon ECR (CloudFormation plantilla) .... 529
Conectarse a las acciones de origen de Amazon S3 que utilizan EventBridge eventos ..... 534
Creación de canalizaciones con un origen de S3 habilitado para los eventos (CLI) ..... 535
Crea canalizaciones con una fuente de S3 habilitada para eventos (CloudFormation plantilla) ..... 539
Conectarse a las acciones de origen de Amazon S3 que utilizan EventBridge y AWS CloudTrail ..... 554
Cree una EventBridge regla para una fuente de Amazon S3 (consola) ..... 555
Cree una EventBridge regla para una fuente de Amazon S3 (CLI) ..... 559
Cree una EventBridge regla para una fuente de Amazon S3 (CloudFormation plantilla) ..... 565
CodeCommit acciones de origen y EventBridge ..... 578
Cree una EventBridge regla para una CodeCommit fuente (consola) ..... 579
Crea una EventBridge regla para una CodeCommit fuente (CLI) ..... 582
Crea una EventBridge regla para una CodeCommit fuente (CloudFormation plantilla) ..... 587
Agrega proveedores de fuentes de terceros a las canalizaciones mediante AWS CodeConnections ..... 595
DevOps Conexiones de Azure ..... 595
Cree una conexión a Azure DevOps (consola) ..... 597
Crear una conexión a Azure DevOps (CLI) ..... 601
Conexiones de Bitbucket Cloud ..... 603
Creación de una conexión a Bitbucket Cloud (consola) ..... 605
Creación de una conexión a Bitbucket Cloud (CLI) ..... 608
GitHub conexiones ..... 610
Crea una conexión a GitHub (consola) ..... 612
Crear una conexión a GitHub (CLI) ..... 615
GitHub Conexiones de Enterprise Server ..... 617
Cree una conexión a GitHub Enterprise Server (consola) ..... 619

Crear un host y una conexión a GitHub Enterprise Server (CLI) ..... 622
GitLab.com conexiones ..... 625
Cree una conexión a GitLab.com (consola) ..... 628
Crear una conexión a GitLab.com (CLI) ..... 632
Conexiones para autogestión GitLab ..... 634
Cree una conexión GitLab autogestionada (consola) ..... 636
Cree un host y una conexión a la red GitLab autogestionada (CLI) ..... 639
Uso de una conexión compartida con otra cuenta ..... 642
Automatización del inicio de las canalizaciones mediante desencadenadores y filtrado ..... 643
Consideraciones sobre los filtros de desencadenadores ..... 645
Eventos de solicitud de extracción para desencadenadores por proveedor ..... 646
Ejemplos de filtros de desencadenador ..... 648
Agregación de un desencadenador con la inserción de código sin filtro ..... 657
Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código ..... 657
Agregación de filtros para los tipos de eventos de solicitud de inserción y extracción (consola) ..... 659
Agregación de filtros para los tipos de eventos de solicitud de inserción y extracción (CLI) .. 662
Añade filtros para los tipos de eventos de solicitud push y pull (CloudFormation plantillas) .. 664
Agregación de un desencadenador para desactivar la detección de cambios ..... 666
Inicio y detención manuales de canalizaciones ..... 667
Inicie una canalización en CodePipeline ..... 667
Iniciar la canalización manualmente ..... 669
Inicio de una canalización según una programación ..... 670
Iniciar una canalización con una anulación de revisión de código fuente ..... 674
Detener la ejecución de una canalización ..... 677
Detener la ejecución de una canalización (consola) ..... 678
Detener una ejecución entrante (consola) ..... 681
Detener la ejecución de una canalización (CLI) ..... 682
Detener una ejecución entrante (CLI) ..... 683
Consulta del historial y configuración del modo de ejecución de las canalizaciones ..... 685
Ver ejecuciones ..... 685
Consulta del historial de ejecución de una canalización (consola) ..... 685
Consulta del estado de ejecución (consola) ..... 687
Ver una ejecución entrante (consola) ..... 689

Consulta de las revisiones de código fuente de las ejecuciones de una canalización (consola) ... 690
Consulta de las ejecuciones de una acción (consola) ... 692
Consulta de artefactos de acción y de información del almacén de artefactos (consola) ... 693
Visualización de los detalles y el historial de la canalización (CLI) ... 693
Configuración o cambio del modo de ejecución de una canalización ... 706
Consideraciones para la visualización de los modos de ejecución ... 707
Consideraciones para cambiar de un modo de ejecución a otro ... 709
Configuración o cambio del modo de ejecución de una canalización (consola) ... 710
Configuración del modo de ejecución de una canalización (CLI) ... 711
Reversión o reintento de etapas ... 715
Configuración del reintento de una etapa fallida o de acciones fallidas ... 715
Consideraciones sobre el reintento de etapas ... 716
Reintento de una etapa fallida manualmente ... 716
Configuración de una etapa para el reintento automático en caso de fallo ... 721
Configuración de la reversión de etapas ... 726
Consideraciones sobre las reversiones ... 727
Reversión manual de una etapa ... 727
Configuración de una etapa para la reversión automática ... 733
Visualización del estado de reversión en la lista de ejecuciones ... 737
Visualización de los detalles del estado de reversión ... 740
Configuración de las condiciones de una etapa ... 745
Casos de uso de condiciones de etapa ... 746
Consideraciones sobre la configuración de los resultados para las condiciones de etapa ... 746
Consideraciones sobre las reglas configuradas para las condiciones de etapa ... 747
Creación de condiciones de entrada ... 748
Creación de condiciones de entrada: ejemplo de CloudWatchAlarm regla (consola) ... 748
Creación de condiciones de entrada con el resultado Omitir y la regla `VariableCheck` (consola) ... 749
Creación de condiciones de entrada (CLI) ... 751
Creación de condiciones de entrada (CFN) ... 753
Creación de condiciones de fallo ... 754
Creación de condiciones de fallo (consola) ... 754
Creación de condiciones onFailure con ejemplo de resultado de reintento (consola) ... 756
Creación de condiciones de fallo (CLI) ... 757
Creación de condiciones de fallo (CFN) ... 759

Creación de condiciones de éxito ... 760
Creación de condiciones de éxito (consola) ... 760
Creación de condiciones de éxito (CLI) ... 762
Creación de una condición de éxito (CFN) ... 764
Eliminación de condiciones de etapa ... 765
Anulación de condiciones de etapa ... 766
Uso de tipos de acciones, acciones personalizadas y acciones de aprobación ... 768
Trabajar con tipos de acciones ... 768
Solicitar un tipo de acción ... 770
Añadir un tipo de acción disponible a una canalización (consola) ... 776
Ver un tipo de acción ... 778
Actualizar un tipo de acción ... 780
Crear una acción personalizada para una canalización ... 781
Crear una acción personalizada ... 784
Crear un proceso de trabajo para la acción personalizada ... 788
Agregar una acción personalizada a una canalización ... 797
Etiqueta una acción personalizada en CodePipeline ... 800
Incorporación de etiquetas en una acción personalizada ... 800
Visualización de las etiquetas de una acción personalizada ... 801
Edición de las etiquetas de una acción personalizada ... 801
Eliminación de etiquetas de una acción personalizada ... 802
Invocar una función de Lambda en una canalización ... 802
Paso 1: Crear una canalización ... 805
Paso 2: Crear la función de Lambda ... 806
Paso 3: Agregue la función Lambda a una canalización en la consola CodePipeline ... 810
Paso 4: Probar la canalización con la función de Lambda ... 811
Paso 5: Siguientes pasos ... 812
Ejemplo de evento JSON ... 813
Más funciones de ejemplo ... 815
Incorporación de una acción de aprobación manual a una etapa ... 828
Opciones de configuración de las acciones de aprobación manual ... 829
Información general sobre la configuración y el flujo de trabajo de las acciones de aprobación ... 830
Conceda permisos de aprobación a un usuario de IAM en CodePipeline ... 831
Concesión de permisos de Amazon SNS a un rol de servicio ... 834
Incorporación de una acción de aprobación manual ... 836

Aprobación o rechazo de una acción de aprobación ... 840
Formato de datos JSON de las notificaciones de aprobación manual ... 844
Agrega una acción interregional a una canalización ... 845
Gestiona las acciones interregionales en una canalización (consola) ... 847
Agregar una acción interregional a una canalización (CLI) ... 850
Añade una acción interregional a una canalización (CloudFormation) ... 855
Trabajar con variables ... 858
Configuración de acciones para variables ... 859
Ver variables de salida ... 863
Ejemplo: Usar variables en aprobaciones manuales ... 866
Ejemplo: usa una BranchName variable con variables de CodeBuild entorno ... 867
Trabajar con transiciones de etapa ... 869
Activar o desactivar transiciones (consola) ... 869
Activar o desactivar transiciones (CLI) ... 871
Monitorear canalizaciones ... 873
Monitorización de CodePipeline eventos ... 874
Tipos de detalles ... 876
Pipeline-level eventos ... 878
Stage-level eventos ... 887
Action-level eventos ... 891
Creación de una regla que envíe una notificación sobre un evento de canalización ... 899
Referencia del bucket de marcadores de posición de eventos ... 903
Nombres de buckets de marcadores de posición de eventos por región ... 904
Registro de llamadas a la API de AWS CloudTrail con ... 908
CodePipeline información en CloudTrail ... 908
Descripción de las entradas del archivo de CodePipeline registro ... 909
CodePipeline CloudWatch métricas ... 911
PipelineDuration ... 912
FailedPipelineExecutions ... 912
Resolución de problemas ... 913
Error de canalización: una canalización configurada con AWS Elastic Beanstalk devuelve un mensaje de error: «Falló la implementación. El rol proporcionado no tiene permisos suficientes: Service:AmazonElasticLoadBalancing» ... 914
Error de implementación: una canalización configurada con un AWS Elastic Beanstalk la acción de despliegue se bloquea en lugar de fallar si falta el permiso DescribeEvents «» ... 915

Error de canalización: una acción de origen devuelve el mensaje de permisos insuficientes: «No se ha podido acceder al nombre del CodeCommit `repositorio`». Asegúrese de que rol de IAM de la canalización tenga permisos suficientes para obtener acceso al repositorio". ..... 916
Error de canalización: una acción de compilación o prueba de Jenkins se ejecuta de manera prolongada y da un error debido a la falta de credenciales o permisos ..... 916
Error de canalización: se creó una canalización en una AWS Región que utiliza un depósito creado en otro AWS La región devuelve un "InternalError" con el código "JobFailed» ..... 917
Error de despliegue: un archivo ZIP que contiene un archivo WAR se ha desplegado correctamente en AWS Elastic Beanstalk, pero la URL de la aplicación indica un error 404 no encontrado ..... 915
Los nombres de la carpeta de artefactos de la canalización parecen estar truncados ..... 918
Añade CodeBuild GitClone permisos para las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com ..... 918
Agregue CodeBuild GitClone permisos para las acciones CodeCommit de origen ..... 920
Error de proceso: una implementación con la CodeDeployToECS acción devuelve un mensaje de error: «Se produjo una excepción al intentar leer el archivo de artefactos de definición de tareas desde: nombre del artefacto de < origen» > ..... 921
GitHub (a través de la aplicación OAuth) acción de origen: la lista de repositorios muestra diferentes repositorios ..... 922
GitHub (mediante GitHub la aplicación) acción de origen: no se ha podido completar la conexión a un repositorio ..... 922
Error de Amazon S3: al CodePipeline ARN del rol < de servicio > se le deniega el acceso a S3 para el bucket de S3 < BucketName > ..... 922
Las canalizaciones con Amazon S3, Amazon ECR o CodeCommit fuente ya no se inician automáticamente ..... 925
Error de conexión al conectarse a GitHub: «Se ha producido un problema, asegúrate de que las cookies estén habilitadas en tu navegador» o «El propietario de una organización debe instalar la GitHub aplicación» ..... 927
Las canalizaciones con el modo de ejecución cambiado a MODO EN COLA o PARALELO fallan cuando se alcanza el límite de ejecución ..... 927
Las canalizaciones en modo PARALELO tienen una definición de canalización desactualizada si se editan al cambiar al modo EN COLA o SUPERSEDED ..... 928
Las canalizaciones que cambien del modo PARALELO a otro mostrarán un modo de ejecución anterior ..... 928
Es posible que las canalizaciones con conexiones que utilicen el filtrado de desencadenadores por rutas de archivo no se inicien al crear la ramificación ..... 929

Es posible que las canalizaciones con conexiones que utilicen el filtrado de desencadenadores por rutas de archivo no se inicien cuando se alcance el límite de archivos ..... 929
CodeCommit o las revisiones del código fuente de S3 en modo paralelo podrían no coincidir con el EventBridge evento ..... 930
La acción de despliegue de EC2 falla y muestra el mensaje de error `No existe tal archivo` ..... 930
La acción EKS Deploy falla y aparece un mensaje de error `inalcanzable para el clúster` ..... 931
La canalización no se activa en todas las sucursales cuando varias acciones de origen hacen referencia al mismo repositorio ..... 932
¿Necesita ayuda con otro problema? ..... 932
Seguridad ..... 934
Protección de datos ..... 935
Privacidad del tráfico entre redes ..... 936
Cifrado en reposo ..... 937
Cifrado en tránsito ..... 937
Administración de claves de cifrado ..... 937
Configurar el cifrado del lado del servidor para los artefactos almacenados en Amazon S3 para CodePipeline ..... 937
Uso AWS Secrets Manager para rastrear las contraseñas de las bases de datos o las claves de API de terceros ..... 941
Identity and Access Management ..... 941
Público ..... 942
Autenticación con identidades ..... 942
Administración del acceso con políticas ..... 943
Cómo AWS CodePipeline funciona con IAM ..... 945
Identity-based ejemplos de políticas ..... 951
Resource-based ejemplos de políticas ..... 980
Resolución de problemas ..... 981
referencia de permisos ..... 983
Administre la función CodePipeline de servicio ..... 994
Respuesta a incidentes ..... 1001
Validación de conformidad ..... 1001
Resiliencia ..... 1002
Seguridad de la infraestructura ..... 1002
Prácticas recomendadas de seguridad ..... 1003

Referencia de estructura de la canalización ..... 1004
Declaración de canalización ..... 1008
name ..... 1010
roleArn ..... 1011
ArtifactStore O ArtifactStores ..... 1011
etapas ..... 1012
versión ..... 1013
executionMode ..... 1013
Tipo de tubería ..... 1014
variables ..... 1014
desencadenadores ..... 1014
metadatos ..... 1019
Declaración de etapas ..... 1020
name ..... 1022
actions ..... 1023
condiciones ..... 1023
reglas ..... 1023
Declaración de acciones ..... 1023
name ..... 1028
region ..... 1028
roleArn ..... 1028
namespace ..... 1028
acción TypeId ..... 1029
InputArtifacts ..... 1030
Artefacto de salida ..... 1031
configuración (por proveedor de acciones) ..... 1032
RunOrder ..... 1034
Proveedores de acciones válidos en CodePipeline ..... 1035
Configuración válida para el parámetro PollForSourceChanges ..... 1040
Artefactos de entrada y salida para cada tipo de acción ..... 1042
Parámetros de configuración válidos para cada tipo de proveedor ..... 1044
Referencia de la estructura de acciones ..... 1049
Referencia de la acción de Amazon EC2 ..... 1050
Tipo de acción ..... 1051
Parámetros de configuración ..... 1051
Artefactos de entrada ..... 1055

Artefactos de salida ..... 1056
Permisos de la política de rol de servicio para la acción de implementación de EC2 ..... 1056
Referencia del archivo de especificaciones de implementación ..... 1059
Declaración de acciones ..... 1060
Ejemplo de declaración de acción con especificaciones de implementación ..... 1062
Véase también ..... 1063
Referencia de acciones de origen de Amazon ECR ..... 1063
Tipo de acción ..... 1064
Parámetros de configuración ..... 1064
Artefactos de entrada ..... 1065
Artefactos de salida ..... 1065
Variables de salida ..... 1065
Permisos del rol de servicio: acción de Amazon ECR ..... 1066
Declaración de acción (ejemplo de Amazon ECR) ..... 1066
Véase también ..... 1068
Referencia de la acción de compilación ECRBuildAndPublish ..... 1068
Tipo de acción ..... 1069
Parámetros de configuración ..... 1069
Artefactos de entrada ..... 1070
Artefactos de salida ..... 1070
Variables de salida ..... 1070
Permisos del rol de servicio: acción ECRBuildAndPublish ..... 1071
Declaración de acciones ..... 1073
Véase también ..... 1074
Referencia sobre las acciones de implementación de Amazon ECS y CodeDeploy azul-verde ..... 1074
Tipo de acción ..... 1075
Parámetros de configuración ..... 1075
Artefactos de entrada ..... 1077
Artefactos de salida ..... 1078
Permisos del rol de servicio: acción de CodeDeployToECS ..... 1078
Declaración de acciones ..... 1080
Véase también ..... 1082
Referencia de acción de implementación de Amazon Elastic Container Service ..... 1083
Tipo de acción ..... 1084
Parámetros de configuración ..... 1084

Artefactos de entrada ..... 1085
Artefactos de salida ..... 1085
Permisos del rol de servicio: acción estándar de Amazon ECS ..... 1086
Declaración de acciones ..... 1088
Véase también ..... 1089
Referencia de la acción de implementación de EKS de Amazon Elastic Kubernetes Service .. 1089
Tipo de acción ..... 1091
Parámetros de configuración ..... 1091
Artefactos de entrada ..... 1092
Artefactos de salida ..... 1093
Variables de entorno ..... 1093
Variables de salida ..... 1093
Permisos para las políticas de roles de servicio ..... 1093
Declaración de acciones ..... 1096
Véase también ..... 1097
Referencia de la acción de implementación de AWS Lambda ..... 1097
Tipo de acción ..... 1098
Parámetros de configuración ..... 1098
Artefactos de entrada ..... 1102
Artefactos de salida ..... 1102
Variables de salida ..... 1102
Permisos de la política de rol de servicio para la acción de implementación de Lambda .... 1102
Declaración de acciones ..... 1104
Véase también ..... 1105
Referencia de acción de implementación de Amazon S3 ..... 1105
Tipo de acción ..... 1106
Parámetros de configuración ..... 1106
Artefactos de entrada ..... 1108
Artefactos de salida ..... 1108
Permisos del rol de servicio: acción de implementación de S3 ..... 1108
Ejemplo de configuración de una acción ..... 1109
Véase también ..... 1112
Referencia sobre la acción de origen de Amazon S3 ..... 1112
Tipo de acción ..... 1113
Parámetros de configuración ..... 1113
Artefactos de entrada ..... 1115

Artefactos de salida ..... 1115
Variables de salida ..... 1116
Permisos del rol de servicio: acción de origen de S3 ..... 1116
Declaración de acciones ..... 1117
Véase también ..... 1119
Referencia de la acción de implementación de AWS AppConfig ..... 1119
Tipo de acción ..... 1119
Parámetros de configuración ..... 1119
Artefactos de entrada ..... 1120
Artefactos de salida ..... 1120
Permisos del rol de servicio: acción de AppConfig ..... 1120
Ejemplo de configuración de una acción ..... 1121
Véase también ..... 1122
Referencia de la acción de implementación de CloudFormation ..... 1123
Tipo de acción ..... 1124
Parámetros de configuración ..... 1124
Artefactos de entrada ..... 1129
Artefactos de salida ..... 1129
Variables de salida ..... 1129
Permisos de rol de servicio: CloudFormation acción ..... 1130
Declaración de acciones ..... 1131
Véase también ..... 1133
CloudFormation StackSets ..... 1133
Cómo CloudFormation StackSets las acciones funcionan ..... 1134
Cómo estructurar StackSets las acciones en una canalización ..... 1136
Acción CloudFormationStackSet ..... 1138
La acción CloudFormationStackInstances ..... 1152
Permisos del rol de servicio: acción de CloudFormationStackSet ..... 1163
Permisos del rol de servicio: acción CloudFormationStackInstances ..... 1163
Modelos de permisos para operaciones de conjunto de pilas ..... 1163
Tipos de datos de parámetros de plantilla ..... 1164
Véase también ..... 1133
AWS CodeBuild referencia de acciones de creación y prueba ..... 1166
Tipo de acción ..... 1167
Parámetros de configuración ..... 1167
Artefactos de entrada ..... 1170

Artefactos de salida ..... 1171
Variables de salida ..... 1172
Permisos de rol de servicio: CodeBuild acción ..... 1172
Declaración de acción (CodeBuild ejemplo) ..... 1173
Véase también ..... 1174
AWS CodePipeline invocar la referencia de la acción ..... 1175
Tipo de acción ..... 1176
Parámetros de configuración ..... 1176
Artefactos de entrada ..... 1178
Artefactos de salida ..... 1179
Permisos de política de roles de servicio para la acción de CodePipeline invocación ..... 1179
Declaración de acciones ..... 1179
Véase también ..... 1180
AWS CodeCommit referencia de acción de origen ..... 1180
Tipo de acción ..... 1181
Parámetros de configuración ..... 1182
Artefactos de entrada ..... 1183
Artefactos de salida ..... 1183
Variables de salida ..... 1184
Permisos de rol de servicio: CodeCommit acción ..... 1185
Ejemplo de configuración de una acción ..... 1185
Véase también ..... 1188
Referencia de la acción de implementación de AWS CodeDeploy ..... 1188
Tipo de acción ..... 1189
Parámetros de configuración ..... 1189
Artefactos de entrada ..... 1189
Artefactos de salida ..... 1190
Permisos de rol de servicio: AWS CodeDeploy acción ..... 1190
Declaración de acciones ..... 1191
Véase también ..... 1192
CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com autogestionadas GitLab ..... 1193
Tipo de acción ..... 1196
Parámetros de configuración ..... 1197
Artefactos de entrada ..... 1198
Artefactos de salida ..... 1198

Variables de salida ..... 1199
Permisos del rol de servicio: AWS CodeConnections acción ..... 1200
Declaración de acciones ..... 1200
Instalación de la aplicación de instalación y creación de una conexión ..... 1202
Véase también ..... 1202
Referencia de la acción de Comandos ..... 1203
Consideraciones sobre la acción de Comandos ..... 1204
Permisos para las políticas de roles de servicio ..... 1205
Tipo de acción ..... 1206
Parámetros de configuración ..... 1206
Artefactos de entrada ..... 1211
Artefactos de salida ..... 1211
Variables de entorno ..... 1211
Permisos del rol de servicio: acción de Comandos ..... 1213
Declaración de acciones (ejemplo) ..... 1213
Véase también ..... 1215
AWS Device Farm referencia de acción de prueba ..... 1215
Tipo de acción ..... 1215
Parámetros de configuración ..... 1216
Artefactos de entrada ..... 1220
Artefactos de salida ..... 1220
Permisos de rol de servicio: AWS Device Farm acción ..... 1220
Declaración de acciones ..... 1221
Véase también ..... 1222
Referencia de la acción de implementación de Elastic Beanstalk ..... 1223
Tipo de acción ..... 1223
Parámetros de configuración ..... 1223
Artefactos de entrada ..... 1224
Artefactos de salida ..... 1224
Permisos del rol de servicio: acción de implementación `ElasticBeanstalk` ..... 1224
Declaración de acciones ..... 1225
Véase también ..... 1226
Referencia de la acción de invocación de `InspectorScan` de Amazon Inspector ..... 1227
ID de tipo de acción ..... 1227
Parámetros de configuración ..... 1228
Artefactos de entrada ..... 1231

Artefactos de salida ..... 1231
Variables de salida ..... 1231
Permisos del rol de servicio: acción `InspectorScan` ..... 1231
Declaración de acciones ..... 1232
Véase también ..... 1233
AWS Lambda invocar la referencia de la acción ..... 1234
Tipo de acción ..... 1234
Parámetros de configuración ..... 1234
Artefactos de entrada ..... 1235
Artefactos de salida ..... 1235
Variables de salida ..... 1235
Ejemplo de configuración de una acción ..... 1235
Ejemplo de evento JSON ..... 1236
Véase también ..... 1239
Referencia de la acción de implementación de AWS OpsWorks ..... 1239
Tipo de acción ..... 1239
Parámetros de configuración ..... 1240
Artefactos de entrada ..... 1240
Artefactos de salida ..... 1240
Permisos de rol de servicio: AWS OpsWorks acción ..... 1240
Ejemplo de configuración de una acción ..... 1241
Véase también ..... 1242
Referencia de la acción de implementación de AWS Service Catalog ..... 1242
Tipo de acción ..... 1242
Parámetros de configuración ..... 1242
Artefactos de entrada ..... 1243
Artefactos de salida ..... 1243
Permisos del rol de servicio: acción de Service Catalog ..... 1243
Ejemplos de configuraciones de la acción por tipo de archivo de configuración ..... 1244
Véase también ..... 1245
AWS Step Functions ..... 1246
Tipo de acción ..... 1239
Parámetros de configuración ..... 1247
Artefactos de entrada ..... 1248
Artefactos de salida ..... 1248
Variables de salida ..... 1248

Permisos del rol de servicio: acción `StepFunctions` ..... 1249
Ejemplo de configuración de una acción ..... 1249
Comportamiento ..... 1252
Véase también ..... 1122
Referencia de estructura de las reglas ..... 1255
CloudWatchAlarm ..... 1255
Tipo de regla ..... 1256
Parámetros de configuración ..... 1256
Ejemplo de configuraciones de regla ..... 1257
Véase también ..... 1258
Regla de CodeBuild ..... 1258
Permisos para las políticas de roles de servicio ..... 1258
Tipo de regla ..... 1259
Parámetros de configuración ..... 1259
Ejemplo de configuraciones de regla ..... 1261
Véase también ..... 1262
Comandos ..... 1262
Consideraciones sobre la regla de comandos ..... 1263
Permisos para las políticas de roles de servicio ..... 1258
Tipo de regla ..... 1259
Parámetros de configuración ..... 1259
Ejemplo de configuraciones de regla ..... 1261
Véase también ..... 1262
DeploymentWindow ..... 1268
Tipo de regla ..... 1268
Parámetros de configuración ..... 1268
Ejemplo de configuraciones de regla ..... 1271
Véase también ..... 1272
LambdaInvoke ..... 1272
Tipo de regla ..... 1256
Parámetros de configuración ..... 1272
Ejemplo de configuraciones de regla ..... 1273
Véase también ..... 1274
VariableCheck ..... 1274
Tipo de regla ..... 1274
Parámetros de configuración ..... 1275

Ejemplo de configuraciones de regla ... 1277
Véase también ... 1278
Referencia del modelo de integración ... 1279
Cómo funcionan los tipos de acciones de terceros con el integrador ... 1279
Conceptos ... 1280
Modelos de integración compatibles ... 1282
Modelo de integración de Lambda ... 1284
Actualice su función Lambda para gestionar la entrada desde CodePipeline ... 1284
Devuelve los resultados de tu función de Lambda a CodePipeline ... 1288
Utilice los tokens de continuación para esperar los resultados de un proceso asíncrono ... 1290
Proporcione CodePipeline los permisos para invocar la función Lambda del integrador en tiempo de ejecución ... 1291
Modelo de integración de procesos de trabajo ... 1291
Elegir y configurar una estrategia de administración de permisos para el proceso de trabajo ... 1292
Referencia del archivo de definiciones de imágenes ... 1294
Archivo imagedefinitions.json para las acciones de implementación estándar de ... 1294
Detail.json archivo de imagen para las acciones de blue/green implementación de Amazon ECS ... 1297
Referencia de variables ... 1302
Conceptos ... 1303
Variables ... 1303
Espacios de nombres ... 1304
Casos de uso de variables ... 1305
Configuración de variables ... 1306
Configuración de variables a nivel de canalización ... 1306
Configuración de variables a nivel de acción ... 1307
Resolución de variables ... 1309
Reglas para variables ... 1310
Variables disponibles para acciones de canalización ... 1311
Acciones con claves de variables definidas ... 1311
Acciones con claves de variables configuradas por el usuario ... 1317
Trabajar con patrones glob en la sintaxis ... 1320
Actualizar canalizaciones de sondeo para utilizar el método de detección de cambios recomendado ... 1322

Actualiza una acción fuente GitHub (a través de una aplicación OAuth) a una acción fuente GitHub (a través GitHub de una aplicación) ..... 1323
Paso 1: Sustituye tu acción (a través de la aplicación OAuth) GitHub ..... 1325
Paso 2: Crear una conexión a GitHub ..... 1325
Paso 3: Guarde la acción GitHub de origen ..... 1326
Cuotas ..... 1328
Apéndice A: acciones de origen GitHub (a través de la aplicación OAuth) ..... 1346
Añadir una acción fuente GitHub (mediante la aplicación OAuth) ..... 1347
GitHub (a través de la aplicación OAuth) referencia a la acción fuente ..... 1348
Tipo de acción ..... 1349
Parámetros de configuración ..... 1350
Artefactos de entrada ..... 1351
Artefactos de salida ..... 1351
Variables de salida ..... 1352
Declaración de acción (GitHub ejemplo) ..... 1353
Conectarse a GitHub (OAuth) ..... 1354
Véase también ..... 1354
Historial de revisión ..... 1356
Actualizaciones anteriores ..... 1394
Referencia de características de CodePipeline ..... 1405
..... mcdix

# ¿Qué es? AWS CodePipeline?

AWS CodePipeline es un servicio de entrega continua que puede usar para modelar, visualizar y automatizar los pasos necesarios para lanzar su software. Puede modelar y configurar rápidamente las diferentes etapas de un proceso de lanzamiento de software. CodePipeline automatiza los pasos necesarios para lanzar los cambios de software de forma continua. Para obtener información sobre los precios de CodePipeline, consulte Precios.

Temas


- Integración y entrega continuas
- ¿Qué puedo hacer con? CodePipeline
- Un vistazo rápido a CodePipeline
- ¿Cómo puedo empezar CodePipeline?
- CodePipeline conceptos
- DevOps ejemplo de canalización
- Cómo funcionan las ejecuciones de canalización
- Artefactos de entrada y salida
- Funcionamiento de las condiciones de las etapas
- Tipos de canalización
- ¿Qué tipo de canalización es el adecuado para mí?


## Integración y entrega continuas

CodePipeline es un servicio de entrega continua que automatiza la creación, las pruebas y la implementación del software en producción.

La entrega continua es una metodología de desarrollo de software en la que el proceso de lanzamiento está automatizado. Cualquier cambio en el software se compila, prueba e implementa en producción de forma automática. Antes del envío final a producción, una persona, una prueba automatizada o una regla de negocio deciden cuándo debe realizarse el envío final. Aunque todos los cambios realizados correctamente en el software pueden enviarse inmediatamente a producción con la entrega continua, no todos los cambios deben publicarse inmediatamente.

La integración continua es una práctica de desarrollo de software en la que los miembros de un equipo usan un sistema de control de versiones e integran frecuentemente su trabajo en la

misma ubicación, como una ramificación principal. Cada cambio se compila y verifica para detectar los errores de integración lo antes posible. La integración continua consiste en la compilación y comprobación automáticas del código, mientras que la entrega continua consiste en automatizar todo el proceso de lanzamiento de software hasta producción.

Para obtener más información, consulte Practicar la integración y la entrega continuas sobre AWS: Acelerar la entrega de software con DevOps.

Puedes usar la CodePipeline consola, los AWS Command Line Interface (AWS CLI), los AWS SDK o cualquier combinación de estos para crear y administrar tus canalizaciones.

# ¿Qué puedo hacer con? CodePipeline

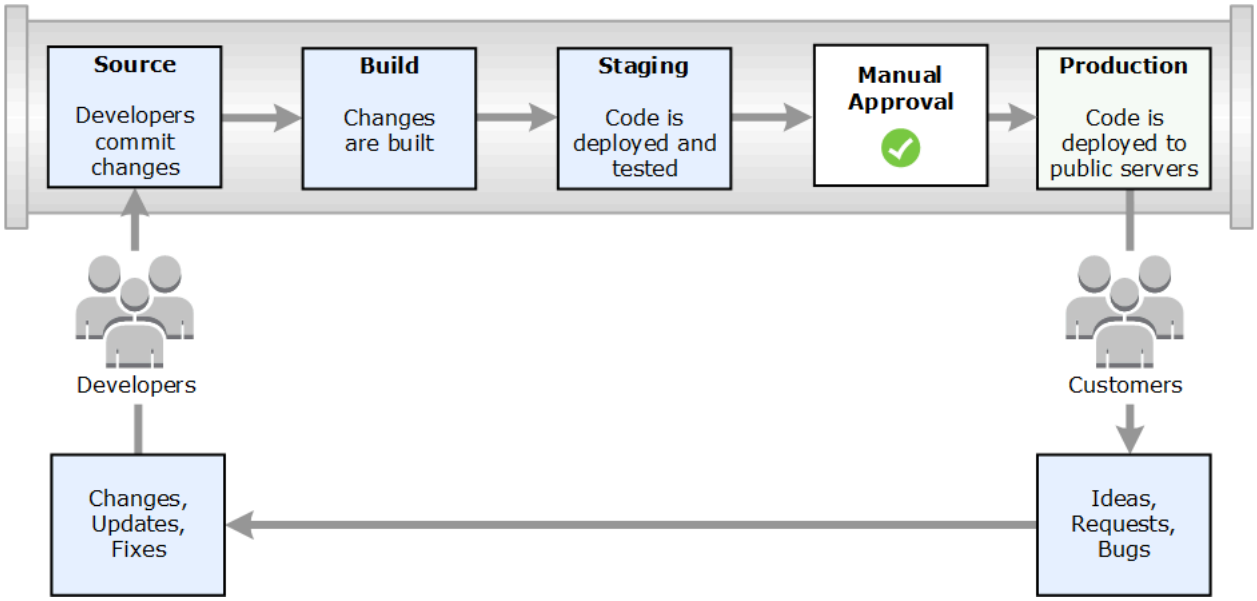
Puede usarlo CodePipeline para crear, probar e implementar automáticamente sus aplicaciones en la nube. En concreto, puede:

- Automatice sus procesos de lanzamiento: automatiza CodePipeline completamente su proceso de lanzamiento de principio a fin, comenzando desde su repositorio de origen hasta la compilación, las pruebas y la implementación. Puede impedir que los cambios pasen por una canalización incluyendo una acción de aprobación manual en cualquier etapa excepto en la etapa de código fuente. Puede lanzar el software cuando lo desee, de la forma que desee, en los sistemas de su elección y en una o varias instancias.
- Establezca un proceso de lanzamiento coherente: defina un conjunto coherente de pasos para cada cambio de código. CodePipeline ejecuta cada etapa de tu lanzamiento de acuerdo con tus criterios.
- Agilizar la entrega y mejorar la calidad: puede automatizar su proceso de lanzamiento para permitir que sus desarrolladores prueben y publiquen código de manera incremental y agilicen el lanzamiento de nuevas características para los clientes.
- Usar sus herramientas favoritas: puede incorporar sus herramientas de código fuente, compilación e implementación existentes en la canalización. Para obtener una lista completa de Servicios de AWS las herramientas de terceros que actualmente son compatibles CodePipeline, consulteIntegraciones de productos y servicios con CodePipeline.
- Ver el progreso de un simple vistazo: puede consultar el estado en tiempo real de sus canalizaciones, comprobar los detalles de todas las alertas, reintentar acciones o etapas que han producido un error, examinar las revisiones de origen usadas en la última ejecución de la canalización en cada etapa y volver a ejecutar manualmente cualquier canalización.

- Ver el detalle del historial de canalizaciones: puede ver los detalles acerca de las ejecuciones de una canalización, que incluyen las horas de inicio y finalización, la duración de la ejecución y los ID de ejecución.

# Un vistazo rápido a CodePipeline

El diagrama siguiente muestra un proceso de lanzamiento de ejemplo con CodePipeline.




En este ejemplo, cuando los desarrolladores confirman los cambios en un repositorio de código fuente, CodePipeline detecta automáticamente los cambios. Esos cambios se compilan y, si se han configurado pruebas, se ejecutan esas pruebas. Una vez completadas las pruebas, el código compilado se implementa en servidores de ensayo para su comprobación. Desde el servidor de ensayo, CodePipeline ejecuta más pruebas, como pruebas de integración o de carga. Tras completar correctamente esas pruebas y tras aprobar una acción de aprobación manual que se agregó al proceso, CodePipeline implementa el código probado y aprobado en las instancias de producción.

CodePipeline puede implementar aplicaciones en instancias EC2 mediante el uso de CodeDeploy AWS Elastic Beanstalk, o. AWS OpsWorks Stacks CodePipeline también puede implementar aplicaciones basadas en contenedores en los servicios mediante Amazon ECS. Los desarrolladores también pueden usar los puntos de integración proporcionados CodePipeline para conectar otras herramientas o servicios, incluidos los servicios de creación, los proveedores de pruebas u otros objetivos o sistemas de implementación.

Una canalización puede ser tan simple o compleja como requiera el proceso de lanzamiento.

# ¿Cómo puedo empezar CodePipeline?

Para empezar con CodePipeline:

1. Aprenda cómo CodePipeline funciona leyendo la CodePipeline conceptos sección.
2. Prepárese para usarlo CodePipeline siguiendo los pasos descritos enEmpezar con CodePipeline.
3. Experimente CodePipeline siguiendo los pasos de los CodePipeline tutoriales tutoriales.
4. CodePipeline Utilízalo para tus proyectos nuevos o existentes siguiendo los pasos que se indican enCreación de una canalización, etapas y acciones.

# CodePipeline conceptos

Modelar y configurar su proceso de publicación automatizada es más fácil si comprende los conceptos y términos utilizados en AWS CodePipeline. Aquí encontrará algunos conceptos que debe conocer sobre el uso de CodePipeline.

Para ver un ejemplo de DevOps canalización, consulteDevOps ejemplo de canalización.

Los siguientes términos se utilizan en CodePipeline:

Temas

- Canalizaciones
- Ejecuciones de canalización
- Operaciones de etapas
- Ejecuciones de acción
- Tipos de ejecución
- Tipos de acción
- Artefactos
- Revisiones de origen
- Desencadenadores
- Variables
- Condiciones
- Reglas

# Canalizaciones

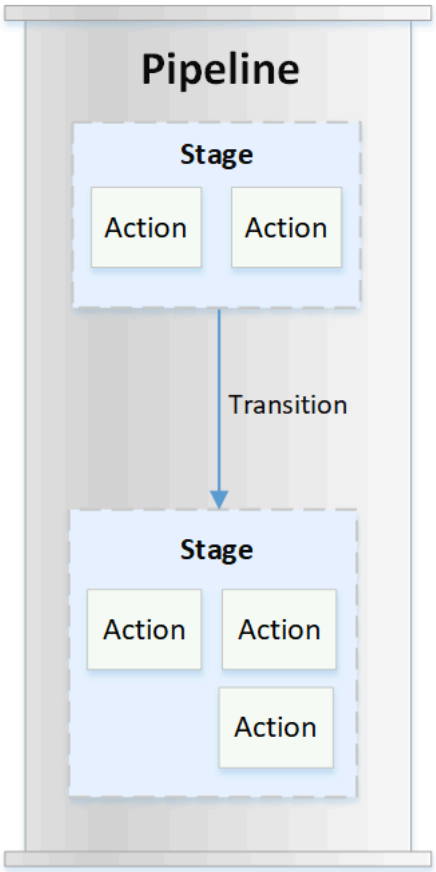
Una canalización es una construcción de flujo de trabajo que describe cómo los cambios en el software pasan por el proceso de lanzamiento. Cada canalización se compone de una serie de etapas.

## Etapas

Una etapa es una unidad lógica que puede utilizar para aislar un entorno y limitar el número de cambios simultáneos en ese entorno. Cada etapa contiene acciones que se realizan en los artefactos de la aplicación. El código fuente es un ejemplo de un artefacto. Una etapa podría ser una etapa de compilación, donde se compila el código fuente y se ejecutan pruebas. También puede ser una etapa de implementación, donde el código se implementa en entornos de tiempo de ejecución. Cada etapa se compone de una serie de acciones en serie o en paralelo.

## Transiciones

Una transición es el punto en el que la ejecución de una canalización se mueve a la siguiente etapa de la canalización. Puede deshabilitar la transición entrante de una etapa para evitar que las ejecuciones entren en esa etapa y, a continuación, puede habilitar la transición para permitir que las ejecuciones continúen. Cuando llega más de una ejecución a una transición deshabilitada, solo la ejecución más reciente continúa hasta la siguiente etapa cuando se habilita la transición. Esto significa que las ejecuciones más recientes continúan sustituyendo a las ejecuciones en espera mientras la transición está deshabilitada y, a continuación, una vez habilitada la transición, la ejecución que continúa es la ejecución sustituida.




## Acciones

Una acción es un conjunto de operaciones realizadas en el código de la aplicación y configuradas para que las acciones se ejecuten en la canalización en un punto especificado. Esto puede incluir cosas como una acción de origen a partir de un cambio de código, una acción para implementar la aplicación en instancias, etc. Por ejemplo, una etapa de implementación puede contener una acción de implementación que implemente código en un servicio informático como Amazon EC2 o AWS Lambda.

Los tipos de CodePipeline acción válidos son `source` `buildtest`,`deploy`,`approval`, `yinvoke`. Para obtener una lista de los proveedores de acciones, consulte Proveedores de acciones válidos en CodePipeline.

Las acciones se pueden ejecutar en serie o en paralelo. Para obtener información sobre las acciones en serie y paralelas en una etapa, consulte la información de `runOrder` en los requisitos de la estructura de acciones.

# Ejecuciones de canalización

Una ejecución es un conjunto de cambios lanzados por una canalización. Cada ejecución de canalización es única y tiene su propio ID. Una ejecución corresponde a un conjunto de cambios, como una confirmación combinada o una versión manual de la última confirmación. Dos ejecuciones pueden lanzar el mismo conjunto de cambios en diferentes momentos.

Mientras que una canalización puede procesar varias ejecuciones al mismo tiempo, una etapa de canalización procesa solo una ejecución a la vez. Para hacer esto, una etapa se bloquea mientras procesa una ejecución. Dos ejecuciones de una canalización no pueden ocupar la misma etapa al mismo tiempo. La ejecución que espera entrar en la fase ocupada es una ejecución entrante. Una ejecución entrante aún puede fallar, ser reemplazada o detenida manualmente. Para obtener más información sobre cómo funcionan las ejecuciones entrantes, consulte Cómo funcionan las ejecuciones entrantes.

Las ejecuciones de canalizaciones atraviesan etapas de canalización en orden. Los estados válidos para las canalizaciones son `InProgress`, `Stopping`, `Stopped`, `Succeeded`, `Superseded` y `Failed`.

Para obtener más información, consulte PipelineExecution.

## Ejecuciones detenidas

La ejecución de la canalización se puede detener manualmente para que la ejecución de la canalización en curso no continúe por la canalización. Si se detiene manualmente, una ejecución de canalización muestra un estado `Stopping` hasta que se detiene por completo. Después, muestra un estado `Stopped`. Se puede volver a intentar una ejecución de canalización `Stopped`.

Hay dos formas de detener una ejecución de canalización:

- Detener y esperar
- Detener y abandonar

Para obtener información sobre los casos de uso para detener una ejecución y detalles de la secuencia para estas opciones, consulte Cómo se detienen las ejecuciones de canalización.

## Ejecuciones con error

Si una ejecución genera un error, se detiene y no atraviesa completamente la canalización. Su estado pasa a ser el `FAILED` escenario y CodePipeline lo desbloquea. Una ejecución más reciente

puede ponerse al día y entrar en la etapa desbloqueada y bloquearla. Puede volver a intentar una ejecución fallida a menos que la ejecución fallida haya sido sustituida o no se pueda volver a intentar. Puede revertir una etapa fallida a una ejecución anterior que se realizara correctamente.

### Modos de ejecución

Para entregar el último conjunto de cambios a través de una canalización, las ejecuciones más recientes pasan y sustituyen las ejecuciones menos recientes que ya se ejecutan a través de la canalización. Cuando esto ocurre, la ejecución más reciente sustituye a la ejecución anterior. Una ejecución más reciente puede sustituir una ejecución en un punto determinado, que es el punto entre etapas. SUPERSEDED es el modo de ejecución predeterminado.

En el modo SUPERSEDED, si una ejecución está esperando para entrar en una etapa bloqueada, una ejecución más reciente podría ponerse al día y sustituirla. La ejecución más reciente ahora espera a que la etapa se desbloquee y la ejecución sustituida se detiene con un estado `SUPERSEDED`. Cuando se sustituye una ejecución de canalización, la ejecución se detiene y no atraviesa completamente la canalización. Ya no puede volver a intentar la ejecución sustituida después de que se haya reemplazado en esta etapa. Otros modos de ejecución disponibles son el modo PARALELO o EN COLA.

Para obtener más información acerca de los modos de ejecución y las etapas bloqueadas, consulte Procesamiento de ejecuciones en el modo SUPERSEDED.

## Operaciones de etapas

Cuando una ejecución de canalización se ejecuta a través de una etapa, la etapa se encontrará en el proceso de completar todas las acciones que haya en ella. Para obtener información sobre cómo funcionan las operaciones de etapas e información sobre las etapas bloqueadas, consulte Procesamiento de ejecuciones en el modo SUPERSEDED.

Los estados válidos para las etapas son `InProgress`, `Stopping`, `Stopped`, `Succeeded` y `Failed`. Puede volver a intentar una etapa fallida a menos que la etapa fallida no se pueda volver a intentar. Para obtener más información, consulte StageExecution. Puede revertir una etapa a una ejecución anterior que se realizara correctamente. Se puede configurar una etapa para que se revierta automáticamente en caso de fallo, tal y como se detalla en Configuración de la reversión de etapas. Para obtener más información, consulte RollbackStage.

## Ejecuciones de acción

Una ejecución de acción es el proceso de completar una acción configurada que opera en artefactos designados. Estos pueden ser artefactos de entrada, artefactos de salida o ambos. Por ejemplo, una acción de compilación podría ejecutar comandos de compilación en un artefacto de entrada, como compilar código fuente de la aplicación. Los detalles de ejecución de acciones incluyen un ID de ejecución de acción, el desencadenador de origen de ejecución de canalización relacionado y los artefactos de entrada y salida de la acción.

Los estados válidos para las acciones son `InProgress`, `Abandoned`, `Succeeded` o `Failed`. Para obtener más información, consulte ActionExecution.

## Tipos de ejecución

Una ejecución de etapa o canalización puede ser una ejecución estándar o una ejecución revertida.

En el caso de los tipos estándar, la ejecución tiene un identificador único y es una ejecución de canalización completa. La reversión de una canalización tiene una etapa que debe revertirse y una ejecución exitosa de la etapa como la ejecución de destino a la que se debe revertir. La ejecución de la canalización de destino se utiliza para recuperar las variables y revisiones de origen para que la etapa se vuelva a ejecutar.

## Tipos de acción

Los tipos de acción son acciones preconfiguradas que se pueden seleccionar en. CodePipeline El tipo de acción se define mediante su propietario, proveedor, versión y categoría. El tipo de acción proporciona parámetros personalizados que se utilizan para completar las tareas de acción de una canalización.

Para obtener información sobre los productos Servicios de AWS y servicios de terceros que puedes integrar en tu cartera en función del tipo de acción, consultaIntegraciones con tipos de CodePipeline acciones.

Para obtener información sobre los modelos de integración compatibles con los tipos de acción CodePipeline, consulteReferencia del modelo de integración.

Para obtener información sobre cómo los proveedores externos pueden configurar y administrar los tipos de acción en CodePipeline, consulteTrabajar con tipos de acciones.

## Artefactos

Los artefactos se refieren a la recopilación de datos, como el código fuente de la aplicación, las aplicaciones compiladas, las dependencias, los archivos de definiciones, las plantillas, etc., en los que se trabaja mediante acciones de canalización. Algunas acciones producen artefactos y otras los consumen. En una canalización, los artefactos pueden ser el conjunto de archivos en los que trabaja una acción (artefactos de entrada) o la salida actualizada de una acción completada (artefactos de salida).

Las acciones transfieren el resultado a otra acción para su posterior procesamiento mediante el depósito de artefactos de canalización. CodePipeline copia los artefactos a la tienda de artefactos, donde la acción los recoge. Para obtener más información acerca de los artefactos, consulte Artefactos de entrada y salida.

## Revisiones de origen

Cuando se realiza un cambio de código fuente, se crea una nueva versión. Una revisión de código fuente es la versión de un cambio de código fuente que desencadena una ejecución de canalización. Una ejecución procesa las revisiones de origen. En el GitHub caso de CodeCommit los repositorios, esta es la confirmación. Para acciones o buckets de S3, es la versión del objeto.

Puede iniciar la ejecución de una canalización con una revisión de código fuente, como una confirmación, que especifique. La ejecución procesará la revisión especificada y anulará la que hubiera sido la revisión utilizada para la ejecución. Para obtener más información, consulte Iniciar una canalización con una anulación de revisión de código fuente.

## Desencadenadores

Los desencadenadores son eventos que inician tu canalización. Algunos desencadenadores, como el inicio manual de una canalización, están disponibles para todos los proveedores de acciones fuente de una canalización. Algunos desencadenadores dependen del proveedor de origen de la canalización. Por ejemplo, CloudWatch los eventos se deben configurar con recursos de eventos de Amazon CloudWatch que tengan el ARN de canalización agregado como destino en la regla de eventos. Amazon CloudWatch Events es el disparador recomendado para la detección automática de cambios en las canalizaciones con una acción fuente CodeCommit o S3. Los webhooks son un tipo de desencadenador que se configura para eventos de repositorios de terceros. Por ejemplo, WebHookV2 es un tipo de activador que permite usar etiquetas de Git para iniciar canalizaciones con proveedores de fuentes externos GitHub.com, como GitHub Enterprise Server GitLab.com,

GitLab Automanaged o Bitbucket Cloud. En la configuración de la canalización, puede especificar un filtro para los desencadenadores, como las solicitudes de inserción o de extracción. Puede filtrar los eventos de inserción de código en ramificaciones, rutas de archivo o etiquetas de Git. Puedes filtrar los eventos de las solicitudes de extracción por eventos (abiertos, actualizados, cerrados), sucursales o rutas de archivos.

Para obtener más información acerca de los desencadenadores, consulte Inicie una canalización en CodePipeline. Para ver un tutorial que te explica cómo usar las etiquetas de Git como desencadenadores de tu canalización, consulte Tutorial: Usar etiquetas de Git para iniciar la canalización.

> **Important**
>
> Las canalizaciones que estén inactivas durante más de 30 días tendrán deshabilitado el sondeo para la canalización. Para obtener más información, consulta el sondeo DisabledAt en la referencia sobre la estructura de canalizaciones. Consulte los pasos para migrar una canalización que usa sondeos para que use la detección de cambios basada en eventos en Métodos de detección de cambios.

## Variables

La variable es un valor que puede utilizarse para configurar dinámicamente las acciones de su canalización. Las variables pueden declararse a nivel de canalización o emitirse mediante acciones en la canalización. Los valores de las variables se resuelven en el momento de la ejecución de la canalización y se pueden ver en el historial de ejecuciones. En el caso de las variables declaradas a nivel de canalización, puede definir los valores predeterminados en la configuración de la canalización o anularlos para una ejecución determinada. En el caso de las variables emitidas por una acción, el valor está disponible después de que la acción se complete correctamente. Para obtener más información, consulte Referencia de variables.

## Condiciones

Una condición contiene un conjunto de reglas que se evalúan. Si todas las reglas de una condición se realizan correctamente, se cumple la condición. Puede configurar las condiciones para que, cuando no se cumplan los criterios, se active el resultado especificado como, por ejemplo, que la etapa falle. Las condiciones también se denominan “puertas”, ya que permiten especificar cuándo una ejecución entrará y se ejecutará en una etapa, o bien cuándo saldrá de la etapa después de

haberse ejecutado en ella. Como analogía, esto equivale a permitir que una línea de tráfico de una carretera se junte en una puerta cerrada y, a continuación, especificar la apertura de dicha puerta para permitir el flujo de tráfico hacia una zona determinada. Los resultados para los tipos de condición incluyen el fallo de la etapa o la reversión de la etapa. Las condiciones ayudan a especificar cuándo se producen estas acciones en una etapa de canalización. Es posible anular las condiciones en tiempo de ejecución.

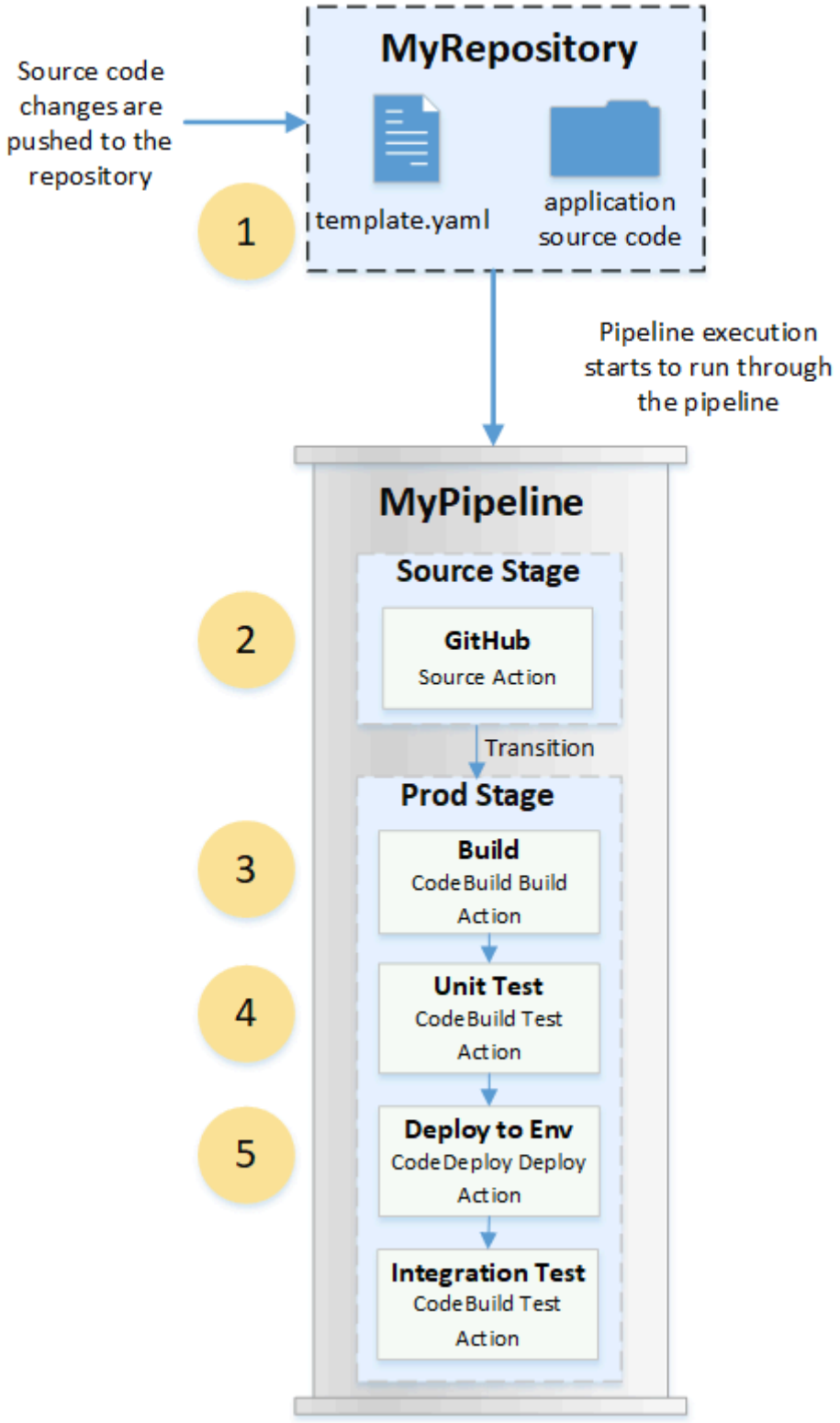
Existen tres tipos de condiciones. Las condiciones de entrada responden a la pregunta “Si se cumplen las reglas de la condición, la ejecución puede entrar en la etapa”. La etapa se bloquea cuando la ejecución entra en dicha etapa y, a continuación, se ejecutan las reglas. En condiciones de fallo, la regla se activa cuando se produce un error en una etapa, con el resultado de revertir una etapa fallida. En condiciones de éxito, la regla se activa cuando una etapa se realiza correctamente, por ejemplo, cuando se comprueba si hay alarmas en una ejecución correcta antes de continuar. Por ejemplo, si la regla determina que hay alarmas en el entorno de implementación, se anularía una etapa correcta si la CloudWatchAlarm regla determina que hay alarmas en el entorno de implementación. Para obtener más información, consulte Funcionamiento de las condiciones de las etapas.

### Reglas

Las condiciones utilizan una o más reglas preconfiguradas que se ejecutan y que realizan comprobaciones que, a su vez, activarán el resultado configurado cuando no se cumpla una condición. Por ejemplo, si se cumplen todas las reglas de una regla de condición de entrada que comprueba el estado de las alarmas y los plazos de implementación, se implementará correctamente una etapa después de que todas las comprobaciones se superen. Para obtener más información, consulte Funcionamiento de las condiciones de las etapas.

## DevOps ejemplo de canalización

Como ejemplo de DevOps canalización, una canalización de dos etapas puede tener una etapa de origen llamada  Source  y una segunda etapa llamada  Prod. En este ejemplo, la canalización está actualizando la aplicación con los últimos cambios e implementando de forma continua el último resultado. Antes de implementar la aplicación más reciente, la canalización compila y prueba la aplicación web. En este ejemplo, un grupo de desarrolladores ha configurado una plantilla de infraestructura y el código fuente de una aplicación web en un GitHub repositorio denominado. MyRepository




Por ejemplo, un desarrollador inserta una corrección en la página de índice de la aplicación web y ocurre lo siguiente:

1. El código fuente de la aplicación se mantiene en un repositorio configurado como una acción GitHub fuente en proceso. Cuando los desarrolladores envían confirmaciones al repositorio, CodePipeline detectan el cambio enviado y la ejecución del proceso comienza desde la fase de origen.
2. La acción GitHub original se completa correctamente (es decir, los cambios más recientes se han descargado y almacenado en el depósito de artefactos exclusivo de esa ejecución). Los artefactos de salida producidos por la acción de GitHub origen, que son los archivos de aplicación del repositorio, se utilizan luego como artefactos de entrada en los que trabajarán las acciones de la siguiente etapa.
3. La ejecución de la canalización pasa de la Source Stage (Etapa de código fuente) a la Prod Stage (Etapa de producción). La primera acción de la fase de producción ejecuta un proyecto de compilación creado CodeBuild y configurado como una acción de compilación en proceso. La tarea de compilación extrae una imagen de entorno de compilación y crea la aplicación web en un contenedor virtual.
4. La siguiente acción de la fase de producción es un proyecto de prueba unitario creado CodeBuild y configurado como una acción de prueba en proceso.
5. A continuación se trabaja con el código de unidad probado mediante una acción de implementación en la etapa de producción que implementa la aplicación en un entorno de producción. Una vez que la acción de implementación se complete correctamente, la última acción de la fase es un proyecto de pruebas de integración creado CodeBuild y configurado como una acción de prueba en proceso. La acción de prueba llama a scripts de shell que instalan y ejecutan una herramienta de prueba, como un comprobador de vínculos, en la aplicación web. Una vez completado con éxito, el resultado es una aplicación web compilada y un conjunto de resultados de prueba.

Los desarrolladores pueden agregar acciones a la canalización que implementen o seguir probando la aplicación después de compilarla y probarla para cada cambio.

Para obtener más información, consulte Cómo funcionan las ejecuciones de canalización.

# Cómo funcionan las ejecuciones de canalización

En esta sección se ofrece una descripción general de la forma en que CodePipeline procesa un conjunto de cambios. CodePipelinerealiza un seguimiento de cada ejecución de canalización que se inicia cuando una canalización se inicia manualmente o se realiza un cambio en el código fuente.

CodePipeline utiliza los siguientes modos de ejecución para controlar la forma en que cada ejecución avanza a lo largo de la canalización. Para obtener más información, consulte Configuración o cambio del modo de ejecución de una canalización.

- Modo SUPERSEDED: una ejecución más reciente puede adelantar a una anterior. Este es el valor predeterminado.
- Modo EN COLA: las ejecuciones se procesan de una en una según el orden en que estén en cola. Esto requiere el tipo de canalización V2.
- Modo PARALELO: en el modo PARALELO, las ejecuciones se ejecutan de forma simultánea e independiente unas de otras. Las ejecuciones no esperan a que se completen otras ejecuciones para iniciarse o finalizar. Esto requiere el tipo de canalización V2.

> **Important**
>
> Para las canalizaciones en modo PARALELO, la reversión de etapas no está disponible. Del mismo modo, las condiciones de error con un tipo de resultado de reversión no se pueden agregar a una canalización en modo PARALELO.

## Cómo se inician las ejecuciones de canalización

Puede desencadenar una ejecución cuando cambie el código fuente o cuando inicie manualmente la canalización. También puedes activar una ejecución mediante una regla de Amazon CloudWatch Events que programes. Por ejemplo, cuando se envía un cambio de código fuente a un repositorio configurado como acción de origen de la canalización, la canalización detecta el cambio e inicia una ejecución.

> **Note**
>
> Si una canalización contiene varias acciones de origen, todas ellas se vuelven a ejecutar aunque solo se detecte un cambio para una de ellas.

## Cómo se procesan las revisiones de origen en ejecuciones de canalización

Para cada ejecución de canalización que se inicie con cambios en el código fuente (revisiones de origen), las revisiones de origen se determinan de la siguiente manera.

- En el caso de las canalizaciones con una CodeCommit fuente, el HEAD se clona CodePipeline en el momento en que se envía la confirmación. Por ejemplo, se inserta una confirmación, lo que inicia la canalización para la ejecución 1. En el momento en que se inserta una segunda confirmación, se inicia la canalización para la ejecución 2.

  > **Note**
  >
  > En el caso de las canalizaciones en modo PARALELO con una CodeCommit fuente, independientemente de la confirmación que haya desencadenado la ejecución de la canalización, la acción de origen siempre clonará el HEAD en el momento en que se inicie. Para obtener más información, consulte CodeCommit o las revisiones del código fuente de S3 en modo paralelo podrían no coincidir con el EventBridge evento.

- En el caso de las canalizaciones con una fuente de S3, se EventBridge usa el evento de actualización del bucket de S3. Por ejemplo, el evento se genera cuando se actualiza un archivo en el bucket de origen, lo que inicia la canalización para la ejecución 1. En el momento en que se realiza el evento de una segunda actualización del bucket, se inicia la canalización para la ejecución 2.

  > **Note**
  >
  > En el caso de las canalizaciones en modo PARALELO con un origen de S3, independientemente de la etiqueta de imagen que haya desencadenado la ejecución, la acción de origen siempre se iniciará con la etiqueta de imagen más reciente. Para obtener más información, consulte CodeCommit o las revisiones del código fuente de S3 en modo paralelo podrían no coincidir con el EventBridge evento.

- En el caso de las canalizaciones con una fuente de conexiones, como Bitbucket, el HEAD se clona CodePipeline en el momento en que se envía la confirmación. Por ejemplo, para una canalización en modo PARALELO, se inserta una confirmación, que inicia la canalización para la ejecución 1, y la segunda ejecución de la canalización utiliza la segunda confirmación.

## Cómo funcionan las anulaciones de código fuente con el transformador de entrada EventBridge

Puede usar las anulaciones para iniciar una canalización con un identificador de revisión de código fuente específico que proporcione para la ejecución de la canalización. Por ejemplo, si quieres iniciar

una canalización que procese un identificador de confirmación específico de tu CodeCommit fuente, puedes añadir el identificador de confirmación como anulación al iniciar la canalización.

Existen cuatro tipos de revisión de código fuente para `revisionType`:

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

> **Note**
>
> En el caso de las revisiones de código fuente de los tipos `COMMIT_ID` y `IMAGE_DIGEST`, el identificador de revisión de código fuente se aplica a todo el contenido del repositorio, en todas las ramificaciones.

> **Note**
>
> En el caso de las revisiones de la fuente `S3_OBJECT_VERSION_ID` y `S3_OBJECT_KEY` los tipos, puedes usar cualquiera de los dos tipos de forma independiente o combinarlos para reemplazar la fuente con un ID de versión específico ObjectKey . Para `S3_OBJECT_KEY`, el parámetro de configuración `AllowOverrideForS3ObjectKey` debe estar establecido en `true`. Para obtener más información sobre los parámetros de configuración del origen de S3, consulte Parámetros de configuración.

Puede especificar las anulaciones de la fuente utilizando el transformador de entrada in. EventBridge Utilice el transformador de entrada para pasar los datos de una de las siguientes maneras:

- Puede usar el transformador de entrada para pasar los datos como parámetros JSON.
- Puede usar el transformador de entrada para pasar variables de canalización.

Para ver ejemplos de cómo pasar los datos como parámetros JSON, consulteAcciones y recursos de origen de Amazon ECR EventBridge, Conectarse a las acciones de origen de Amazon S3 que

utilizan EventBridge y AWS CloudTrail para S3 y CodeCommit acciones de origen y EventBridge para CodeCommit.

## Cómo se detienen las ejecuciones de canalización

Para utilizar la consola para detener una ejecución de canalización, puede seleccionar Stop execution (Detener la ejecución) en la página de visualización de canalización, en la página de historial de ejecución o en la página de historial detallado. Para utilizar la CLI para detener una ejecución de canalización, utilice el comando `stop-pipeline-execution`. Para obtener más información, consulte Detener la ejecución de una canalización en CodePipeline.

Hay dos formas de detener una ejecución de canalización:

- Detener y esperar: todas las ejecuciones de acción en curso pueden completarse y las acciones posteriores no se inician. La ejecución de canalización no continúa en etapas posteriores. No puede utilizar esta opción en una ejecución que ya se encuentra en un estado `Stopping`.
- Detener y abandonar: todas las ejecuciones de acción en curso se abandonan y no se completan, y las acciones posteriores no se inician. La ejecución de canalización no continúa en etapas posteriores. Puede utilizar esta opción en una ejecución que ya se encuentra en un estado `Stopping`.

> **Note**
>
> Esta opción puede conducir a tareas con error o fuera de secuencia.

Cada opción da lugar a una secuencia diferente de fases de canalización y ejecución de acciones, como se indica a continuación.

Opción 1: Detener y esperar

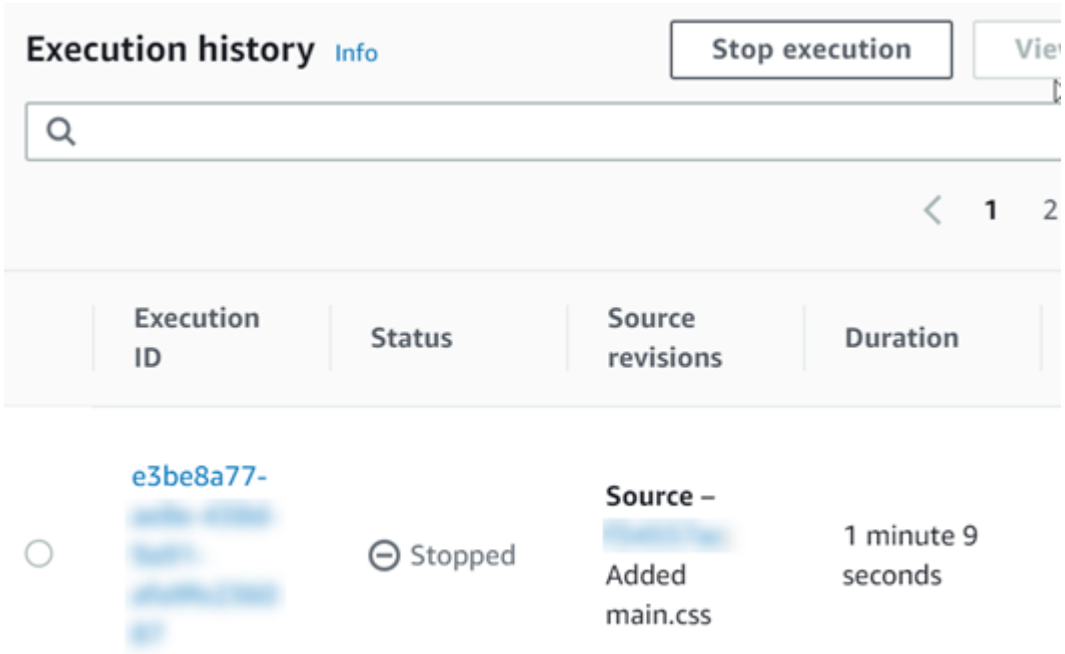
Cuando elige detener y esperar, la ejecución seleccionada continúa hasta que se completen las acciones en curso. Por ejemplo, la siguiente ejecución de canalización se ha detenido mientras la acción de compilación estaba en curso.

1. En la vista de canalización, se muestra el banner del mensaje de realización correcta y la acción de compilación continúa hasta que se completa. El estado de ejecución de canalización es Stopping (Deteniéndose).

En la vista de historial, el estado de las acciones en curso, como la acción de compilación, es In progress (En curso) hasta que se completa la acción de compilación. Mientras las acciones están en curso, el estado de ejecución de canalización es Stopping (Deteniéndose).

2. La ejecución se detiene cuando el proceso de detención se completa. Si la acción de compilación se completa correctamente, su estado es Succeeded (Correcto) y la ejecución de canalización muestra un estado de Stopping (Deteniéndose). Las acciones posteriores no se inician. El botón Retry (Reintentar) está habilitado.

   En la vista de historial, el estado de ejecución es Stopped (Detenido) después de completar la acción en curso.



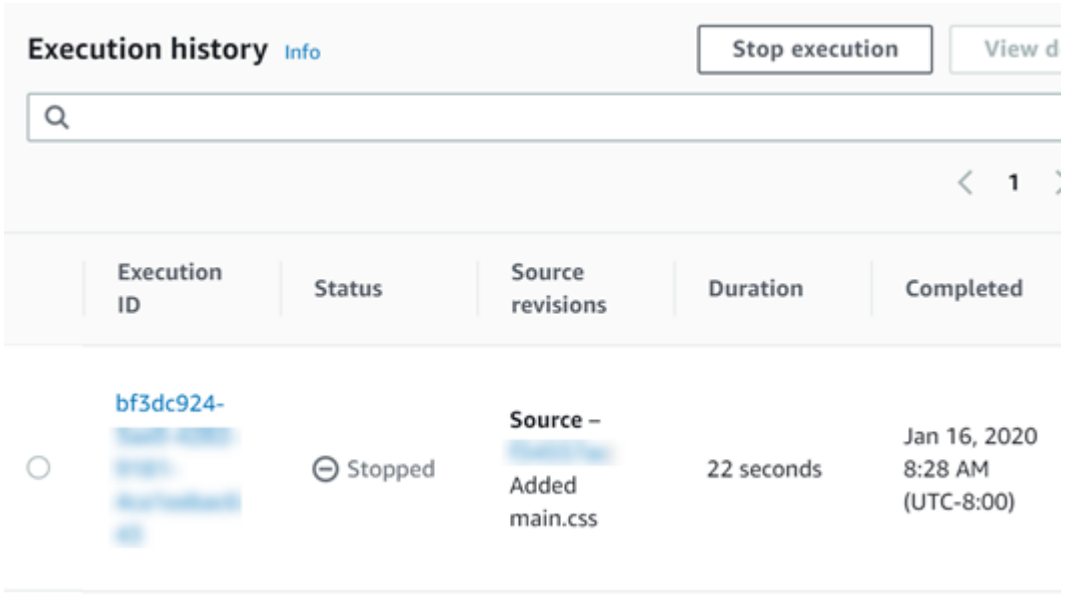

Opción 2: Detener y abandonar

Cuando elige detener y abandonar, la ejecución seleccionada no espera a que se completen las acciones en curso. Las acciones se abandonan. Por ejemplo, la siguiente ejecución de canalización se ha detenido y abandonado mientras la acción de compilación estaba en curso.

1. En la vista de canalización, se muestra el mensaje de banner de realización correcta, la acción de compilación muestra un estado de In progress (En curso) y la ejecución de canalización muestra un estado de Stopping (Deteniéndose).
2. Después de que la ejecución de canalización se detiene, la acción de compilación muestra un estado de Abandoned (Abandonado) y la ejecución de canalización muestra un estado de

Stopped (Detenido). Las acciones posteriores no se inician. El botón Retry (Reintentar) está habilitado.

3. En la vista de historial, el estado de ejecución es Stopped (Detenido).




Casos de uso para detener una ejecución de canalización

Le recomendamos que utilice la opción de detener y esperar para detener una ejecución de canalización. Esta opción es más segura porque evita las posibles tareas fallidas o fuera de secuencia en su canalización. Cuando se abandona una acción en CodePipeline, el proveedor de la acción continúa con las tareas relacionadas con la acción. En el caso de una CloudFormation acción, se abandona la acción de despliegue en proceso, pero la actualización de la pila puede continuar y provocar un error en la actualización.

Como ejemplo de acciones abandonadas que pueden dar lugar a tareas fuera de secuencia, si va a implementar un archivo grande (1 GB) mediante una acción de implementación de S3 y decide detener y abandonar la acción mientras la implementación ya está en curso, la acción se abandona en Amazon S3 CodePipeline, pero continúa en Amazon S3. Amazon S3 no encuentra ninguna instrucción para cancelar la carga. A continuación, si inicia una nueva ejecución de canalización con un archivo muy pequeño, ahora hay dos implementaciones en curso. Debido a que el tamaño del archivo de la nueva ejecución es pequeño, la nueva implementación se completa mientras que la implementación anterior aún se está cargando. Cuando finaliza la implementación anterior, el archivo nuevo se sobrescribe con el anterior.

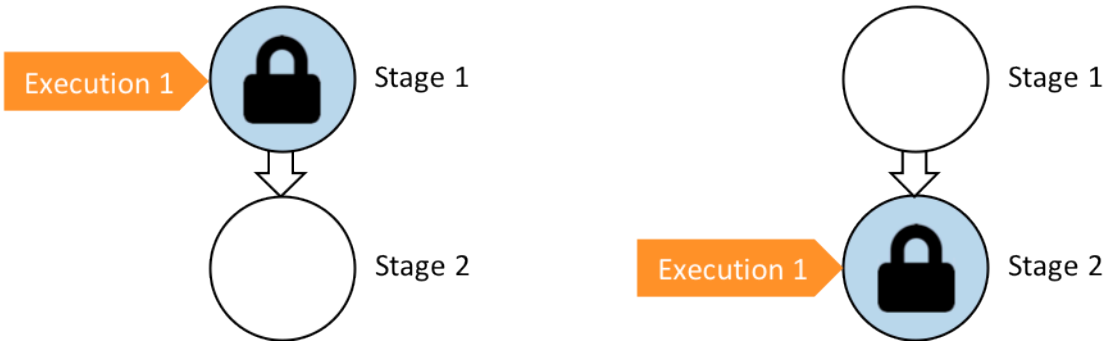
Es posible que desee utilizar la opción de detener y abandonar en el caso en que tenga una acción personalizada. Por ejemplo, puede abandonar una acción personalizada con un trabajo que no necesita terminarse antes de iniciar una nueva ejecución con una corrección de errores.

## Procesamiento de ejecuciones en el modo SUPERSEDED

El modo predeterminado para procesar las ejecuciones es el modo SUPERSEDED. Una ejecución consta de un conjunto de cambios recogidos y procesados por la ejecución. Las canalizaciones pueden procesar varias ejecuciones al mismo tiempo. Cada ejecución se ejecuta a través de la canalización por separado. La canalización procesa cada ejecución en orden y puede reemplazar una ejecución anterior por otra posterior. Las siguientes reglas se utilizan para procesar ejecuciones en una canalización bajo el modo SUPERSEDED.

Regla 1: las etapas se bloquean cuando se está procesando una ejecución

Dado que cada etapa puede procesar solo una ejecución a la vez, la etapa se bloquea mientras está en curso. Cuando la ejecución completa una etapa, pasa a la siguiente etapa de la canalización.

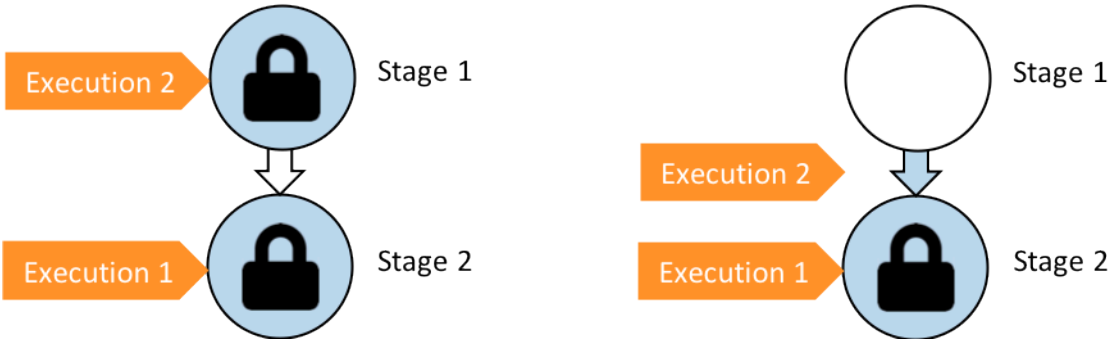



Antes: Stage 1 is locked as Execution 1 enters. Después: Stage 2 is locked as Execution 1 enters.

Regla 2: ejecuciones posteriores esperan a que se desbloquee la etapa

Mientras una etapa está bloqueada, las ejecuciones en espera se llevan a cabo delante de la etapa bloqueada. Todas las acciones configuradas para una etapa se deben completar correctamente para que la etapa se considere completada Un error libera el bloqueo en la etapa. Cuando se detiene una ejecución, esta no continúa en una etapa y la etapa se desbloquea.

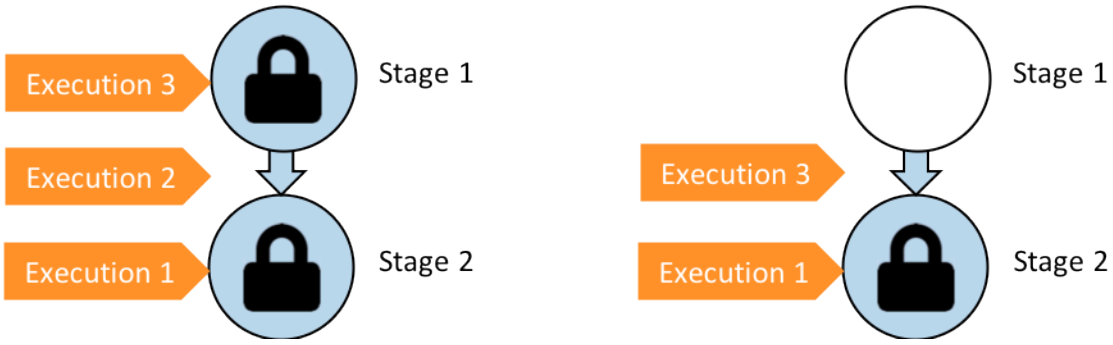
> **Note**
>
> Antes de detener una ejecución, le recomendamos que desactive la transición antes de la etapa. De esta forma, cuando la etapa se desbloquea debido a la ejecución detenida, la etapa no acepta una ejecución de canalización posterior.




Antes: Stage 2 is locked as Execution 1 enters. Después: Execution 2 exits Stage 1 and waits between stages.

Regla 3: las ejecuciones en espera se sustituyen por ejecuciones más recientes

Las ejecuciones solo se sustituyen entre etapas. Una etapa bloqueada tiene una ejecución delante de la etapa a la espera de que la etapa se complete. Una ejecución más reciente adelanta a una ejecución en espera y continúa a la siguiente etapa tan pronto como se desbloquea la etapa. La ejecución sustituida no continúa. En este ejemplo, Ejecución 2 ha sido sustituido por Ejecución 3 mientras esperaba la etapa bloqueada. La ejecución 3 entra en la etapa siguiente.




Antes: la ejecución 2 espera entre etapas mientras que la ejecución 3 entra en la etapa 1. Después: la ejecución 3 sale de la etapa 1. La ejecución 2 se sustituye por la ejecución 3.

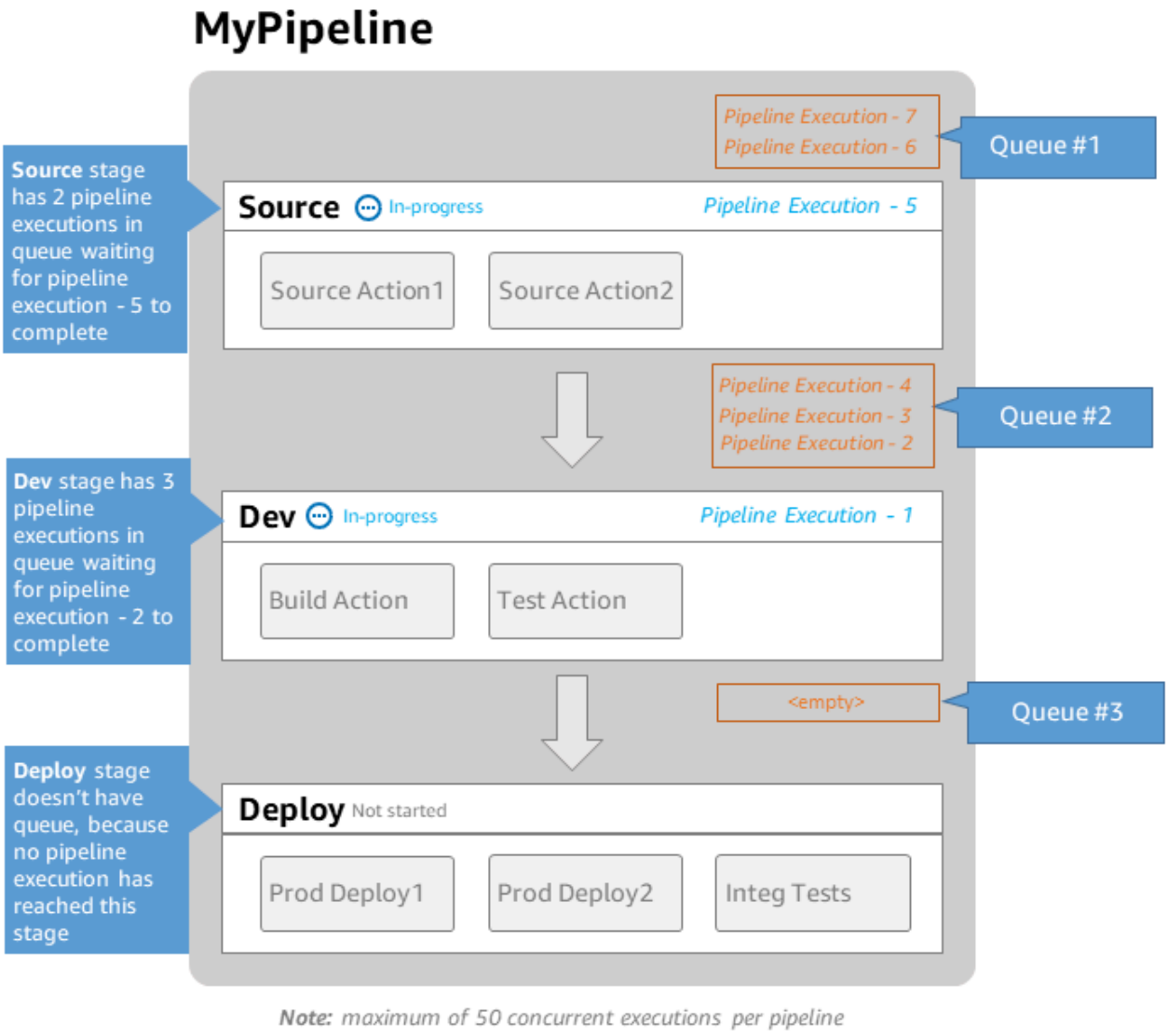
Para obtener más información sobre qué aspectos hay que tener en cuenta a la hora de ver y cambiar de modo de ejecución, consulte Configuración o cambio del modo de ejecución de una canalización. Para obtener más información acerca de las cuotas con modos de ejecución, consulte Cuotas en AWS CodePipeline.

## Procesamiento de ejecuciones en modo EN COLA

En el caso de las canalizaciones en modo EN COLA, las etapas se bloquean mientras se procesa una ejecución; sin embargo, las ejecuciones en espera no adelantan a las que ya se hayan iniciado.

Las ejecuciones en espera se agrupan en los puntos de entrada a etapas bloqueadas en el orden en que llegan a la etapa, formando una cola de ejecuciones en espera. Con el modo EN COLA, puede tener varias colas en la misma canalización. Cuando una ejecución en cola entra en una etapa, dicha etapa se bloquea y no deja entrar a otras ejecuciones. Este comportamiento sigue siendo el mismo que en el modo SUPERSEDED. Cuando la ejecución finaliza la etapa, dicha etapa queda desbloqueada y lista para la siguiente ejecución.

El siguiente diagrama muestra cómo las etapas en una canalización en modo EN COLA procesa las ejecuciones. Por ejemplo, mientras la etapa de Origen procesa la ejecución 5, las ejecuciones 6 y 7 forman la Cola 1 y esperan en el punto de entrada de dicha etapa. La siguiente ejecución de la cola se procesará una vez que se desbloquee la etapa.



Para obtener más información sobre qué aspectos hay que tener en cuenta a la hora de ver y cambiar de modo de ejecución, consulte Configuración o cambio del modo de ejecución de una canalización. Para obtener más información acerca de las cuotas con modos de ejecución, consulte Cuotas en AWS CodePipeline.

## Procesamiento de ejecuciones en modo PARALELO

En el caso de las canalizaciones en modo paralelo, las ejecuciones son independientes entre sí y no hay que esperar a que finalicen las demás para empezar. En este modo, no hay ninguna cola. Para ver las ejecuciones paralelas en la canalización, utilice la vista del historial de ejecuciones.

Utilice el modo PARALELO en entornos de desarrollo en los que cada característica tenga su propia ramificación de características y se implemente en destinos que otros usuarios no comparten.

Para obtener más información sobre qué aspectos hay que tener en cuenta a la hora de ver y cambiar de modo de ejecución, consulte Configuración o cambio del modo de ejecución de una canalización. Para obtener más información acerca de las cuotas con modos de ejecución, consulte Cuotas en AWS CodePipeline.

## Gestión del flujo de canalización

El flujo de ejecuciones de canalización puede controlarse mediante:

- Una transición, que controla el flujo de ejecuciones en la etapa. Las transiciones se pueden habilitar o deshabilitar. Cuando una transición está deshabilitada, las ejecuciones de canalizaciones no pueden entrar en la etapa. La ejecución en canalización que espera entrar en una etapa en la que la transición está deshabilitada se denomina ejecución entrante. Después de habilitar la transición, una ejecución entrante se moverá a la etapa y la bloqueará.

  Similar a las ejecuciones en espera de una etapa bloqueada, cuando una transición está deshabilitada, la ejecución que espera entrar en la etapa todavía puede ser sustituida por una nueva ejecución. Cuando se vuelve a habilitar una transición deshabilitada, entra en la etapa la ejecución más reciente, incluida la que sustituyó las ejecuciones anteriores mientras la transición estaba deshabilitada.
- Una acción de aprobación, que impide que una canalización pase a la siguiente acción hasta que se conceda permiso (por ejemplo, mediante la aprobación manual de una identidad autorizada). Puede utilizar una acción de aprobación si desea controlar el tiempo en que una canalización pasa a una etapa de producción final, por ejemplo.

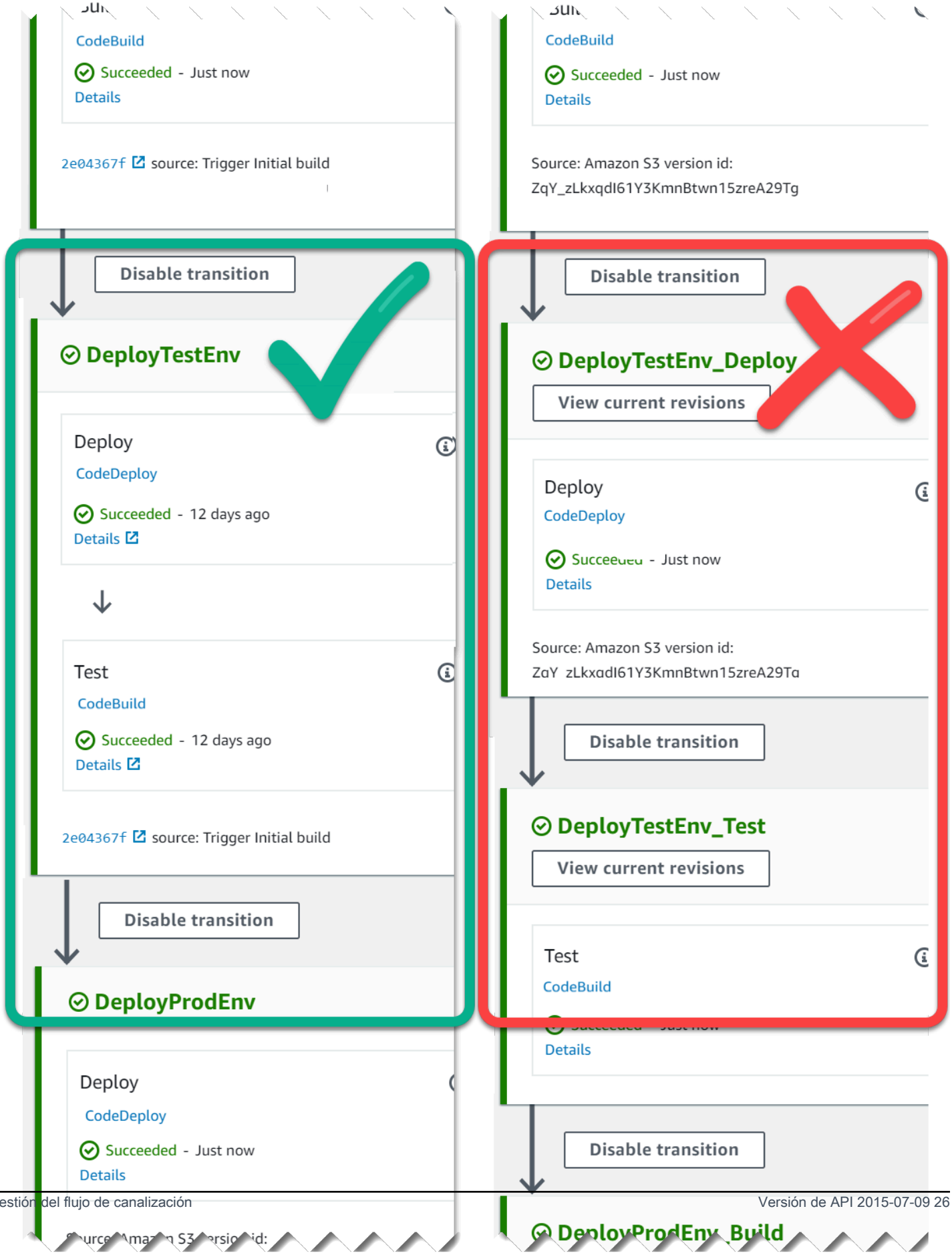
> **Note**
>
> Una etapa con una acción de aprobación se bloquea hasta que se apruebe o rechace la acción de aprobación o se haya agotado el tiempo de espera. Una acción de aprobación con tiempo de espera se procesa de la misma manera que una acción con error.

- Un error, cuando una acción en una etapa no se completa correctamente. La revisión no pasa a la siguiente acción de la etapa o a la siguiente etapa de la canalización. Puede ocurrir lo siguiente:
  - Se repite manualmente la etapa que contiene las acciones con error. Esto reanuda la ejecución (vuelve a intentar las acciones fallidas y, si tienen éxito, continúa en). stage/pipeline
  - Otra ejecución entra en la etapa con error y sustituye a la ejecución con error. En este punto, la ejecución con error no se puede volver a intentar.

## Estructura recomendada de canalizaciones

Al decidir cómo debe fluir un cambio de código a través de su canalización, lo mejor es agrupar acciones relacionadas dentro de una etapa para que, cuando la etapa se bloquee, todas las acciones procesen la misma ejecución. Puede crear una etapa para cada entorno de aplicación o zona de disponibilidad, etc. Región de AWS Una canalización con demasiadas etapas (es decir, demasiado detallada) puede permitir demasiados cambios simultáneos, mientras que una canalización con muchas acciones en una etapa grande (demasiado amplia) puede tardar demasiado en lanzar un cambio.

Por ejemplo, una acción de prueba después de una acción de implementación en la misma etapa está garantizada para probar el mismo cambio que se ha implementado. En este ejemplo, se implementa un cambio en un entorno de prueba, se prueba y, a continuación, se implementa el último cambio desde el entorno de prueba en un entorno de producción. En el ejemplo recomendado, el entorno de prueba y el entorno de producción son etapas separadas.


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

Izquierda: acciones relacionadas de prueba, implementación y aprobación agrupadas (recomendado). Derecha: acciones relacionadas en etapas separadas (no recomendado).

### Cómo funcionan las ejecuciones entrantes

Una ejecución entrante es una ejecución que espera a que una etapa, transición o acción no disponible esté disponible antes de continuar. Es posible que la siguiente etapa, transición o acción no esté disponible porque:

- Otra ejecución ya ha entrado en la siguiente etapa y la ha bloqueado.
- La transición para entrar en la siguiente etapa está desactivada.

Puede deshabilitar una transición para detener una ejecución entrante si quiere controlar si una ejecución actual tiene tiempo de completarse en las etapas posteriores o si desea detener todas las acciones en un momento determinado. Para determinar si tiene una ejecución entrante, puede ver la canalización en la consola o ver el resultado desde el comando get-pipeline-state.

Las ejecuciones entrantes se realizan con las siguientes consideraciones:

- En cuanto la acción, la transición o la fase bloqueada estén disponibles, la ejecución entrante en curso entrará en la fase y continuará su proceso.
- Mientras la ejecución entrante está en espera, se puede detener manualmente. Una ejecución entrante puede tener un estado `InProgress`, `Stopped` o `Failed`.
- Cuando una ejecución entrante se detiene o se produce un error, no se puede volver a intentar porque no hay acciones fallidas que reintentar. Cuando se detiene una ejecución entrante y se habilita la transición, la ejecución entrante detenida no continúa en la fase.

Puede ver o detener una ejecución entrante.

## Artefactos de entrada y salida

CodePipeline se integra con las herramientas de desarrollo para comprobar si hay cambios en el código y, a continuación, compilarlo e implementarlo en todas las etapas del proceso de entrega continua. Los artefactos son los archivos en los que se trabaja mediante acciones en proceso, como archivos o carpetas con código de aplicación, archivos de páginas de índice, scripts, etc. Por ejemplo, el artefacto de la acción de origen de Amazon S3 es un nombre de archivo (o ruta de

archivo) en el que se proporcionan los archivos de código fuente de la aplicación para la acción de origen de canalización. Los archivos se proporcionan como un archivo ZIP, como el siguiente ejemplo de nombre de artefacto: `SampleApp_Windows.zip`. El artefacto de salida de la acción de origen, los archivos de código fuente de la aplicación, es el artefacto de salida de la acción de origen y es el artefacto de entrada para la siguiente acción, como una acción de compilación. Como otro ejemplo, una acción de compilación puede ejecutar comandos de compilación que compilan el código fuente de la aplicación para un artefacto de entrada, que son los archivos de código fuente de la aplicación. Consulta la página de referencia de configuración de acciones para ver una acción específica y obtener más información sobre los parámetros del artefacto, AWS CodeBuild referencia de acciones de creación y prueba por ejemplo, los de la CodeBuild acción.

Las acciones utilizan artefactos de entrada y salida que se almacenan en el depósito de artefactos de Amazon S3 que eligió al crear la canalización. CodePipeline comprime y transfiere los archivos en busca de artefactos de entrada o salida, según corresponda al tipo de acción de la etapa.

> **Note**
>
> El bucket de artefactos no es el mismo depósito que el bucket utilizado como ubicación del archivo de origen para una canalización en la que la acción de origen elegida es S3.

Por ejemplo:

1. CodePipeline activa la canalización para que se ejecute cuando hay una confirmación en el repositorio de origen, proporcionando el artefacto de salida (cualquier archivo que se vaya a crear) de la fase de origen.
2. El artefacto de salida (los archivos que compilar) del paso anterior se recibe como un artefacto de entrada en la etapa de compilación. Un artefacto de salida (la aplicación de compilación) de la etapa de compilación puede ser una aplicación actualizada o una imagen de Docker actualizada compilada en un contenedor.
3. El artefacto de salida del paso anterior (la aplicación de compilación) se acepta como artefacto de entrada de la etapa Implementación, como los entornos de ensayo o de producción de en la Nube de AWS. Puede implementar aplicaciones en una flota de implementación, o puede implementar aplicaciones basadas en contenedores en tareas que se ejecutan en clústeres de ECS.

Cuando crea o edita una acción, designa el artefacto o artefactos de entrada y salida para la acción. Por ejemplo, para una canalización de dos etapas con una etapa origen y Implementar en Editar

acción, elija el nombre de artefacto de la acción de origen para el artefacto de entrada para la acción de implementación.

- Cuando usas la consola para crear tu primera canalización, CodePipeline crea un bucket de Amazon S3 en la misma canalización Cuenta de AWS y almacena Región de AWS los elementos de todas las canalizaciones. Cada vez que utilice la consola para crear otra canalización en esa región, CodePipeline crea una carpeta para esa canalización en el bucket. Utiliza esa carpeta para almacenar los artefactos de la canalización conforme se ejecuta el proceso de lanzamiento automático. Este bucket se denomina codepipeline- *region* -*12345EXAMPLE-1a2b-3c4d-67890EXAMPLE*, que *region* es la AWS región en la que creaste el pipeline, y *12345EXAMPLE-1a2b-3c4d-67890EXAMPLE* es un valor aleatorio basado en un identificador único universal (UUID) que garantiza que el nombre del bucket sea único.

  > **Note**
  >
  > Si ya tienes un bucket que empieza por codepipeline - region, en la región en la que vas a crear el canalización, CodePipeline usa ese bucket como el bucket predeterminado. También sigue un orden lexicográfico; por ejemplo, se elige codepipeline-region-abcexample antes de codepipeline-region-defexample.

  CodePipeline trunca los nombres de los artefactos, lo que puede hacer que algunos nombres de cubos parezcan similares. Aunque el nombre del artefacto parezca truncado, se CodePipeline asigna al depósito de artefactos de forma que no se vea afectado por los artefactos con nombres truncados. La canalización puede funcionar con normalidad. Esto no supone un problema con la carpeta ni con los artefactos. Los nombres de las canalizaciones tienen una longitud máxima de 100 caracteres. Aunque el nombre de la carpeta de artefactos parezca estar acortado, sigue siendo único para la canalización.

  Al crear o editar un oleoducto, debes tener un depósito de artefactos en proceso Cuenta de AWS y Región de AWS debes tener un depósito de artefactos por región en la que planees ejecutar una acción. Si utiliza la consola para crear una canalización o acciones entre regiones, los buckets de artefactos predeterminados los configura CodePipeline en las regiones en las que estén las acciones.

  Si lo usa AWS CLI para crear una canalización, puede almacenar los artefactos de esa canalización en cualquier bucket de Amazon S3, siempre Cuenta de AWS y Región de AWS cuando ese bucket esté en el mismo lugar que la canalización. Podría utilizar este método si le

preocupa sobrepasar los límites de los buckets de Amazon S3 permitidos para su cuenta. Si lo usa AWS CLI para crear o editar una canalización y agrega una acción interregional (una acción con un AWS proveedor en una región diferente a la de su canalización), debe proporcionar un bucket de artefactos para cada región adicional en la que planee ejecutar una acción.

- Cada acción tiene un tipo. En función del tipo, la acción podría tener alguno de estos elementos o ambos:
  - Un artefacto de entrada, que es el artefacto utilizado durante el curso de ejecución de la acción
  - Un artefacto de salida, que es la salida de la acción

  Cada artefacto de salida de la canalización debe tener un nombre único. Cada artefacto de entrada de una acción debe coincidir con el artefacto de salida de una acción anterior en la canalización, tanto si la acción es inmediatamente anterior a la acción en una etapa como si se ha ejecutado varias etapas antes.

  Varias acciones pueden trabajan en el mismo artefacto.

# Funcionamiento de las condiciones de las etapas

Para cada condición que especifique una regla, se ejecuta dicha regla. Si la condición no se cumple, la etapa obtiene el resultado especificado. Si lo desea, como parte de la regla, también puede especificar qué recursos se CodePipeline deben utilizar en determinados casos. Por ejemplo, la `CloudWatchAlarm` regla utilizará un recurso de CloudWatch alarma para comprobar la condición.

Es posible que una condición coincida con varias reglas, y cada regla puede especificar uno de los tres proveedores.

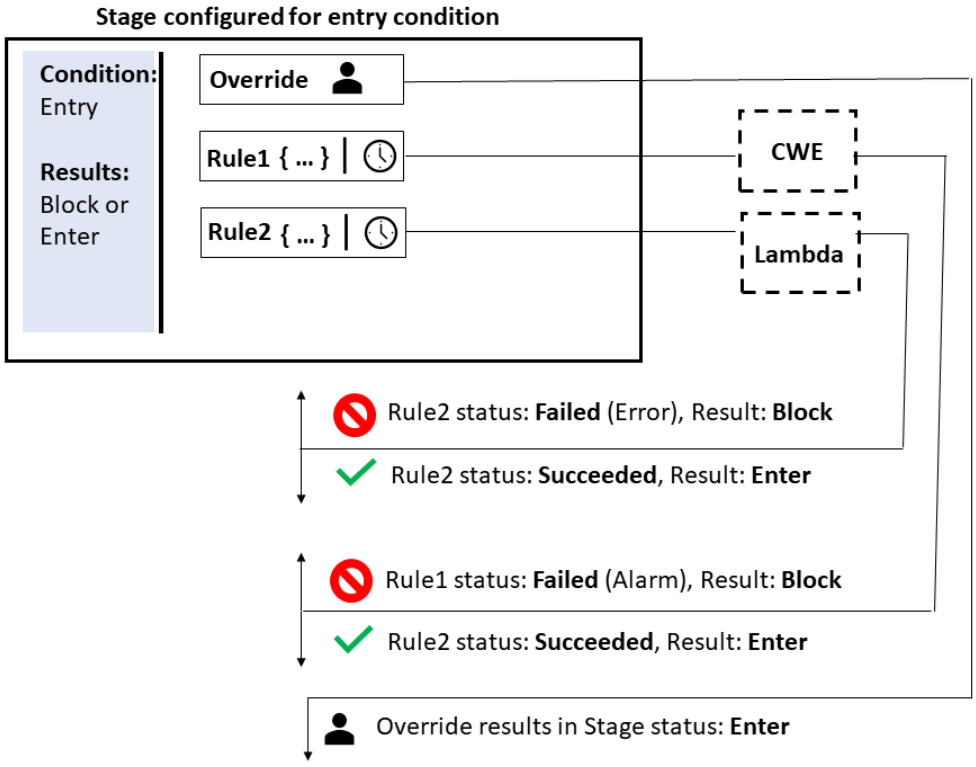
El flujo de alto nivel para crear condiciones es el siguiente:

1. Elija el tipo de condición entre los tipos de condición disponibles en CodePipeline. Por ejemplo, utilice el tipo de condición de éxito para configurar una etapa de modo que, una vez finalizada correctamente, se pueda utilizar un conjunto de reglas para realizar comprobaciones antes de continuar.
2. Elija la regla . Por ejemplo, la regla `CloudWatchAlarm` comprobará si hay alarmas, y utiliza EB para comprobar si hay un umbral de alarma preconfigurado. Si la comprobación se realiza correctamente y la alarma está por debajo del umbral, la etapa puede continuar.
3. Configure el resultado, como por ejemplo una reversión que se utilizaría si la regla fallara.

Las condiciones se utilizan para tipos específicos de expresiones, y cada una tiene opciones específicas de resultados disponibles, de la siguiente manera:

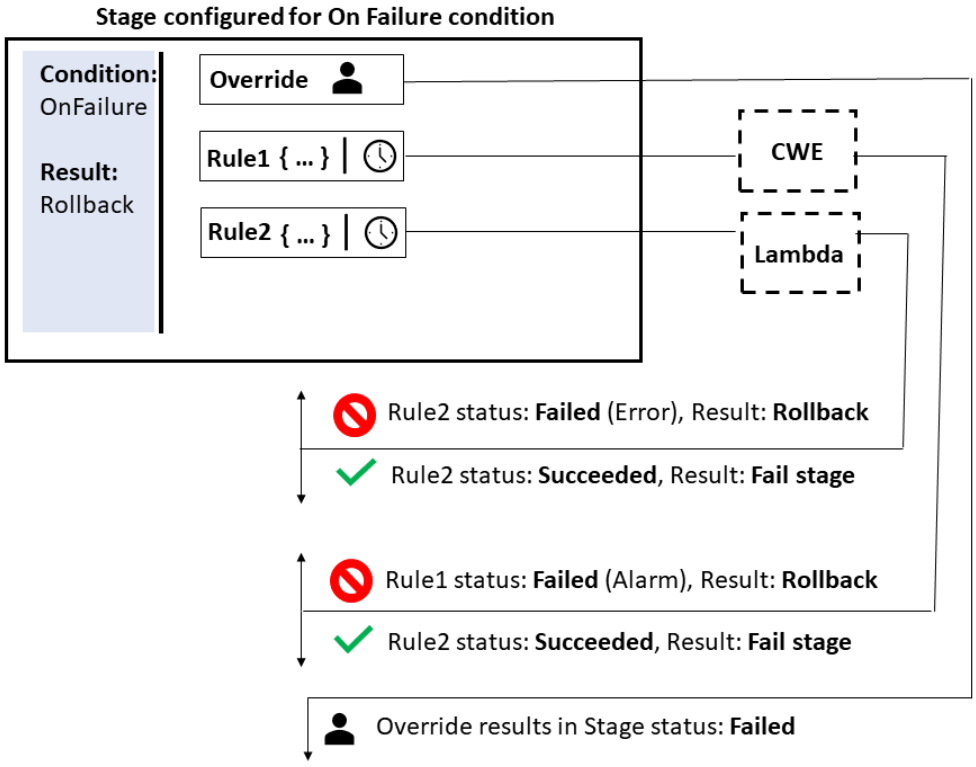
- Entrada: las condiciones para realizar comprobaciones que, si se cumplen, permiten la entrada a una etapa. Las reglas se activan con las siguientes opciones de resultado: Fallar u Omitir.
- Fallo: las condiciones para realizar las comprobaciones de la etapa en caso de que se produzca un error. Las reglas se activan con la siguiente opción de resultado: Reversión.
- Éxito: las condiciones para realizar las comprobaciones de la etapa en caso de éxito. Las reglas se activan con las siguientes opciones de resultado: Reversión o Fallar.

El siguiente diagrama muestra un ejemplo de flujo para el tipo de condición de entrada CodePipeline. Las condiciones responden a la pregunta: ¿Qué debería ocurrir si no se cumple la condición, es decir, si se produce un error en una regla? En el siguiente flujo, se configura una condición de entrada con una LambdaInvoke regla y una `CloudWatchAlarm` regla. Si se produjera un error en la regla, se activaría el resultado configurado, como por ejemplo, “Error”.

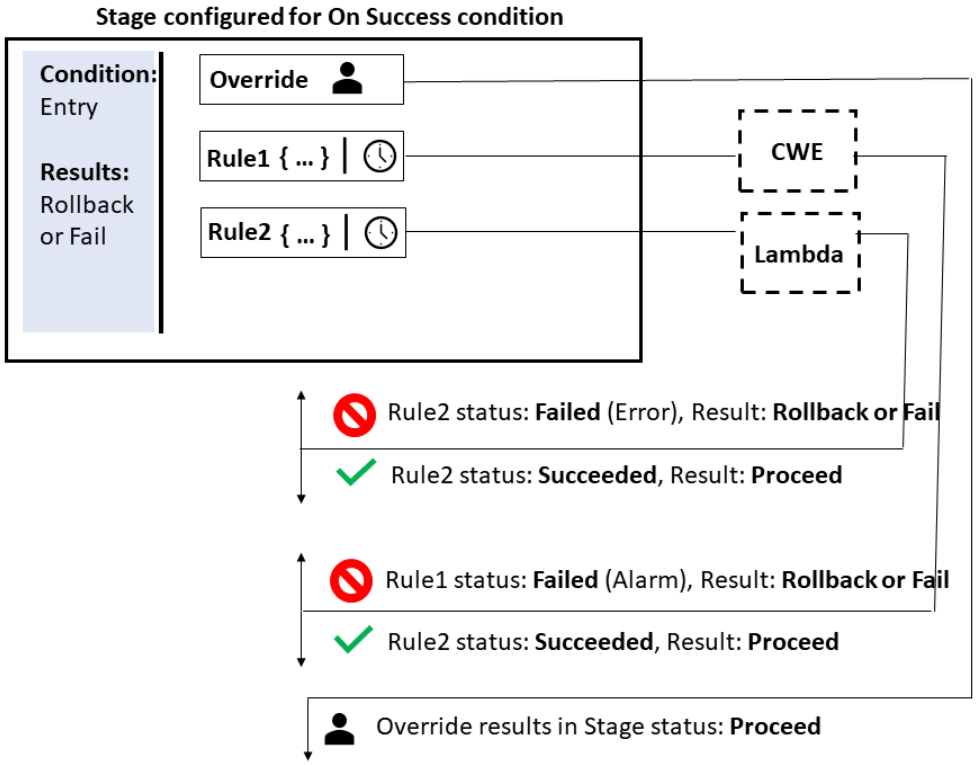



En el siguiente diagrama se muestra un ejemplo de flujo en el que la condición On Failure escribe in CodePipeline. Las condiciones responden a la pregunta: ¿Qué debería ocurrir si se cumple la condición, es decir, si todas las reglas se comprueban correctamente? En el siguiente flujo, la

condición On Failure se configura con una LambdaInvoke regla y una `CloudWatchAlarm` regla. Si la regla se lleva a cabo correctamente, se activaría el resultado configurado, como por ejemplo, “Error”.




En el siguiente diagrama se muestra un ejemplo de flujo para escribir la condición En caso de éxito CodePipeline. Las condiciones responden a la pregunta: ¿Qué debería ocurrir si se cumple la condición, es decir, si todas las reglas se comprueban correctamente? En el siguiente flujo, se configura una condición de éxito con una regla `LambdaInvoke` y una regla `CloudWatchAlarm`. Si la regla se lleva a cabo correctamente, se activaría el resultado configurado, como por ejemplo, “Error”.




## Reglas para condiciones de etapas

Al configurar condiciones de etapas, debe seleccionar una de las reglas predefinidas y especificar los resultados de dicha regla. El estado de una condición será Error si alguna de las reglas de la condición fallan y Correcto si todas las reglas se realizan correctamente. La forma en que se cumplen los criterios de las condiciones de fallo y de éxito depende del tipo de regla.

A continuación se detallan las reglas administradas que se pueden agregar a las condiciones de etapa.

- Las condiciones pueden usar la regla Comandos para especificar comandos que cumplan con los criterios de la regla para las mismas. Para obtener más información acerca de esta regla, consulte Comandos.
- Las condiciones pueden usar la AWS DeploymentWindow regla para especificar los tiempos de implementación aprobados para permitir una implementación. Los criterios de la regla se medirán con una expresión cron proporcionada para una ventana de implementación. La regla se ejecuta correctamente cuando la fecha y la hora de la ventana de implementación cumplen los criterios de la expresión cron de la regla. Para obtener más información acerca de esta regla, consulte DeploymentWindow.

- Las condiciones pueden usar la regla de AWS Lambda para comprobar los estados de error que devuelven las funciones de Lambda configuradas. La regla se cumple cuando la comprobación recibe el resultado de la función de Lambda. Un error de la función de Lambda cumple los criterios de las condiciones de fallo. Para obtener más información acerca de esta regla, consulte LambdaInvoke.
- Las condiciones pueden usar la AWS CloudWatchAlarm regla para comprobar si hay alarmas configuradas a partir de CloudWatch eventos. La regla se cumple cuando la comprobación devuelve un estado de alarma de OK, ALARM o INSUFFICIENT_DATA. En condiciones de éxito, OK y INSUFFICIENT_DATA cumplen los criterios. ALARM cumple los criterios para las condiciones de fallo. Para obtener más información acerca de esta regla, consulte CloudWatchAlarm.
- Las condiciones pueden usar la VariableCheck regla para crear una condición en la que la variable de salida se compare con una expresión proporcionada. La regla supera la comprobación cuando el valor de la variable cumple los criterios de la regla, por ejemplo, si el valor es igual o mayor que una variable de salida especificada. Para obtener más información acerca de esta regla, consulte VariableCheck.

# Tipos de canalización

CodePipeline proporciona los siguientes tipos de tuberías, que difieren en características y precio, para que pueda adaptar las características y el costo de las tuberías a las necesidades de sus aplicaciones.

- Las canalizaciones de tipo V1 tienen una estructura JSON que contiene parámetros estándares de canalización, etapa y nivel de acción.
- Las canalizaciones de tipo V2 tienen la misma estructura que las de tipo V1, además de parámetros adicionales para la seguridad de liberación y la configuración de los desencadenadores.

Para obtener información sobre los precios de CodePipeline, consulte Precios.

Consulte la página CodePipeline referencia de estructura de tuberías para obtener más información sobre los parámetros de cada tipo de canalización. Para obtener información sobre el tipo de canalización que debe elegir, consulte ¿Qué tipo de canalización es el adecuado para mí?.

# ¿Qué tipo de canalización es el adecuado para mí?

El tipo de canalización viene determinado por el conjunto de características y funciones compatibles con cada versión de canalización.

A continuación se presenta un resumen de los casos de uso y las características de que están disponibles para cada tipo de canalización.

| | Tipo V1 | Tipo V1 |
|---|---|---|
| Características | | |
| Casos de uso | • Implementaciones estándar | • Implementaciones con una configuración basada en el paso de variables a nivel de canalización en tiempo de ejecución<br>• Implementaciones en las que las canalizaciones están configuradas para iniciarse en etiquetas de Git |
| Variables de Action-level | Soportado | compatible |
| Modo de ejecución PARALELO | No compatible | compatible |
| Pipeline-level variables | No compatible | compatible |
| Modo de ejecución EN COLA | No compatible | compatible |
| Reversión de las etapas de la canalización | No compatible | compatible |
| Anulaciones de revisión de origen | No compatible | compatible |
| Condiciones de etapa | No compatible | compatible |

| | Tipo V1 | Tipo V1 |
|---|---|---|
| Características | | |
| Desencadena y filtra etiquetas de Git, solicitudes de extracción, ramificaciones o rutas de archivo | No compatible | compatible |
| La acción de Comandos | No compatible | compatible |
| Creación de condiciones de entrada con el resultado Omitir | No compatible | compatible |
| Configuración de una etapa para el reintento automático en caso de fallo | No compatible | compatible |

Para obtener información sobre los precios de CodePipeline, consulte Precios.

Puede crear y ejecutar un script de Python para obtener ayuda a la hora de analizar el posible costo de trasladar una canalización de tipo V1 a una canalización de tipo V2.

> **Note**
>
> El script de ejemplo que se muestra a continuación se proporciona únicamente con fines de demostración y evaluación. No se trata de una herramienta de cotización y no garantiza el costo del uso real de una canalización de tipo V2, ni incluye los impuestos que puedan aplicarse. Para obtener información sobre los precios de CodePipeline, consulte Precios.

Para crear y ejecutar un script que le ayude a evaluar el costo de trasladar una canalización de tipo V1 a una de tipo V2

1. Descargue e instale Python.
2. Abra una ventana de terminal. Ejecute el siguiente comando para crear una nueva secuencia de comandos de Python denominada PipelineCostAnalyzer.py.

```
vi PipelineCostAnalyzer.py
```

3. Copia y pega el siguiente código en el PipelineCostAnalyzer.py script.

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

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

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

    while hasNextToken:
        if nextToken=="":
            response =
 codepipeline.list_action_executions(pipelineName=pipeline_name)
        else:
            response =
 codepipeline.list_action_executions(pipelineName=pipeline_name,
 nextToken=nextToken)
        if 'nextToken' in response:
            nextToken = response['nextToken']
        else:
            hasNextToken= False
        for action_execution in response['actionExecutionDetails']:
            start_time = action_execution['startTime']
```

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

                total_blling_action_executions += 1
                action_execution_minutes = (end_time -
 start_time).total_seconds()/60
                action_execution_cost = math.ceil(action_execution_minutes) * 0.002
                total_action_execution_minutes += action_execution_minutes
                cost = round(cost + action_execution_cost, 2)

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

analyze_cost_in_v2(pipeline)
```

4. Desde la terminal o el símbolo del sistema, cambie los directorios en los que creó el script del analizador.

   Desde ese directorio, ejecuta el siguiente comando, donde  region  es el lugar Región de AWS donde creaste las canalizaciones de la V1 que deseas analizar. Si lo desea, también tiene la opción de evaluar una canalización específica; para ello, proporcione el nombre de dicha canalización:

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

Por ejemplo, ejecuta el siguiente comando para ejecutar la secuencia de comandos de Python denominada PipelineCostAnalyzer.py. En este ejemplo, la región es `us-west-2`.

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

> **Note**
>
> Este script analizará todas las canalizaciones de tipo V1 de la Región de AWS especificada, a menos que especifique el nombre de una canalización específica.

5. En el siguiente ejemplo de salida del script, podemos ver la lista de ejecuciones de acciones, la lista de ejecuciones de acciones aptas para su facturación, el tiempo de ejecución total de estas ejecuciones de acciones y el costo estimado de esas acciones tal como se realizaban en una canalización de tipo V2.

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

En este ejemplo, el valor negativo de la última fila representa la cantidad estimada que se podría ahorrar si se hiciera el traslado a canalizaciones de tipo V2.

> **Note**
>
> La salida del script y los ejemplos relacionados que muestran los costos y otra información son únicamente estimaciones. Están destinados únicamente a fines de demostración y evaluación, y no garantizan ningún ahorro real. Para obtener información sobre los precios de CodePipeline, consulte Precios.

# Empezar con CodePipeline

Si es la primera vez que lo usa CodePipeline, puede seguir los tutoriales de esta guía después de seguir los pasos de este capítulo para configurarlo.

La CodePipeline consola incluye información útil en un panel plegable que puede abrir desde el icono de información o desde cualquier enlace de información de la página.

(  ).

Puede cerrar este panel en cualquier momento.

La CodePipeline consola también te permite buscar rápidamente tus recursos, como repositorios, proyectos de compilación, aplicaciones de implementación y canalizaciones. Elija Ir a recurso o pulse la tecla / y, a continuación, escriba el nombre del recurso. Se muestran todas las coincidencias en la lista. En las búsquedas, no se distingue entre mayúsculas y minúsculas. Solo puede ver los recursos para los que tiene permiso. Para obtener más información, consulte Visualización de recursos en la consola.

Antes de poder utilizarlos AWS CodePipeline por primera vez, debe crear su propio usuario administrativo Cuenta de AWS y su primer usuario administrativo.

Temas

- Inscríbase en un Cuenta de AWS
- Aplique una política gestionada para el acceso administrativo a CodePipeline
- Instala la AWS CLI
- Abre la consola para CodePipeline
- Siguientes pasos

## Inscríbase en un Cuenta de AWS

Para empezar AWS, necesitas un Cuenta de AWS. Hay dos formas de inscribirse en AWS. Si desea tener un control total sobre la configuración de la cuenta y el lugar donde se almacena su contenido, o si tiene cargas de trabajo reguladas, como HIPAA o FedRAMP, utilice Sign up for (avanzado). AWS Si es un usuario nuevo y desea utilizar un inicio de sesión que ya tiene para iniciar sesión AWS y crear un AWS entorno con los valores predeterminados preconfigurados en Select, utilice Sign up for. Regiones de AWSAWS

# Aplique una política gestionada para el acceso administrativo a CodePipeline

Debe conceder permisos para interactuar con CodePipeline. El modo más rápido de hacerlo es aplicar la política administrada `AWSCodePipeline_FullAccess` al usuario administrativo.

> **Note**
>
> La `AWSCodePipeline_FullAccess` política incluye permisos que permiten al usuario de la consola transferir una función de IAM a CodePipeline otra Servicios de AWS. Esto permite al servicio asumir el rol y ejecutar acciones en su nombre. Al asociar la política a un usuario, rol o grupo, se aplican los permisos de `iam:PassRole`. Asegúrese de que la política solo se aplica a usuarios de confianza. Cuando los usuarios con estos permisos utilizan la consola para crear o editar una canalización, están disponibles las siguientes opciones:
>
> - Cree un rol CodePipeline de servicio o elija uno existente y transfiéralo a CodePipeline
> - Puede optar por crear una regla de CloudWatch eventos para la detección de cambios y pasar la función del servicio de CloudWatch CloudWatch eventos a Events
>
> Para obtener más información, consulte Otorgar permisos a un usuario para transferir un rol a un Servicio de AWS.

> **Note**
>
> La `AWSCodePipeline_FullAccess` política proporciona acceso a todas CodePipeline las acciones y recursos a los que tiene acceso el usuario de IAM, así como a todas las acciones posibles al crear etapas en una canalización, como la creación de etapas que incluyan CodeDeploy Elastic Beanstalk o Amazon S3. Como práctica recomendada, debe conceder a las personas únicamente los permisos necesarios para llevar a cabo sus tareas. Para obtener más información sobre cómo restringir a los usuarios de IAM a un conjunto limitado de CodePipeline acciones y recursos, consulte. Elimine los permisos de la función CodePipeline de servicio

Para dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en AWS IAM Identity Center:

  Cree un conjunto de permisos. Siga las instrucciones de Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center .

- Usuarios gestionados en IAM a través de un proveedor de identidades:

  Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.

- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o agregue un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

## Instala la AWS CLI

Para llamar a CodePipeline los comandos de la AWS CLI en una máquina de desarrollo local, debe instalar la AWS CLI. Este paso es opcional si desea empezar a utilizar únicamente los pasos de esta guía para la CodePipeline consola.

Para instalar y configurar el AWS CLI

1. En su máquina local, descargue e instale el AWS CLI. Esto le permitirá interactuar con él CodePipeline desde la línea de comandos. Para obtener más información, consulte Cómo configurar la interfaz de línea de AWS comandos.

   > **Note**
   >
   > CodePipeline solo funciona con AWS CLI las versiones 1.7.38 y posteriores. Para determinar qué versión de la AWS CLI que puede haber instalado, ejecute el comando. aws --version Para actualizar una versión anterior AWS CLI a la versión más reciente, siga las instrucciones de Desinstalar el y AWS CLI, a continuación, siga las instrucciones de Instalación del AWS Command Line Interface.

2. Configure AWS CLI con el configure comando, de la siguiente manera:

```
aws configure
```

Cuando se le solicite, especifique la clave de AWS acceso y la clave de acceso AWS secreta del usuario de IAM con CodePipeline el que vaya a utilizar. Cuando se le pida el nombre de la región por defecto, especifique la región en la que va a crear la canalización, por ejemplo. `us-east-2` Cuando se le pregunte el formato de salida predeterminado, indique `json`. Por ejemplo:

```
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**
>
> Para obtener más información sobre IAM, claves de acceso y claves secretas, consulte Gestión de claves de acceso para usuarios de IAM y ¿Cómo puedo obtener credenciales?
> Para obtener más información sobre las regiones y los puntos finales disponibles CodePipeline, consulte AWS CodePipeline puntos finales y cuotas.

## Abre la consola para CodePipeline

- Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

## Siguientes pasos

Ha completado los requisitos previos. Puedes empezar a usar CodePipeline. Para comenzar a trabajar con CodePipeline, consulte CodePipeline tutoriales.

# Integraciones de productos y servicios con CodePipeline

De forma predeterminada, AWS CodePipeline se integra con varios productos y servicios asociados. Servicios de AWS Utilice la información de las siguientes secciones como ayuda CodePipeline para configurar la integración con los productos y servicios que utiliza.

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con este servicio.

Temas


- Integraciones con tipos de CodePipeline acciones
- Integraciones generales con CodePipeline
- Ejemplos de la comunidad


## Integraciones con tipos de CodePipeline acciones

La información sobre las integraciones de este tema está organizada por tipo de CodePipeline acción.

Temas


- Integraciones de acciones de código fuente
- Integraciones de acciones de compilación
- Integraciones de acciones de prueba
- Integraciones de acciones de implementación
- Integración de la acción de aprobación con Amazon Simple Notification Service
- Integraciones de acciones de invocación


### Integraciones de acciones de código fuente

La siguiente información está organizada por tipo de CodePipeline acción y puede ayudarle CodePipeline a configurar la integración con los siguientes proveedores de acciones de origen.

Temas


- acciones de origen de Amazon ECR
- acciones de origen de Amazon S3

- Conexiones a Bitbucket Cloud GitHub (mediante una GitHub aplicación), a GitHub Enterprise Server y GitLab.com GitLab autogestionadas
- CodeCommit acciones de origen
- GitHub (a través de la aplicación OAuth), acciones de origen

## acciones de origen de Amazon ECR

Amazon ECR es un servicio de repositorio de imágenes de AWS Docker. Puede utilizar Docker para enviar y extraer comandos para cargar imágenes de Docker en su repositorio. En las definiciones de tareas de Amazon ECS se utilizan un URI de repositorio de Amazon ECR y una imagen para hacer referencia a la información de la imagen de origen.

Más información:

- Para ver los parámetros de configuración y un JSON/YAML fragmento de ejemplo, consulte Referencia de acciones de origen de Amazon ECR
- Creación de una canalización, etapas y acciones
- Tutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR

## acciones de origen de Amazon S3

Amazon S3 es un servicio de almacenamiento para Internet. Puede utilizar Amazon S3 para almacenar y recuperar cualquier cantidad de datos en cualquier momento y desde cualquier parte de la web. Puede configurarlo CodePipeline para usar un bucket de Amazon S3 versionado como acción fuente de su código.

> **Note**
>
> También es posible incluir Amazon S3 en una canalización como una acción de implementación.

Más información:

- Para ver los parámetros de configuración y un JSON/YAML fragmento de ejemplo, consulte Referencia sobre la acción de origen de Amazon S3

- Paso 1: creación de un bucket de origen de S3 para la aplicación
- Crear una canalización (CLI)
- CodePipeline usa Amazon EventBridge (anteriormente Amazon CloudWatch Events) para detectar cambios en el bucket de código fuente de Amazon S3. Consulte Integraciones generales con CodePipeline.

## Conexiones a Bitbucket Cloud GitHub (mediante una GitHub aplicación), a GitHub Enterprise Server y GitLab.com GitLab autogestionadas

Las conexiones (`CodeStarSourceConnection`acciones) se utilizan para acceder a Bitbucket Cloud GitHub, GitHub Enterprise Server o GitLab repositorio autogestionado GitLab.com de terceros.

> **Note**
>
> Esta función no está disponible en las regiones Asia Pacífico (Hong Kong), Asia Pacífico (Hyderabad), Asia Pacífico (Yakarta), Asia Pacífico (Melbourne), Asia Pacífico (Osaka), África (Ciudad del Cabo), Oriente Medio (Baréin), Oriente Medio (Emiratos Árabes Unidos), Europa (España), Europa (Zúrich), Israel (Tel Aviv) o AWS GovCloud (). US-West Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline. Para ver consideraciones sobre esta acción en la región de Europa (Milán), consulte la nota que aparece en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

| | |
|---|---|
| Bitbucket Cloud | Puedes configurarlo CodePipeline para usar un repositorio de Bitbucket Cloud como fuente de tu código. Previamente debe haber creado una cuenta de Bitbucket y por lo menos un repositorio de Bitbucket Cloud. Puede añadir una acción de origen a su repositorio de Bitbucket Cloud creando una canalización o editando una existente.<br><br>**Note**<br>Puede crear conexiones a un repositorio de Bitbucket Cloud. Los tipos de proveedores de Bitbucket instalados, como Bitbucket Server, no son compatibles. |

| | |
|---|---|
| | Puede configurar recursos denominados conexiones para permitir que las canalizaciones obtengan acceso a repositorios de código de terceros. Al crear una conexión, se instala la aplicación Connector con el repositorio de código de terceros y, a continuación, se asocia a la conexión.<br><br>Para Bitbucket Cloud, use la opción Bitbucket en la consola o la acción `CodestarSourceConnection` en la CLI. Consulte Conexiones de Bitbucket Cloud.<br><br>Puede usar la opción Clonación completa para que esta acción haga referenci a a los metadatos de Git del repositorio, de modo que las acciones posterior es puedan ejecutar comandos de Git directamente. Esta opción solo la pueden utilizar las acciones CodeBuild posteriores.<br><br>Más información:<br><br>• Para ver los parámetros de configuración y un JSON/YAML fragmento de ejemplo, consulte. CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogesti onadas<br>• Para ver un tutorial de introducción que crea una canalización con una fuente de Bitbucket, consulte Introducción a las conexiones. |
| GitHub o Enterprise Cloud GitHub | Puede configurarlo CodePipeline para usar un GitHub repositorio como fuente de su código. Debes haber creado previamente una GitHub cuenta y al menos un GitHub repositorio. Puedes añadir una acción de origen a tu GitHub repositorio creando una canalización o editando una existente.<br><br>Puede configurar recursos denominados conexiones para permitir que las canalizaciones obtengan acceso a repositorios de código de terceros. Al crear una conexión, se instala la aplicación Connector con el repositorio de código de terceros y, a continuación, se asocia a la conexión.<br><br>Utilice la opción de proveedor GitHub (a través de la GitHub aplicación) en la consola o la `CodestarSourceConnection` acción en la CLI. Consulte GitHub conexiones. |

| | |
|---|---|
| | Puede usar la opción Clonación completa para que esta acción haga referenci a a los metadatos de Git del repositorio, de modo que las acciones posterior es puedan ejecutar comandos de Git directamente. Esta opción solo la pueden utilizar las acciones CodeBuild posteriores.<br><br>Más información:<br><br>• Para ver los parámetros de configuración y un JSON/YAML fragmento de ejemplo, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas<br>• Para ver un tutorial que muestra cómo conectarse a un GitHub repositorio y utilizar la opción de clonación completa, consulte. Tutorial: Utilice un clon completo con una fuente de GitHub canalización<br>• La acción actual GitHub (a través de la GitHub aplicación) es la acción fuente de la versión 2 GitHub. La acción GitHub (a través de la aplicación OAuth) es la acción de la versión 1 GitHub que se gestiona con la autentica ción mediante token de OAuth. Si bien no recomendamos usar la acción GitHub (mediante la aplicación OAuth), las canalizaciones existentes con la acción GitHub (mediante la aplicación OAuth) seguirán funcionando sin ningún impacto. Ahora puedes usar una acción de CodeStarSourceConn ection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas origen en tu proceso que gestione tu GitHub acción de origen con las aplicaciones. GitHub Si tienes una canalización que usa la acción GitHub (a través de la aplicación OAuth), consulta los pasos para actualizarla y usar una acción GitHub (a través de la GitHub aplicación) en ella. Actualizar una acción fuente GitHub (a través de la aplicación OAuth) a una acción fuente GitHub (a través de GitHub la aplicación) |
| GitHub Servidor empresarial | Puede configurarlo CodePipeline para usar un repositorio de GitHub Enterpris e Server como fuente de su código. Debe haber creado previamente una GitHub cuenta y al menos un GitHub repositorio. Puede añadir una acción de origen para su repositorio de GitHub Enterprise Server creando una canalizac ión o editando una existente. |

| | |
|---|---|
| | Puede configurar recursos denominados conexiones para permitir que las canalizaciones obtengan acceso a repositorios de código de terceros. Al crear una conexión, se instala la aplicación Connector con el repositorio de código de terceros y, a continuación, se asocia a la conexión.<br><br>Utilice la opción GitHub Enterprise Server provider en la consola o la `CodestarSourceConnection` acción en la CLI. Consulte GitHub Conexiones de Enterprise Server.<br><br>Puede usar la opción Clonación completa para que esta acción haga referenci a a los metadatos de Git del repositorio, de modo que las acciones posterior es puedan ejecutar comandos de Git directamente. Esta opción solo la pueden utilizar las acciones CodeBuild posteriores.<br><br>Más información:<br><br>• Para ver los parámetros de configuración y un JSON/YAML fragmento de ejemplo, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas<br>• Para ver un tutorial que muestra cómo conectarse a un GitHub repositorio y utilizar la opción de clonación completa, consulte. Tutorial: Utilice un clon completo con una fuente de GitHub canalización |
| GitLab.com | Puede configurarlo CodePipeline para usar un GitLab.com repositorio como fuente de su código. Debes haber creado previamente una GitLab.com cuenta y al menos un GitLab.com repositorio. Puedes añadir una acción de origen a tu GitLab.com repositorio creando una canalización o editando una existente.<br><br>Utilice la opción GitLabde proveedor en la consola o la `CodestarS ourceConnection` acción con el `GitLab` proveedor en la CLI. Consulte GitLab.com conexiones.<br><br>Más información:<br><br>• Para ver los parámetros de configuración y un JSON/YAML fragmento de ejemplo, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas |

| | |
|---|---|
| GitLab autogestionado | Puede configurarlo CodePipeline para utilizar una instalación GitLab autogestionada como fuente del código. Debe haber creado previamente una GitLab cuenta y tener una suscripción autogestionada GitLab (Enterprise Edition o Community Edition). Puedes añadir una acción de origen a tu repositorio GitLab autogestionado creando una canalización o editando una existente.<br><br>Puede configurar recursos denominados conexiones para permitir que las canalizaciones obtengan acceso a repositorios de código de terceros. Al crear una conexión, se instala la aplicación Connector con el repositorio de código de terceros y, a continuación, se asocia a la conexión.<br><br>Utilice la opción de proveedor GitLab autogestionado en la consola o la `CodestarSourceConnection` acción en la CLI. Consulte Conexiones para GitLab autogestión.<br><br>Puede usar la opción Clonación completa para que esta acción haga referencia a los metadatos de Git del repositorio, de modo que las acciones posteriores puedan ejecutar comandos de Git directamente. Esta opción solo la pueden utilizar las acciones CodeBuild posteriores.<br><br>Más información:<br><br>• Para ver los parámetros de configuración y un JSON/YAML fragmento de ejemplo, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas<br>• Para conocer los pasos para crear una conexión con este tipo de proveedor, consulte Conexiones para GitLab autogestión. |

## CodeCommit acciones de origen

CodeCommit es un servicio de control de versiones que puede utilizar para almacenar y administrar recursos de forma privada (como documentos, código fuente y archivos binarios) en la nube. Puedes configurarlo CodePipeline para usar una rama en un CodeCommit repositorio como fuente de tu código. Cree el repositorio y asócielo a un directorio de trabajo de su equipo local. A continuación, puede crear una canalización que utilice la bifurcación como parte de una acción de origen en una etapa. Puedes conectarte al CodeCommit repositorio creando una canalización o editando una existente.

Puede usar la opción Clonación completa para que esta acción haga referencia a los metadatos de Git del repositorio, de modo que las acciones posteriores puedan ejecutar comandos de Git directamente. Esta opción solo la pueden utilizar las acciones CodeBuild posteriores.

Más información:

- Para ver los parámetros de configuración y un JSON/YAML fragmento de ejemplo, consulte. CodeCommit referencia de acción de origen
- Tutorial: Crear una canalización simple (CodeCommit repositorio)
- CodePipeline usa Amazon CloudWatch Events para detectar cambios en los CodeCommit repositorios que se utilizan como fuente para una canalización. Cada acción de código fuente tiene una regla de evento correspondiente. Esta regla de evento inicia la canalización cuando se produce un cambio en el repositorio. Consulte Integraciones generales con CodePipeline.

## GitHub (a través de la aplicación OAuth), acciones de origen

La acción GitHub (a través de la aplicación OAuth) es la acción de la versión 1 que se gestiona con OAuth GitHub Apps. En las regiones disponibles, también puedes usar una acción de CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas origen en tu proceso que gestione tu acción de GitHub origen con Apps. GitHub Si tienes una canalización que usa la acción GitHub (a través de la aplicación OAuth), consulta los pasos para actualizarla y usar una acción GitHub (a través de la GitHub aplicación) en ella. Actualizar una acción fuente GitHub (a través de la aplicación OAuth) a una acción fuente GitHub (a través de GitHub la aplicación)

> **Note**
>
> Si bien no recomendamos usar la acción GitHub (a través de la aplicación OAuth), las canalizaciones existentes con la acción GitHub (a través de la aplicación OAuth) seguirán funcionando sin ningún impacto.

Más información:

- Para obtener más información sobre el acceso OAuth-based GitHub (a través de la aplicación OAuth) en comparación con el acceso basado en aplicaciones, consulta. GitHub https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps

- Para ver un apéndice que contiene los detalles de la acción GitHub (a través de la aplicación OAuth), consulte. Apéndice A: acciones de GitHub origen (a través de la aplicación OAuth)

## Integraciones de acciones de compilación

La siguiente información está organizada por tipo de CodePipeline acción y puede ayudarte a configurarla CodePipeline para integrarla con los siguientes proveedores de acciones de compilación.

Temas

- CodeBuild acciones de construcción
- CloudBees acciones de construcción
- Acciones de compilación y publicación de Amazon ECR
- Acciones de compilación Jenkins
- TeamCity acciones de construcción

### CodeBuild acciones de construcción

CodeBuild es un servicio de compilación completamente administrado que compila código fuente, ejecuta pruebas unitarias y produce artefactos listos para su implementación.

Puedes añadirla CodeBuild como acción de construcción a la etapa de creación de una canalización. Para obtener más información, consulte la referencia de configuración de CodePipeline acciones paraAWS CodeBuild referencia de acciones de creación y prueba.

> **Note**
>
> CodeBuild también se puede incluir en una canalización como acción de prueba, con o sin un resultado de compilación.

Más información:

- Para ver los parámetros de configuración y un JSON/YAML fragmento de ejemplo, consulte. AWS CodeBuild referencia de acciones de creación y prueba
- ¿Qué es? CodeBuild
- CodeBuild— Servicio de construcción totalmente gestionado

## CloudBees acciones de construcción

Puedes configurarlo CodePipeline para usarlo CloudBeespara compilar o probar tu código en una o más acciones de una canalización.

Más información:

- re:Invent 2017: Cloud First con AWS

## Acciones de compilación y publicación de Amazon ECR

Amazon ECR es un servicio de repositorio de imágenes de AWS Docker. Puede utilizar Docker para enviar y extraer comandos para cargar imágenes de Docker en su repositorio.

Puede añadir la acción `ECRBuildAndPublish` a su canalización para automatizar la compilación e inserción de una imagen. Para obtener más información, consulte la referencia de configuración de CodePipeline Action para. Referencia de la acción de compilación `ECRBuildAndPublish`

## Acciones de compilación Jenkins

Puedes configurarlo CodePipeline para usar el CI de Jenkins para compilar o probar tu código en una o más acciones de una canalización. Debes haber creado previamente un proyecto de Jenkins e instalado y configurado el CodePipeline complemento de Jenkins para ese proyecto. La conexión al proyecto Jenkins es posible mediante la creación de una nueva canalización o la modificación de una existente.

El acceso para Jenkins se configura según cada proyecto. Debe instalar el CodePipeline complemento para Jenkins en todas las instancias de Jenkins con las que desee utilizarlas. CodePipeline También debes configurar el CodePipeline acceso al proyecto de Jenkins. Proteja su proyecto de Jenkins configurándolo para que solo acepte HTTPS/SSL conexiones. Si su proyecto de Jenkins está instalado en una instancia de Amazon EC2, considere la posibilidad de proporcionar AWS sus credenciales AWS CLI instalándolas en cada instancia. A continuación, configure un AWS perfil en esas instancias con las credenciales que desee usar para las conexiones. Esta es una alternativa a añadirlas y almacenarlas a través de la interfaz web de Jenkins.

Más información:

- Accessing Jenkins
- Tutorial: Crear una canalización de cuatro etapas

### TeamCity acciones de construcción

Puedes configurarlo CodePipeline para usarlo TeamCitypara compilar y probar tu código en una o más acciones de una canalización.

Más información:

- TeamCity Complemento para CodePipeline

## Integraciones de acciones de prueba

La siguiente información está organizada por tipo de CodePipeline acción y puede ayudarle CodePipeline a configurar la integración con los siguientes proveedores de acciones de prueba.

Temas

- CodeBuild acciones de prueba
- AWS Device Farm acciones de prueba
- Acciones de prueba de Ghost Inspector
- OpenText LoadRunner Acciones de prueba en la nube
- Automatización de pruebas mediante Reflect

### CodeBuild acciones de prueba

CodeBuildes un servicio de construcción en la nube totalmente gestionado. CodeBuild compila el código fuente, ejecuta pruebas unitarias y produce artefactos listos para su despliegue.

Puedes añadirlos CodeBuild a una canalización como acción de prueba. Para obtener más información, consulte la Referencia de configuración de acciones de CodePipeline de AWS CodeBuild referencia de acciones de creación y prueba.

> **Note**
>
> CodeBuild también se puede incluir en una canalización como una acción de construcción, con un artefacto de salida de compilación obligatorio.

Más información:

- Para ver los parámetros de configuración y un JSON/YAML fragmento de ejemplo, consulte. AWS CodeBuild referencia de acciones de creación y prueba
- ¿Qué es? CodeBuild

## AWS Device Farm acciones de prueba

AWS Device Farm es un servicio de pruebas de aplicaciones que le permite probar sus aplicaciones Android, iOS y web e interactuar con ellas en teléfonos y tablets físicos reales. Puedes configurarlo CodePipeline AWS Device Farm para probarlo en una o más acciones de una canalización. AWS Device Farm te permite cargar tus propias pruebas o utilizar pruebas de compatibilidad integradas y sin scripts. Dado que las pruebas se realizan de forma automática en paralelo, en pocos minutos comienzan pruebas en varios dispositivos. Un informe de prueba que contiene resultados de alto nivel, registros de bajo nivel, capturas de pantalla píxel a píxel y datos de rendimiento se actualiza a medida que se completan las pruebas. AWS Device Farm permite probar aplicaciones nativas e híbridas de Android, iOS y Fire OS, incluidas las creadas con Titanium PhoneGap, Xamarin, Unity y otros marcos. Admite el acceso remoto de aplicaciones Android, lo que le permite interactuar directamente con los dispositivos de prueba.

Más información:

- Para ver los parámetros de configuración y un JSON/YAML fragmento de ejemplo, consulte. AWS Device Farm referencia de acción de prueba
- ¿Qué es? AWS Device Farm
- Utilización AWS Device Farm en una fase CodePipeline de prueba

## Acciones de prueba de Ghost Inspector

Puedes configurarlo CodePipeline para usar Ghost Inspector para probar tu código en una o más acciones de una canalización.

Más información:

- Documentación de Ghost Inspector para la integración del servicio con CodePipeline

### OpenText LoadRunner Acciones de prueba en la nube

Puedes configurarlo CodePipeline para usar OpenText LoadRunner Cloud en una o más acciones de una canalización.

Más información:

- LoadRunner Documentación en la nube para la integración con CodePipeline

### Automatización de pruebas mediante Reflect

Reflect es la solución de automatización de AI-powered pruebas que le permite simplificar las pruebas y superar los desafíos de los procesos manuales. Con la automatización de pruebas sin código, Reflect agiliza la creación, la ejecución y el mantenimiento de las pruebas, lo que permite crear pruebas sólidas y repetibles sin necesidad de conocimientos técnicos. Al eliminar la complejidad y garantizar una interrupción mínima de los flujos de trabajo, lo ayuda a agilizar las pruebas y a ofrecer aplicaciones de alta calidad con confianza en todo momento.

Más información:

- AWS CodePipeline pruebe la integración con Reflect

## Integraciones de acciones de implementación

La siguiente información está organizada por tipo de CodePipeline acción y puede ayudarle CodePipeline a configurar la integración con los siguientes proveedores de acciones de despliegue.

Temas

- Acciones de implementación de Amazon EC2
- Acciones de implementación de Amazon Elastic Kubernetes Service (EKS)
- Acción de implementación de Amazon S3
- AWS AppConfig implementar acciones
- CloudFormation implementar acciones
- CloudFormation StackSets implementar acciones
- Acciones de implementación de Amazon ECS
- Acciones de implementación de Elastic Beanstalk

- OpsWorks implementar acciones
- Acciones de implementación de Service Catalog
- Acciones de implementación de Amazon Alexa
- CodeDeploy implementar acciones
- XebiaLabs implementar acciones

### Acciones de implementación de Amazon EC2

Amazon EC2 permite crear y administrar la computación en la nube. Puede añadir una acción a una canalización que use Amazon EC2 como proveedor de implementación que implemente su aplicación en sus instancias.

Más información:

- Consulte la página de referencia de la acción en Referencia de la acción de Amazon EC2.
- Para ver un tutorial, consulte Tutorial: Implemente en instancias de Amazon EC2 con CodePipeline.

### Acciones de implementación de Amazon Elastic Kubernetes Service (EKS)

Amazon EKS le permite crear y administrar clústeres de Kubernetes. Puede añadir una acción a una canalización que use Amazon EKS como proveedor de implementación que implemente su imagen en su clúster. Puede utilizar plantillas de Helm o archivos de manifiesto de Kubernetes.

Más información:

- Consulte la página de referencia de la acción en Referencia de acción de implementación de Amazon Elastic Kubernetes Service EKS.
- Para ver un tutorial, consulte Tutorial: Implemente en Amazon EKS con CodePipeline.

### Acción de implementación de Amazon S3

Amazon S3 es un servicio de almacenamiento para Internet. Puede utilizar Amazon S3 para almacenar y recuperar cualquier cantidad de datos en cualquier momento y desde cualquier parte

de la web. Puede añadir una acción a una canalización que use Amazon S3 como proveedor de implementación.

> **Note**
>
> También es posible incluir Amazon S3 en una canalización como acción de origen.

Más información:

- Creación de una canalización, etapas y acciones
- Tutorial: Crear una canalización que utilice Amazon S3 como proveedor de implementación

## AWS AppConfig implementar acciones

AWS AppConfig es la capacidad de AWS Systems Manager crear, administrar e implementar rápidamente configuraciones de aplicaciones. Puede usarlo AppConfig con aplicaciones alojadas en instancias EC2 AWS Lambda, contenedores, aplicaciones móviles o dispositivos de IoT.

Más información:

- CodePipeline Referencia de configuración de acciones para Referencia de la acción de implementación de AWS AppConfig
- Tutorial: Crear una canalización que se utilice AWS AppConfig como proveedor de despliegue

## CloudFormation implementar acciones

CloudFormationofrece a los desarrolladores y administradores de sistemas una forma sencilla de crear y gestionar un conjunto de AWS recursos relacionados mediante plantillas para aprovisionar y actualizar dichos recursos. Puede utilizar las plantillas de ejemplo del servicio o crear las suyas propias. Las plantillas describen los AWS recursos y cualquier dependencia o parámetro de tiempo de ejecución necesarios para ejecutar la aplicación.

El modelo de aplicaciones AWS sin servidor (AWS SAM) se amplía CloudFormation para proporcionar una forma simplificada de definir e implementar aplicaciones sin servidor. AWS SAM admite las API de Amazon API Gateway, las funciones de AWS Lambda y las tablas de Amazon DynamoDB. Puede usarlo CodePipeline con CloudFormation y el AWS SAM para entregar sus aplicaciones sin servidor de forma continua.

Puede añadir una acción a una canalización que se utilice AWS CloudFormation como proveedor de implementación. Si lo utilizas AWS CloudFormation como proveedor de despliegues, puedes tomar medidas sobre las CloudFormation pilas y los conjuntos de cambios como parte de la ejecución de una canalización. CloudFormation puede crear, actualizar, reemplazar y eliminar pilas y conjuntos de cambios cuando se ejecuta una canalización. Como resultado, AWS los recursos personalizados se pueden crear, aprovisionar, actualizar o finalizar durante la ejecución de una canalización de acuerdo con las especificaciones que proporciones en las CloudFormation plantillas y las definiciones de parámetros.

Más información:

- CodePipeline Referencia de configuración de acciones para Referencia de la acción de implementación de CloudFormation
- Entrega continua con CodePipeline: aprenda a utilizarla CodePipeline para crear un flujo de trabajo de entrega continua para CloudFormation.
- Automatización del despliegue de Lambda-based aplicaciones: aprenda a utilizar el modelo de aplicaciones AWS sin servidor y CloudFormation a crear un flujo de trabajo de entrega continua para su Lambda-based aplicación.

## CloudFormation StackSets implementar acciones

CloudFormationle ofrece una forma de implementar recursos en varias cuentas y AWS regiones.

Puede utilizarla CodePipeline CloudFormation para actualizar la definición del conjunto de pilas e implementar actualizaciones en sus instancias.

Puedes añadir las siguientes acciones a una canalización para utilizarlas AWS CloudFormation StackSets como proveedor de despliegues.

- CloudFormationStackSet
- CloudFormationStackInstances

Más información:

- CodePipeline Referencia de configuración de acciones para CloudFormation Referencia de la acción de implementación de StackSets
- Tutorial: Crea una canalización con AWS CloudFormation StackSets acciones de despliegue

## Acciones de implementación de Amazon ECS

Amazon ECS es un servicio de administración de contenedores de alto rendimiento y elevada escalabilidad que le permite ejecutar aplicaciones basadas en contenedores en la Nube de AWS. Cuando cree una canalización, puede seleccionar Amazon ECS como proveedor de implementación. Un cambio en el código del repositorio de control de origen activa la canalización y esta crea una nueva imagen de Docker, la envía al registro de contenedores e implementa la imagen actualizada en Amazon ECS. También puede utilizar la acción del proveedor de ECS (Blue/Green) CodePipeline para enrutar e implementar el tráfico a Amazon ECS con CodeDeploy.

Más información:

- ¿Qué es Amazon ECS?
- Tutorial: Implementación continua con CodePipeline
- Creación de una canalización, etapas y acciones
- Tutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR

## Acciones de implementación de Elastic Beanstalk

Elastic Beanstalk es un servicio para implementar y escalar aplicaciones y servicios web desarrollados con Java, .NET, PHP, Python, Ruby Node.js, Go y Docker en servidores conocidos como Apache, Nginx, Passenger e IIS. Puede configurar el uso CodePipeline de Elastic Beanstalk para implementar el código. Puede crear el entorno y la aplicación de Elastic Beanstalk que se usarán en una acción de implementación de una etapa, ya sea antes de crear la canalización o al usar el asistente Crear canalización.

> **Note**
>
> Esta característica no está disponible en las regiones de Asia-Pacífico (Hyderabad), Asia-Pacífico (Melbourne), Medio Oriente (EAU), Europa (España) o Europa (Zúrich). Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline.

Más información:

- Introducción a Elastic Beanstalk

- Creación de una canalización, etapas y acciones

## OpsWorks implementar acciones

OpsWorks es un servicio de gestión de la configuración que le ayuda a configurar y operar aplicaciones de todas las formas y tamaños con Chef. Con AWS OpsWorks Stacksél, puede definir la arquitectura de la aplicación y las especificaciones de cada componente, incluida la instalación del paquete, la configuración del software y los recursos, como el almacenamiento. Puede configurarlo CodePipeline AWS OpsWorks Stacks para implementar su código junto con libros de cocina y aplicaciones personalizados de Chef. OpsWorks

- Libros de cocina personalizados para Chef: OpsWorks utiliza los libros de cocina de Chef para realizar tareas como la instalación y configuración de paquetes y el despliegue de aplicaciones.
- Aplicaciones: una OpsWorks aplicación consiste en un código que se desea ejecutar en un servidor de aplicaciones. El código de aplicación se almacena en un repositorio, como un bucket de Amazon S3.

Antes de crear la canalización, debe crear la OpsWorks pila y la capa. Puede crear la OpsWorks aplicación para utilizarla en una acción de despliegue en una etapa antes de crear la canalización o cuando utilice el asistente Crear canalización.

CodePipeline Actualmente, el soporte para solo OpsWorks está disponible en la región EE. UU. Este (Virginia del Norte) (us-east-1).

Más información:

- Utilizándolo con CodePipeline AWS OpsWorks Stacks
- Cookbooks and Recipes
- AWS OpsWorks Apps (Aplicaciones)

## Acciones de implementación de Service Catalog

Service Catalog permite a las organizaciones crear y administrar catálogos de productos aprobados para su uso. AWS

Puede configurarlo CodePipeline para implementar actualizaciones y versiones de las plantillas de sus productos en Service Catalog. Puede crear el producto de Service Catalog que se utilizará en

una acción de implementación y, a continuación, utilizar el asistente Crear canalización para crear la canalización.

Más información:

- Tutorial: Crear una canalización que se implemente en Service Catalog
- Creación de una canalización, etapas y acciones

## Acciones de implementación de Amazon Alexa

El kit de habilidades de Amazon Alexa le permite crear y distribuir habilidades basadas en la nube a los usuarios de los Alexa-enabled dispositivos.

> **Note**
>
> Esta característica no está disponible en las regiones de Asia Pacífico (Hong Kong) ni Europa (Milán). Para utilizar otras acciones de implementación disponibles en esa región, consulte Integraciones de acciones de implementación.

Puede añadir una acción a una canalización que utilice Alexa Skills Kit como proveedor de implementación. La canalización detecta los cambios en el código fuente y, a continuación, implementa las actualizaciones de la habilidad de Alexa en el servicio Alexa.

Más información:

- Tutorial: Crear una canalización que implemente una habilidad de Amazon Alexa

## CodeDeploy implementar acciones

CodeDeploycoordina las implementaciones de aplicaciones en EC2/on-premises instancias de Amazon, plataformas informáticas de Amazon Elastic Container Service y plataformas AWS Lambda informáticas sin servidor. Puede configurarlo CodePipeline para usarlo para CodeDeploy implementar su código. Puede crear el grupo de implementaciones, la implementación y la aplicación de CodeDeploy que se usarán en una acción de implementación de una etapa, ya sea antes de crear la canalización o al usar el asistente Create Pipeline (Crear canalización).

Más información:

- Paso 3: Crea una aplicación en CodeDeploy
- Tutorial: Crear una canalización simple (CodeCommit repositorio)

### XebiaLabs implementar acciones

Puedes configurarlo CodePipeline para usarlo XebiaLabspara implementar tu código en una o más acciones de una canalización.

Más información:

- Uso de XL Deploy con CodePipeline

## Integración de la acción de aprobación con Amazon Simple Notification Service

Amazon SNS es un servicio de notificaciones push rápido, flexible y totalmente administrado que le permite enviar mensajes individuales o distribuir mensajes a gran cantidad de destinatarios. Amazon SNS hace que enviar notificaciones push a usuarios de dispositivos móviles o destinatarios de correo electrónico, o incluso enviar mensajes a otros servicios distribuidos, resulte sencillo y rentable.

Al crear una solicitud de aprobación manual en CodePipeline, si lo desea, puede publicarla en un tema de Amazon SNS para que todos los usuarios de IAM suscritos a ella reciban una notificación de que la acción de aprobación está lista para ser revisada.

Más información:

- ¿Qué es Amazon SNS?
- Otorgue permisos de Amazon SNS a un rol de servicio CodePipeline

## Integraciones de acciones de invocación

La siguiente información está organizada por tipo de CodePipeline acción y puede ayudarle CodePipeline a configurar la integración con los siguientes proveedores de invocación de acciones.

Temas

- Acciones de invocación de Amazon Inspector

- Acciones de invocación de Lambda
- Acciones de invocación de Step Functions

## Acciones de invocación de Amazon Inspector

Amazon Inspector es un servicio de administración de vulnerabilidades que detecta automáticamente las cargas de trabajo y las analiza de forma continua en busca de vulnerabilidades de software y exposiciones de red no deseadas. Amazon Inspector admite varios formatos de archivo, incluidos TAR y WAR, así como archivos binarios, como los de Rust y Go.

Puede configurar la CodePipeline `InspectorScan` acción para automatizar el escaneo del código fuente o del repositorio de imágenes de Amazon ECR en busca de vulnerabilidades.

Más información:

- CodePipeline referencia de configuración de acciones para Referencia de la acción de invocación de `InspectorScan` de Amazon Inspector

## Acciones de invocación de Lambda

Lambda le permite ejecutar código sin aprovisionar ni administrar servidores. Puede configurarlo CodePipeline para utilizar las funciones de Lambda a fin de añadir flexibilidad y funcionalidad a sus canalizaciones. Puede crear la función de Lambda para añadirla como una acción en una etapa, ya sea antes de crear la canalización o al usar el asistente Create Pipeline (Crear canalización).

Más información:

- CodePipeline Referencia de configuración de acciones para AWS Lambda invocar referencia a la acción
- Invoca un AWS Lambda función en una canalización en CodePipeline

## Acciones de invocación de Step Functions

Step Functions le permite crear y configurar máquinas de estado. Puede configurar CodePipeline el uso de acciones de invocación de Step Functions para activar ejecuciones de máquinas de estado.

Más información:

- CodePipeline Referencia de configuración de acciones para AWS Step Functions invocar referencia de acción
- Tutorial: Usa una acción de AWS Step Functions invocación en una canalización

## Integraciones generales con CodePipeline

Las siguientes Servicio de AWS integraciones no se basan en tipos de CodePipeline acciones.

| | |
|---|---|
| Amazon CloudWatch | Amazon CloudWatch supervisa tus AWS recursos.<br><br>Más información:<br><br>• ¿Qué es Amazon CloudWatch? |
| Amazon EventBridge | Amazon EventBridge es un servicio web que detecta tus cambios en Servicios de AWS función de las reglas que tú definas e invoca una acción en una o varias especificadas Servicios de AWS cuando se produce un cambio.<br><br>• Inicia la ejecución de una canalización automáticamente cuando algo cambia: puedes CodePipeline configurarlo como un objetivo en las reglas configuradas en Amazon EventBridge. Esto configura las canalizaciones para que se inicien automáticamente cuando otro servicio sufra algún cambio.<br><br>Más información:<br><br>• ¿Qué es Amazon EventBridge?<br>• Inicie una canalización en CodePipeline.<br>• CodeCommit acciones de origen y EventBridge<br>• Reciba notificaciones cuando cambie el estado de una canalización: puede configurar EventBridge reglas para detectar y reaccionar ante los cambios en el estado de ejecución de una canalización, etapa o acción.<br><br>Más información:<br><br>• Supervisión de CodePipeline eventos |

| | |
|---|---|
| | • Tutorial: configurar una regla de CloudWatch eventos para recibir notificaciones por correo electrónico sobre los cambios en el estado de los oleoductos |
| AWS Cloud9 | AWS Cloud9 es un IDE en línea al que puede acceder a través de su navegador web. El IDE ofrece una completa experiencia de edición de código, con soporte para varios lenguajes de programación y depurador es de tiempo de ejecución, así como un terminal integrado. En segundo plano, una instancia de Amazon EC2 aloja un entorno de AWS Cloud9 desarrollo. Para obtener más información, consulte la Guía del usuario de AWS Cloud9.<br><br>Más información:<br><br>• Configuración de AWS Cloud9 |
| AWS CloudTrail | CloudTrailcaptura las llamadas a la AWS API y los eventos relaciona dos realizados por una AWS cuenta o en su nombre y entrega los archivos de registro a un bucket de Amazon S3 que usted especifiq ue. Puede configurarlo CloudTrail para capturar las llamadas a la API desde la CodePipeline consola y los CodePipeline comandos desde la AWS CLI CodePipeline API y desde ella.<br><br>Más información:<br><br>• Registro de llamadas a la CodePipeline API con AWS CloudTrail |
| AWS CodeStar Notificaciones | Puede configurar notificaciones para que los usuarios conozcan los cambios importantes, como cuando se inicia la ejecución de una canalización. Para obtener más información, consulte Creación de una regla de notificación. |

| | |
|---|---|
| AWS Key Management Service | AWS KMS es un servicio administrado que le permite crear y controlar fácilmente las claves de cifrado que se utilizan para cifrar datos. De forma predeterminada, se CodePipeline utiliza AWS KMS para cifrar los artefactos de las canalizaciones almacenadas en los buckets de Amazon S3.<br><br>Más información:<br><br>• Para crear una canalización que utilice un bucket de origen, un bucket de artefactos y un rol de servicio de una AWS cuenta y CodeDeploy recursos de una AWS cuenta diferente, debe crear una clave de KMS administrada por el cliente, añadir la clave a la canalización y configurar las políticas y los roles de la cuenta para permitir el acceso entre cuentas. Para obtener más información, consulte Cree una canalización en la CodePipeline que se utilicen los recursos de otra AWS inscrita.<br>• Para crear una canalización a partir de una AWS cuenta que despliega una CloudFormation pila en otra AWS cuenta, debes crear una clave de KMS administrada por el cliente, añadir la clave a la canalización y configurar las políticas y funciones de la cuenta para implementar la pila en otra AWS cuenta. Para obtener más información, consulta ¿Cómo puedo CodePipeline implementar una AWS CloudFormation pila en otra cuenta?<br>• Para configurar el cifrado del lado del servidor para el depósito de artefactos de S3 de tu canalización, puedes usar la clave de KMS AWS administrada predeterminada o crear una clave de KMS administrada por el cliente y configurar la política del depósito para usar la clave de cifrado. Para obtener más información, consulte Configurar el cifrado del lado del servidor para los artefactos almacenados en Amazon S3 para CodePipeline.<br><br>Si se trata de una AWS KMS key, puede utilizar el ID de la clave, el ARN de la clave o el ARN del alias. |

> **Note**
>
> Los alias se reconocen únicamente en la cuenta que ha creado la clave de KMS. Para las acciones entre cuentas, solo puedes usar el ID de clave o el ARN de clave para identificar la clave. Cross-account las acciones implican el uso del rol de la otra cuenta (AccountB), por lo que al especificar el ID de clave se utilizará la clave de la otra cuenta (AccountB).

# Ejemplos de la comunidad

En las siguientes secciones se incluyen enlaces a entradas de blogs, artículos y ejemplos proporcionados en la comunidad.

> **Note**
>
> Estos enlaces se proporcionan únicamente con fines informativos y no deben considerarse una lista exhaustiva ni una aprobación del contenido de los ejemplos. AWS no es responsable del contenido o la precisión del contenido externo.

Temas

- Ejemplos de integración: entradas de blogs

## Ejemplos de integración: entradas de blogs

- Seguimiento del estado de la AWS CodePipeline compilación desde el repositorio de Git de terceros

  Aprenda a configurar recursos que muestren el estado de su proceso y de sus acciones de compilación en un repositorio de terceros, lo que permitirá al desarrollador realizar un seguimiento del estado sin tener que cambiar de contexto.

  Fecha de publicación: marzo de 2021

- Completo CI/CD con AWS CodeCommitAWS CodeBuild, AWS CodeDeploy, y AWS CodePipeline

Aprenda a configurar una canalización que utilice los CodeDeploy servicios CodeCommit, CodePipeline CodeBuild, y para compilar, crear e instalar una aplicación Java con control de versiones en un conjunto de instancias Linux de Amazon EC2.

Fecha de publicación: septiembre de 2020

- Probar y crear CI/CD canalizaciones para AWS Step Functions

  Aprenda a configurar los recursos que coordinarán su máquina de estados de Step Functions y su canalización.

  Fecha de publicación: marzo de 2020

- Implementación mediante DevSecOps CodePipeline

  Aprenda a utilizar una CI/CD canalización CodePipeline para automatizar los controles de seguridad preventivos y de detección. En esta publicación, se explica cómo utilizar una canalización para crear un grupo de seguridad sencillo y realizar comprobaciones de seguridad durante las fases de origen, prueba y producción para mejorar la seguridad de sus AWS cuentas.

  Fecha de publicación: marzo de 2017

- Implementación continua en Amazon ECS mediante CodePipeline Amazon ECR y CodeBuild CloudFormation

  Aprenda a crear un canalización de implementación continua en Amazon Elastic Container Service (Amazon ECS). Las aplicaciones se entregan como contenedores Docker mediante CodePipeline Amazon ECR y. CodeBuild CloudFormation

  - Descargue una CloudFormation plantilla de muestra e instrucciones para usarla para crear su propia canalización de implementación continua desde el repositorio ECS Reference Architecture: Continuous Deployment en adelante. GitHub

  Fecha de publicación: enero de 2017

- Continuous Deployment for Serverless Applications

  Aprenda a usar una colección de Servicios de AWS para crear una canalización de despliegue continuo para sus aplicaciones sin servidor. Utilizará el modelo de aplicaciones sin servidor (SAM) para definir la aplicación y sus recursos y CodePipeline para organizar la implementación de la aplicación.

  - Vea una aplicación de ejemplo escrita en Go con la plataforma Gin y un shim de proxy de API Gateway.

  Fecha de publicación: diciembre de 2016

- Escalar DevOps las implementaciones con Dynatrace CodePipeline

  Descubra cómo usar las soluciones de monitoreo de Dynatrace para escalar los procesos CodePipeline, analizar automáticamente las ejecuciones de las pruebas antes de que se ejecute el código y mantener los plazos de entrega óptimos.

  Fecha de publicación: noviembre de 2016

- Cree una canalización para su uso y AWS Elastic Beanstalk CodePipeline AWS CloudFormation CodeCommit

  Aprenda a implementar la entrega continua en una CodePipeline canalización para una aplicación en AWS Elastic Beanstalk. Todos los AWS recursos se aprovisionan automáticamente mediante el uso de una CloudFormation plantilla. Este tutorial también incorpora CodeCommit y AWS Identity and Access Management (IAM).

  Fecha de publicación: mayo de 2016

- Automatiza CodeCommit y entra CodePipeline CloudFormation

  Se utiliza CloudFormation para automatizar el aprovisionamiento de AWS recursos para una canalización de entrega continua que utiliza CodeCommit, CodePipeline CodeDeploy, y AWS Identity and Access Management.

  Fecha de publicación: abril de 2016

- Cree una Cross-Account canalización en AWS CodePipeline

  Aprenda a automatizar el aprovisionamiento del acceso entre cuentas a las canalizaciones en AWS CodePipeline mediante AWS Identity and Access Management. Incluye ejemplos en una CloudFormation plantilla.

  Fecha de publicación: marzo de 2016

- Explorando la parte ASP.NET principal 2: Entrega continua

  Aprenda a crear un sistema de entrega continua completo para una aplicación ASP.NET principal utilizando CodeDeploy y AWS CodePipeline.

Fecha de publicación: marzo de 2016

- Cree una canalización mediante la AWS CodePipeline consola

  Aprenda a usar la AWS CodePipeline consola para crear una canalización de dos etapas en un tutorial basado en la. AWS CodePipeline Tutorial: Crear una canalización de cuatro etapas

  Fecha de publicación: marzo de 2016

- Burlándose de Pipelines AWS CodePipeline con AWS Lambda

  Aprenda a invocar una función Lambda que le permita visualizar las acciones y etapas de CodePipeline un proceso de entrega de software a medida que lo diseña, antes de que la canalización entre en funcionamiento. Mientras diseña la estructura de la canalización, puede usar la función de Lambda para probar si la canalización se completará correctamente.

  Fecha de publicación: febrero de 2016

- Ejecutar AWS Lambda funciones en uso CodePipeline AWS CloudFormation

  Aprenda a crear una AWS CloudFormation pila que aprovisione todos los AWS recursos utilizados en la tarea de la guía del usuarioInvoca un AWS Lambda función en una canalización en CodePipeline.

  Fecha de publicación: febrero de 2016

- Aprovisionamiento de CodePipeline acciones personalizadas en AWS CloudFormation

  Aprenda a utilizar para AWS CloudFormation aprovisionar acciones personalizadas en CodePipeline.

  Fecha de publicación: enero de 2016

- Aprovisionamiento con CodePipeline AWS CloudFormation

  Aprenda a aprovisionar una canalización básica de entrega continua CodePipeline con el uso AWS CloudFormation.

  Fecha de publicación: diciembre de 2015

- Implementar desde CodePipeline hasta OpsWorks usar una acción personalizada y AWS Lambda

  Aprenda a configurar su canalización y la AWS Lambda función que debe implementarse para AWS OpsWorks utilizarla CodePipeline.

Fecha de publicación: julio de 2015

# CodePipeline tutoriales

Tras completar los pasosEmpezar con CodePipeline, puede probar uno de los AWS CodePipeline tutoriales de esta guía del usuario.

Temas

- Tutorial: Implemente en instancias de Amazon EC2 con CodePipeline
- Tutorial: Cree y envíe una imagen de Docker a Amazon ECR con CodePipeline (tipo V2)
- Tutorial: Implemente en Amazon EKS con CodePipeline
- Tutorial: creación de una canalización que ejecute comandos mediante computación (tipo V2)
- Tutorial: Usar etiquetas de Git para iniciar la canalización
- Tutorial: filtrado de solicitudes de extracción por nombres de ramificación para iniciar una canalización (tipo V2)
- Tutorial: Uso de variables a nivel de canalización
- Tutorial: Crear una canalización simple (bucket de S3)
- Tutorial: Crear una canalización simple (CodeCommit repositorio)
- Tutorial: Crear una canalización de cuatro etapas
- Tutorial: configurar una regla de CloudWatch eventos para recibir notificaciones por correo electrónico sobre los cambios en el estado de los oleoductos
- Tutorial: Crea una canalización que compile y pruebe tu aplicación para Android con AWS Device Farm
- Tutorial: Crea una canalización que pruebe tu aplicación para iOS con AWS Device Farm
- Tutorial: Crear una canalización que se implemente en Service Catalog
- Tutorial: Crea una canalización con AWS CloudFormation
- Tutorial: Cree una canalización que utilice variables de AWS CloudFormation acciones de despliegue
- Tutorial: implementación estándar de Amazon ECS con CodePipeline
- Tutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR
- Tutorial: Crear una canalización que implemente una habilidad de Amazon Alexa
- Tutorial: Crear una canalización que utilice Amazon S3 como proveedor de implementación

- Tutorial: Cree una canalización que publique su aplicación sin servidor en el AWS Serverless Application Repository
- Tutorial: Implementaciones de funciones Lambda con CodePipeline
- Tutorial: Uso de variables con acciones de invocación de Lambda
- Tutorial: Usa una acción de AWS Step Functions invocación en una canalización
- Tutorial: Crear una canalización que se utilice AWS AppConfig como proveedor de despliegue
- Tutorial: Utilice un clon completo con una fuente de GitHub canalización
- Tutorial: Usa la clonación completa con una fuente de CodeCommit canalización
- Tutorial: Crea una canalización con AWS CloudFormation StackSets acciones de despliegue
- Tutorial: Cómo crear una regla de verificación de variables para una canalización como una condición de entrada

# Tutorial: Implemente en instancias de Amazon EC2 con CodePipeline

Este tutorial le ayuda a crear una acción de implementación para implementar el código en las instancias CodePipeline que haya configurado en Amazon EC2.

> **Note**
>
> Como parte de la creación de una canalización en la consola, los artefactos utilizarán un depósito de artefactos de CodePipeline S3. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

> **Note**
>
> La acción de implementación de EC2 solo está disponible para canalizaciones de tipo V2.

## Requisitos previos

Para poder usar este tutorial para crear su propia canalización de implementación continua debe tener instalados algunos recursos. Esto es lo que necesita para empezar:

> **Note**
>
> Todos estos recursos deben crearse en la misma región. AWS

- Un repositorio de control de código fuente (se utiliza en este tutorial GitHub) en el que añadirá un `script.sh` archivo de muestra.
- Debe usar un rol de CodePipeline servicio existente que se haya actualizado con los permisos para esta acción. Para actualizar su rol de servicio, consulte Permisos de la política de rol de servicio para la acción de implementación de EC2.

Una vez satisfechos estos requisitos previos, puede continuar con el tutorial y crear su canalización de implementación continua.

## Paso 1: creación de instancias de Amazon EC2 en Linux

En este paso, va a crear las instancias de Amazon EC2 en las que va a implementar una aplicación de ejemplo. Como parte de este proceso, cree un rol de instancia en IAM si aún no ha creado un rol de instancia en la región en la que desea crear los recursos.

Para crear un rol de instancia

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de la consola, elija Roles.
3. Elija Crear rol.
4. En Seleccionar el tipo de entidad de confianza, seleccione Servicio de AWS. En Elegir un caso de uso, seleccione EC2. En Select your use case (Seleccionar su caso de uso), elija EC2. Elija Siguiente.
5. Busque y seleccione la política denominada **`AmazonSSMManagedEC2InstanceDefaultPolicy`**.
6. Busque y seleccione la política denominada **`AmazonSSMManagedInstanceCore`**. Elija Siguiente: Etiquetas.

7. Elija Siguiente: Revisar. Escriba el nombre del rol (por ejemplo, **`EC2InstanceRole`**).

   > **Note**
   >
   > Anote el nombre del rol para utilizarlo en el siguiente paso. Tendrá que elegir este rol cuando cree la instancia.

   > **Note**
   >
   > Agregará permisos a este rol para permitir el acceso al bucket de artefactos de S3 para su canalización una vez creada la canalización.

   Elija Crear rol.

Para lanzar las instancias

1. Abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En la barra de navegación lateral, elija Instancias y seleccione Lanzar instancias en la parte superior de la página.
3. En Name (Nombre), escriba **`MyInstances`**. Esto asigna a la instancia una clave de etiqueta **`Name`** y un valor de etiqueta **`MyInstances`**.
4. En Imágenes de aplicaciones y sistemas operativos (Amazon Machine Image), localice la opción de AMI de Amazon Linux con el AWS logotipo y asegúrese de que está seleccionada. (Esta AMI se describe como AMI de Amazon Linux 2 (HVM) y se denomina "Free tier eligible" (Apta para capa gratuita).)
5. En Tipo de instancia, elija el tipo `t2.micro` apto para la capa gratuita como configuración de hardware de la instancia.
6. En Par de claves (inicio de sesión), seleccione un par de claves o cree uno.
7. En Configuración de red, asegúrese de que el estado sea Habilitar.
8. Amplíe Advanced details (Detalles avanzados). En Perfil de instancia de IAM, elija el rol de IAM que creó en el procedimiento anterior (por ejemplo, **`EC2InstanceRole`**).

> **Note**
>
> No deje el rol de instancia en blanco, ya que esto crea un rol predeterminado y no selecciona el rol que ha creado.

9. En Resumen, en Número de instancias, especifique 2.
10. Seleccione Iniciar instancia.
11. Puede ver el estado del lanzamiento en la página Instances. Al lanzar una instancia, su estado inicial es `pending`. Una vez iniciada la instancia, el estado cambia a `running` y recibe un nombre de DNS público. (Si no aparece la columna DNS público, elija el Show/Hide icono y, a continuación, seleccione DNS público).

## Paso 2: agregación de los permisos del bucket de artefactos al rol de instancia de EC2

Debe actualizar el rol de instancia de EC2 que creó para su instancia a fin de que pueda acceder al bucket de artefactos de su canalización.

> **Note**
>
> Al crear la instancia, se crea o se utiliza un rol de instancia de EC2 existente. Para evitar `Access Denied` errores, debes agregar los permisos del bucket de S3 al rol de instancia para otorgar a la instancia los permisos de la instancia al bucket de CodePipeline artefactos. Cree un rol predeterminado o actualice el rol actual con el permiso `s3:GetObject` limitado al bucket de artefactos correspondiente a la región de su canalización.

1. Navega hasta tu canalización en la CodePipeline consola. Seleccione Configuración. Vea el nombre y la ubicación del almacén de artefactos de una canalización existente. Anote el nombre de recurso de Amazon (ARN) del bucket de artefactos y cópielo.
2. Vaya a la consola de IAM y elija Roles (Roles). Elija el rol de instancia que creó en el paso 1 de este tutorial.
3. En la pestaña Permisos, elija Añadir política en línea.
4. Agregue el siguiente JSON al documento de política y sustituya el valor del campo `Resource` por el ARN del bucket.

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

5. Elija Actualizar.

## Paso 3: agregación del archivo de script al repositorio

Pegue este texto de ejemplo para crear el archivo `script.sh` para el paso posterior al script de la implementación.

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

Para añadir un **`script.sh`** archivo a tu repositorio fuente

1. Abra un editor de texto y luego copie y pegue el archivo anterior en un nuevo archivo.
2. Confirme la operación e inserte el archivo `script.sh` en el repositorio de código fuente.

   a. Añada el archivo.

   ```
   git add .
   ```

   b. Valide el cambio con.

   ```
   git commit -m "Adding script.sh."
   ```

   c. Envíe la confirmación.

   ```
   git push
   ```

   Anote la ruta en su repositorio.

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

# Paso 4: creación de una canalización

Usa el CodePipeline asistente para crear tus etapas de canalización y conectar tu repositorio de origen.

Para crear la canalización

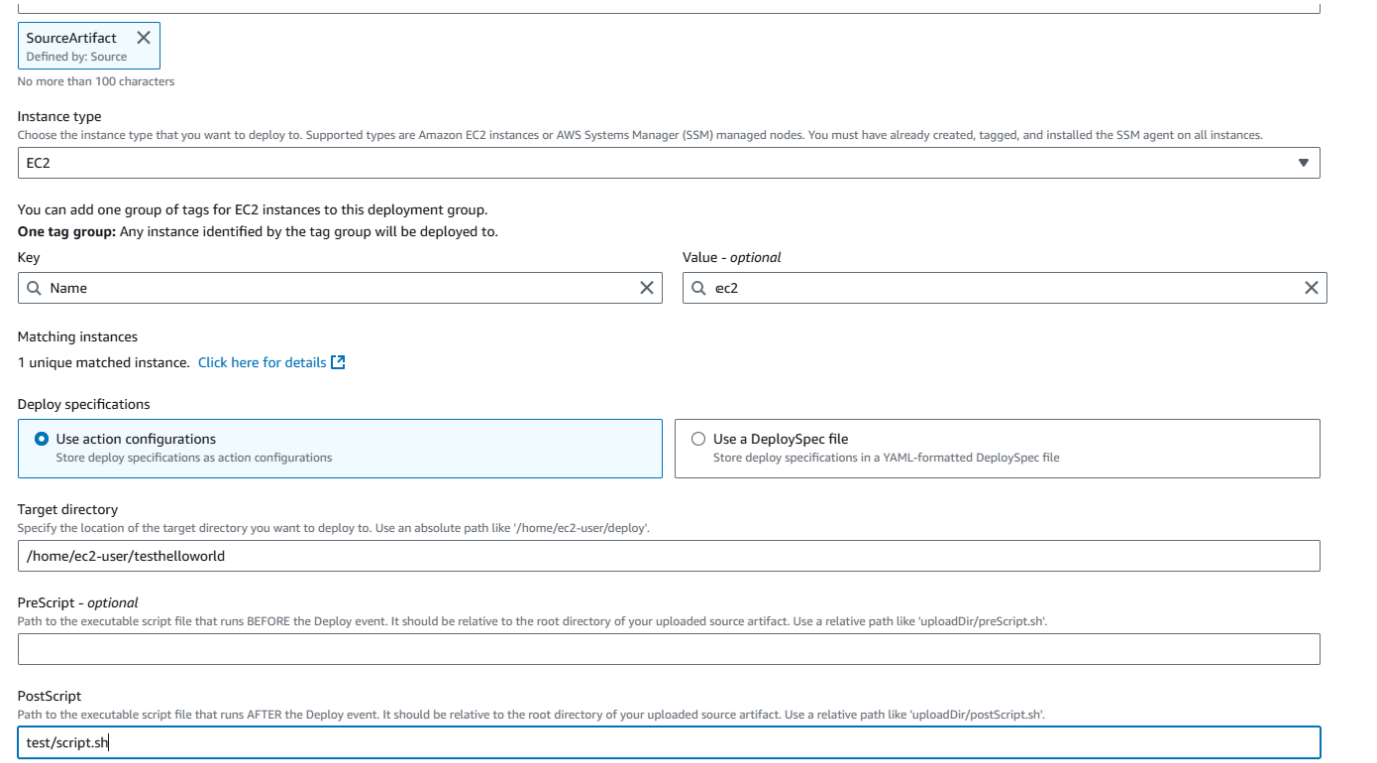
1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyPipeline`**.
5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.
6. En Rol de servicio, elija Usar el rol de servicio existente y, a continuación, elija el rol de CodePipeline servicio que se ha actualizado con los permisos necesarios para esta acción. Para configurar su rol CodePipeline de servicio para esta acción, consultePermisos de la política de rol de servicio para la acción de implementación de EC2.
7. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
8. En la página Paso 3: agregar la etapa de origen, agregue una etapa de origen:

   a. En Proveedor de origen, elija GitHub (a través de GitHub la aplicación).

   b. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para tu acción GitHub de origen, consultaGitHub conexiones.

   c. En Repository name (Nombre de repositorio), elija el nombre de su repositorio de GitHub .

   Elija Siguiente.
9. En la página Paso 4: agregación de la etapa de compilación, seleccione Omitir.
10. En la página Paso 5: agregación de la etapa de implementación, elija EC2.



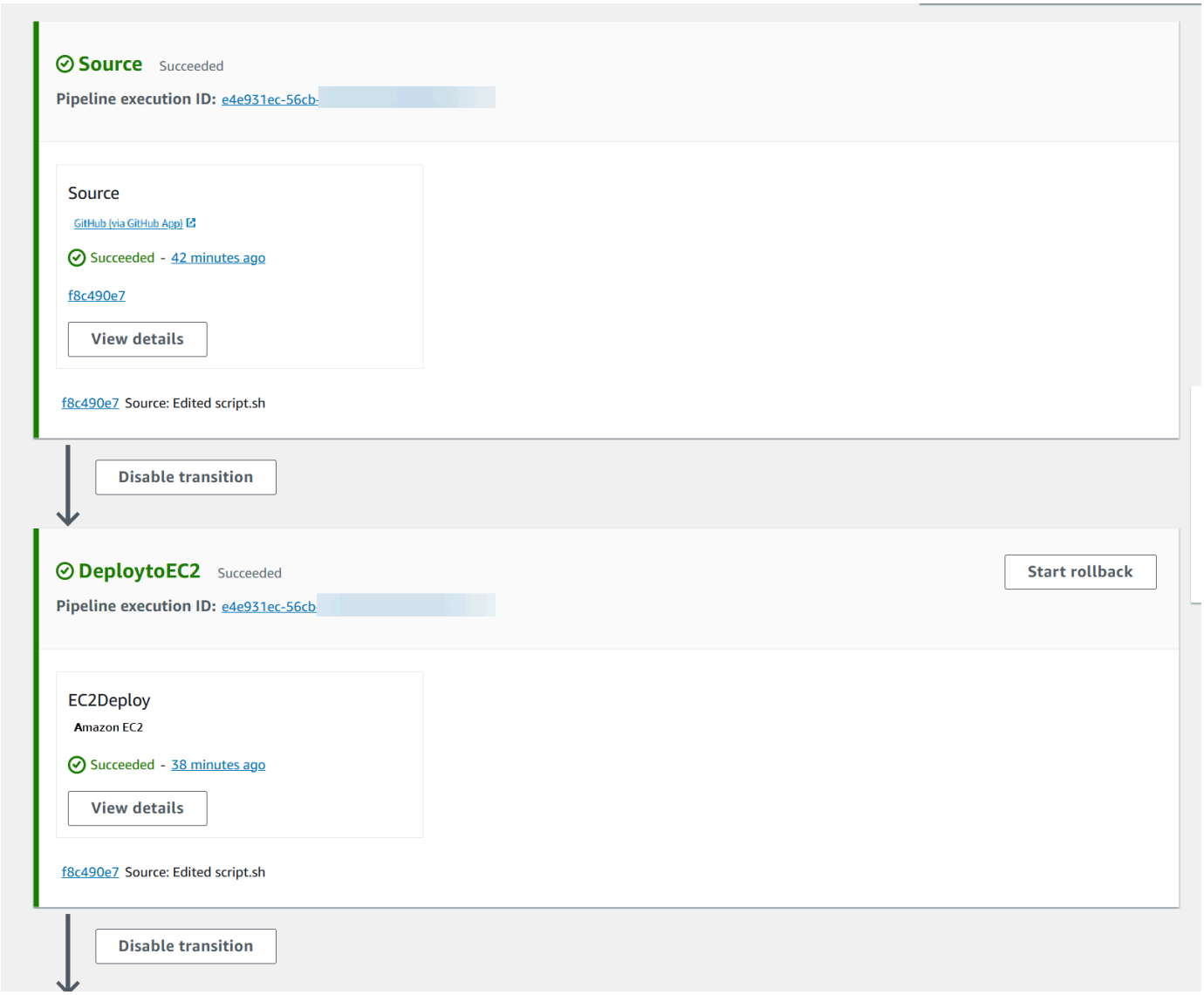

a. En el directorio de destino, especifique el directorio de la instancia en la que desee realizar la implementación, por ejemplo, `/home/ec2-user/testhelloworld`.

> **Note**
>
> Especifique el directorio de implementación que desea que utilice la acción en la instancia. La acción automatizará la creación del directorio especificado en la instancia como parte de la implementación.

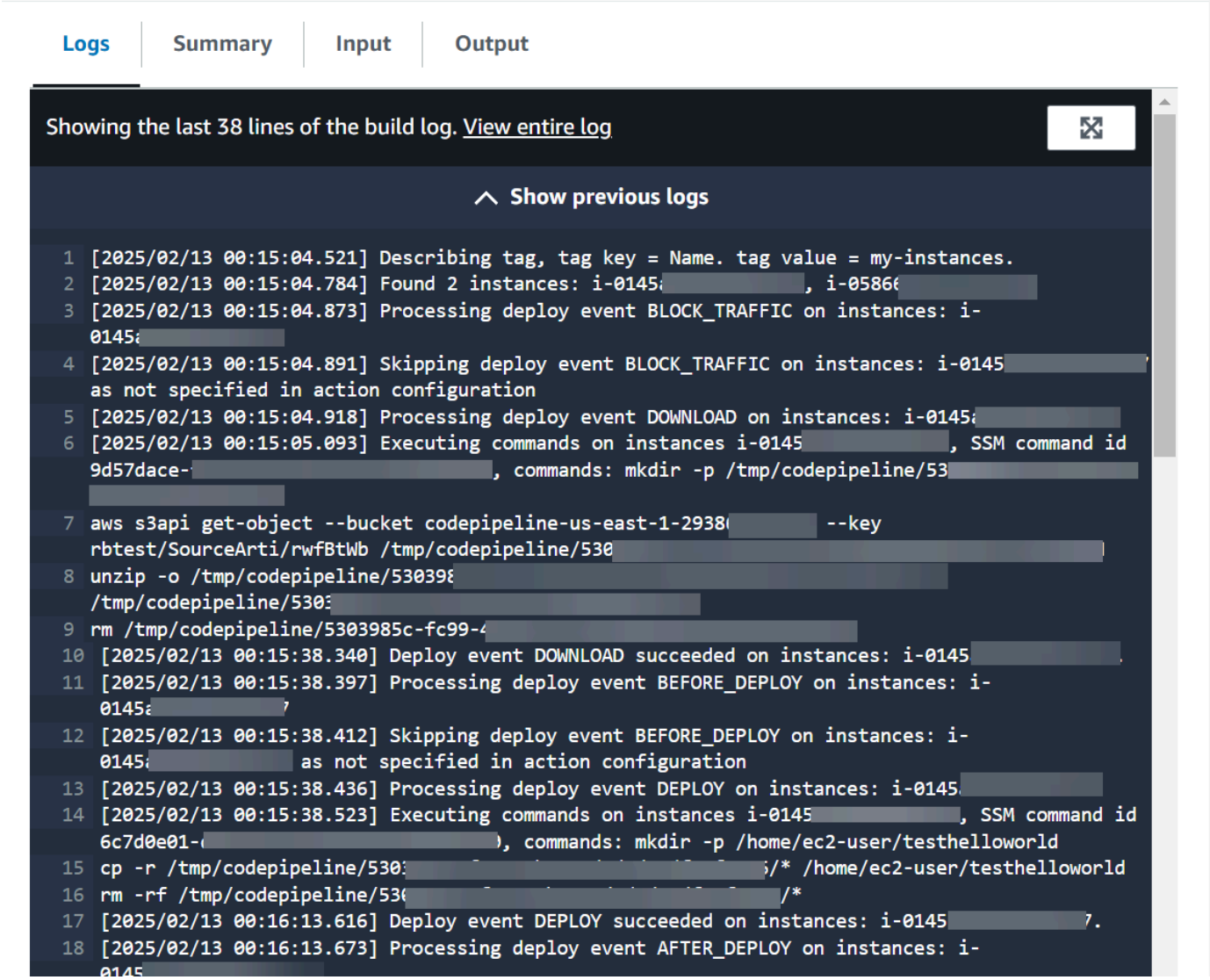
b. Para PostScript, introduzca la ruta y el nombre de archivo del script, por ejemplo`test/script.sh`.

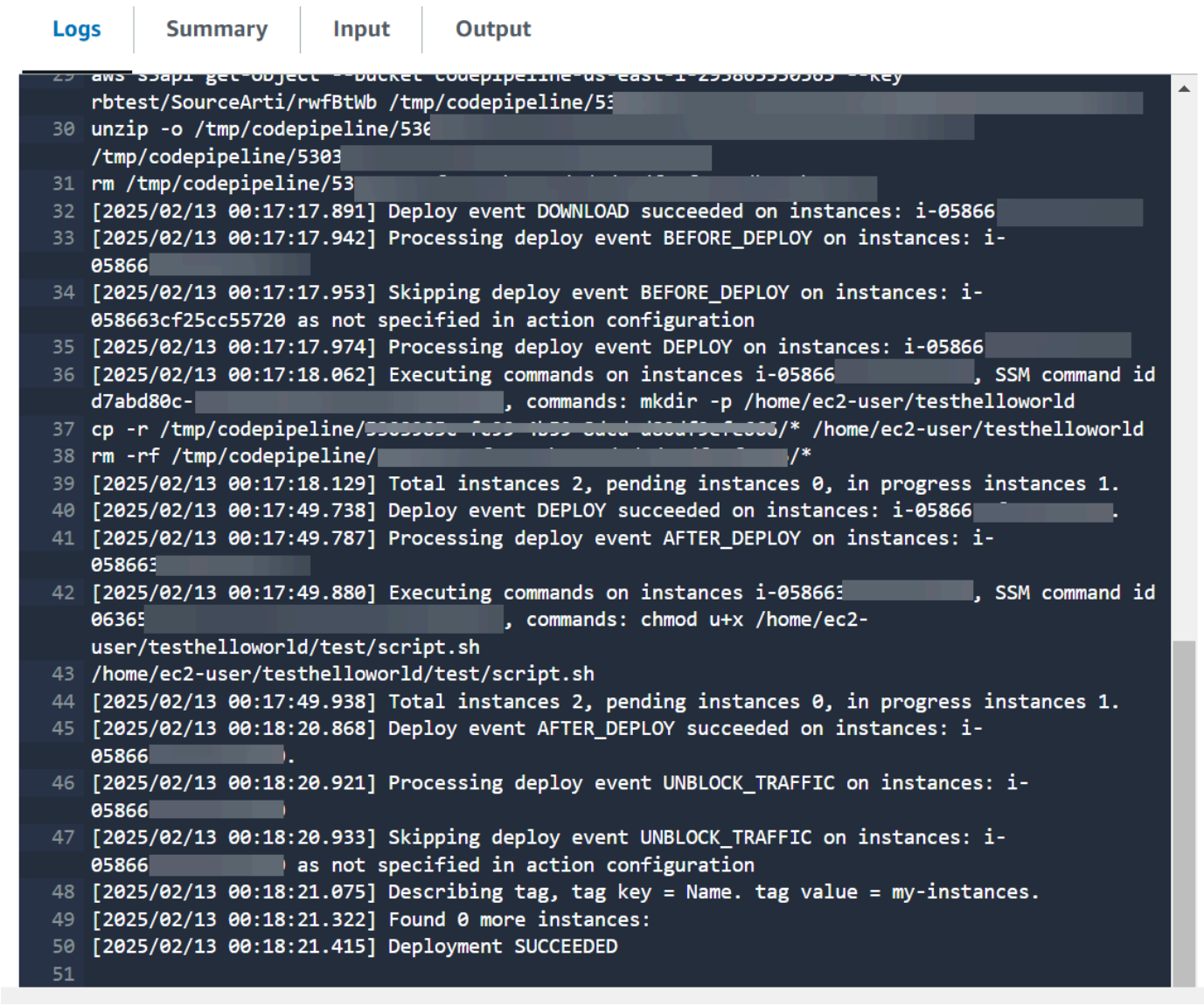
c. Elija Siguiente.

11. En la página Step 6: Review, revise la configuración de la canalización y elija Create pipeline para crear la canalización.




12. Cuando la canalización se haya ejecutado correctamente, elija Ver detalles para consultar los registros de la acción y ver la salida de la acción de computación administrada.


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




## Paso 5: Pon a prueba tu canalización

Su canalización debe tener todo lo necesario para ejecutar una implementación AWS continua nativa de extremo a extremo. Ahora, pruebe su funcionalidad enviando un cambio de código al repositorio de código fuente.

Para probar la canalización

1. Realice una modificación del código en el repositorio de código fuente configurado, valide y envíe el cambio.
2. Abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

3. Seleccione su canalización de la lista.
4. Vea el progreso en la canalización a través de sus etapas. La canalización debería completarse y la acción implementar el script en las instancias.
5. Para obtener más información sobre la solución de problemas, consulte La acción de despliegue de EC2 falla y muestra el mensaje de error `No existe tal archivo`.

# Tutorial: Cree y envíe una imagen de Docker a Amazon ECR con CodePipeline (tipo V2)

Este tutorial le ayuda a crear una acción de compilación CodePipeline que ejecute y envíe su imagen de Docker a Amazon ECR tras un cambio en el código fuente. En este tutorial, también se muestra cómo agregar una acción de implementación de Amazon ECS que implemente la imagen insertada.

> **Important**
>
> Como parte de la creación de una canalización en la consola, los artefactos utilizarán un depósito de artefactos de S3. CodePipeline (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

> **Note**
>
> Este tutorial trata sobre la acción de ECRBuildAndPublish creación de una CodePipeline canalización con un repositorio de GitHub origen y una acción estándar de Amazon ECS para la implementación en un clúster de Amazon ECS. Para ver un tutorial en el que se usa una canalización con un repositorio de imágenes de ECR como fuente para que un Amazon ECS CodeDeploy blue/green implemente acciones en él CodePipeline, consulteTutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR.

> **Important**
>
> Esta acción utiliza la CodeBuild informática CodePipeline gestionada para ejecutar comandos en un entorno de compilación. Si ejecuta la acción de Comandos, se le cobrarán cargos por separado en AWS CodeBuild.

## Requisitos previos

Para poder usar este tutorial para crear su propia canalización de implementación continua debe tener instalados algunos recursos. Esto es lo que necesita para empezar:

> **Note**
>
> Todos estos recursos deben crearse en la misma AWS región.

- Un repositorio de control de código fuente (usado en este tutorial GitHub) donde añadirá lo siguiente para este tutorial:
  - En el paso 1, añadirá un Dockerfile de muestra a su repositorio de origen como artefacto de entrada para la acción de ECRBuildAndPublish compilación. CodePipeline
  - En el paso 2, añadirá un archivo imagedefinitions.json de muestra a su repositorio de origen como requisito para la acción de implementación estándar de Amazon ECS en. CodePipeline
- Un repositorio de imágenes de Amazon ECR que contiene una imagen compilada desde el Dockerfile. Para obtener más información, consulte Creación de un repositorio e Inserción de una imagen en la Guía del usuario de Amazon Elastic Container Registry.
- Un servicio y un clúster de Amazon ECS creados en la misma región que el repositorio de imágenes. Para obtener más información, consulte Creación de un clúster y Creación de un servicio en la Guía para desarrolladores de Amazon Elastic Container Service.

Una vez satisfechos estos requisitos previos, puede continuar con el tutorial y crear su canalización de implementación continua.

## Paso 1: agregación de un Dockerfile al repositorio de origen

En este tutorial, se utiliza la ECRBuildAndPublish acción para crear su imagen de Docker y enviarla a Amazon ECR. La acción informática gestionada se CodePipeline utiliza CodeBuild para ejecutar los

comandos de inicio de sesión del ECR y enviar imágenes. No necesitas añadir un `buildspec.yml` archivo a tu repositorio de código fuente para saber CodeBuild cómo hacerlo. Para este ejemplo, solo debe proporcionar el Dockerfile en el repositorio tal y como se indica a continuación.

Pegue este texto de ejemplo para crear su archivo `Dockerfile`. Este Dockerfile de ejemplo es el mismo que el utilizado en las instrucciones de las imágenes de ECR en los requisitos previos.

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

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

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

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

EXPOSE 80

CMD /root/run_apache.sh
```

Para añadir un **`Dockerfile`** archivo a su repositorio de origen

1. Abra un editor de texto y luego copie y pegue el Dockerfile anterior en un archivo nuevo.
2. Confirme la operación e inserte el archivo `Dockerfile` en el repositorio de código fuente.
   a. Añada el archivo.

      ```
      git add .
      ```

   b. Valide el cambio con.

      ```
      git commit -m "Adding Dockerfile."
      ```

   c. Envíe la confirmación.

```
git push
```

Asegúrese de colocar el archivo en el nivel raíz del repositorio.

```
/ Dockerfile
```

## Paso 2: agregación del archivo imagedefinitions.json a su repositorio de origen

Este tutorial utiliza la acción de implementación estándar de Amazon ECS CodePipeline para implementar su contenedor en su clúster de Amazon ECS. La acción de implementación estándar de Amazon ECS requiere un archivo imagedefinitions.json que contenga el nombre y el URI de la imagen. Para obtener más información sobre el archivo imagedefinitions.json, consulte Archivo imagedefinitions.json para las acciones de implementación estándar de.

Pegue este texto de ejemplo para crear su archivo `imagedefinitions.json`. Use el nombre del Dockerfile, por ejemplo, `hello-world`, y use el URI del repositorio de Amazon ECR donde está almacenada la imagen.

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

Para agregar un **`imagedefinitions.json`** archivo a tu repositorio fuente

1. Abra un editor de texto y luego copie y pegue el archivo anterior en un archivo nuevo.
2. Confirme la operación e inserte el archivo `imagedefinitions.json` en el repositorio de código fuente.

   a. Añada el archivo.

   ```
   git add .
   ```

b. Valide el cambio con.

```
git commit -m "Adding imagedefinitions.json."
```

c. Envíe la confirmación.

```
git push
```

Asegúrese de colocar el archivo en el nivel raíz del repositorio.

```
/ imagedefinitions.json
```

## Paso 3: creación de una canalización

Usa el CodePipeline asistente para crear tus etapas de canalización y conectar tu repositorio de origen.

Para crear la canalización

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyPipeline`**.
5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.
6. En Función de servicio, seleccione Nueva función de servicio CodePipeline para poder crear una función de servicio en IAM.
7. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
8. En la página Paso 3: agregar la etapa de origen, agregue una etapa de origen:

a. En Proveedor de origen, elija GitHub (a través de GitHub la aplicación).

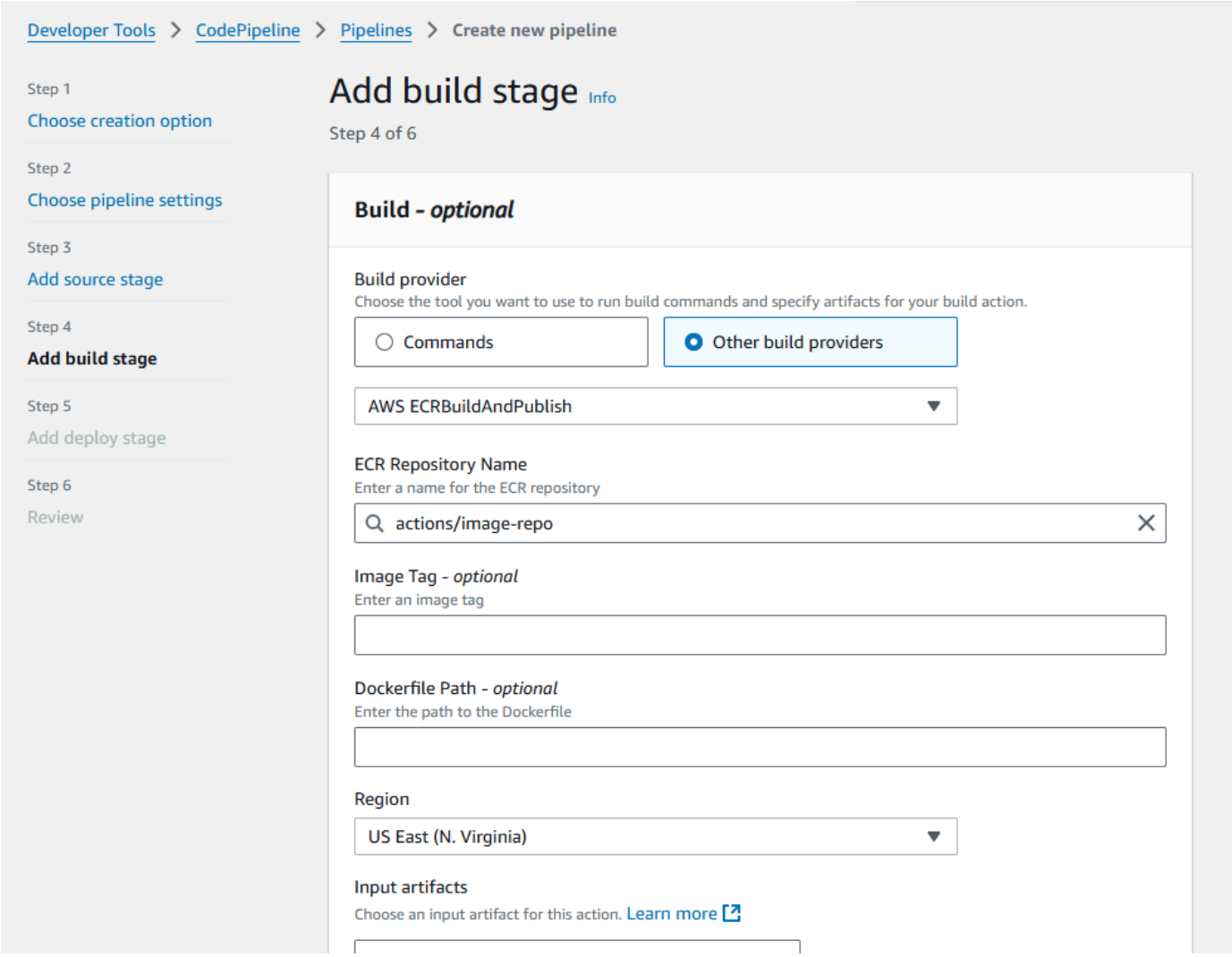
b. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para tu acción GitHub de origen, consultaGitHub conexiones.

c. En Repository name (Nombre de repositorio), elija el nombre de su repositorio de GitHub .

d. En la Ramificación predeterminada, elige la ramificación que quiere especificar cuando la canalización se inicie manualmente o con un evento de origen que no sea una etiqueta de Git. Si la fuente del cambio no es el desencadenador o si la ejecución de una canalización se inició manualmente, el cambio utilizado será la confirmación de HEAD desde la ramificación predeterminada.

Elija Siguiente.

9. En la página Paso 4: Añadir fase de compilación, elija Otros proveedores de compilación y, a continuación, elija ECRBuildAndPublish.




a. En Nombre del repositorio de ECR, elija su repositorio de imágenes.

b. Elija Siguiente.

10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.

11. En la página Paso 6: agregación de la etapa de implementación, elija Omitir etapa de implementación. Agregará la acción de ECS en el paso siguiente.

12. En la página Paso 7: Revisión, revisa la configuración de tu canalización y elige Crear canalización para crear la canalización.

13. Edite su canalización para agregar la acción de implementación de Amazon ECS a la canalización:

a. En la parte superior derecha, elija Edit (Editar).

b. En la parte inferior del diagrama, seleccione + Add stage (Añadir etapa). En Nombre de la etapa, escriba un nombre; por ejemplo, **`Deploy`**.

c. Elija + Add action group (Añadir grupo de acciones).

d. En Nombre de la acción, escriba un nombre.

e. En Proveedor de acción, elija Amazon ECS. En el campo Region (Región) conserve el valor predeterminado de la región de la canalización.

f. En Artefactos de entrada, elija el artefacto de entrada de la etapa de origen, como `SourceArtifact`.

g. En Nombre del clúster, elija el clúster de en el que se ejecuta el servicio.

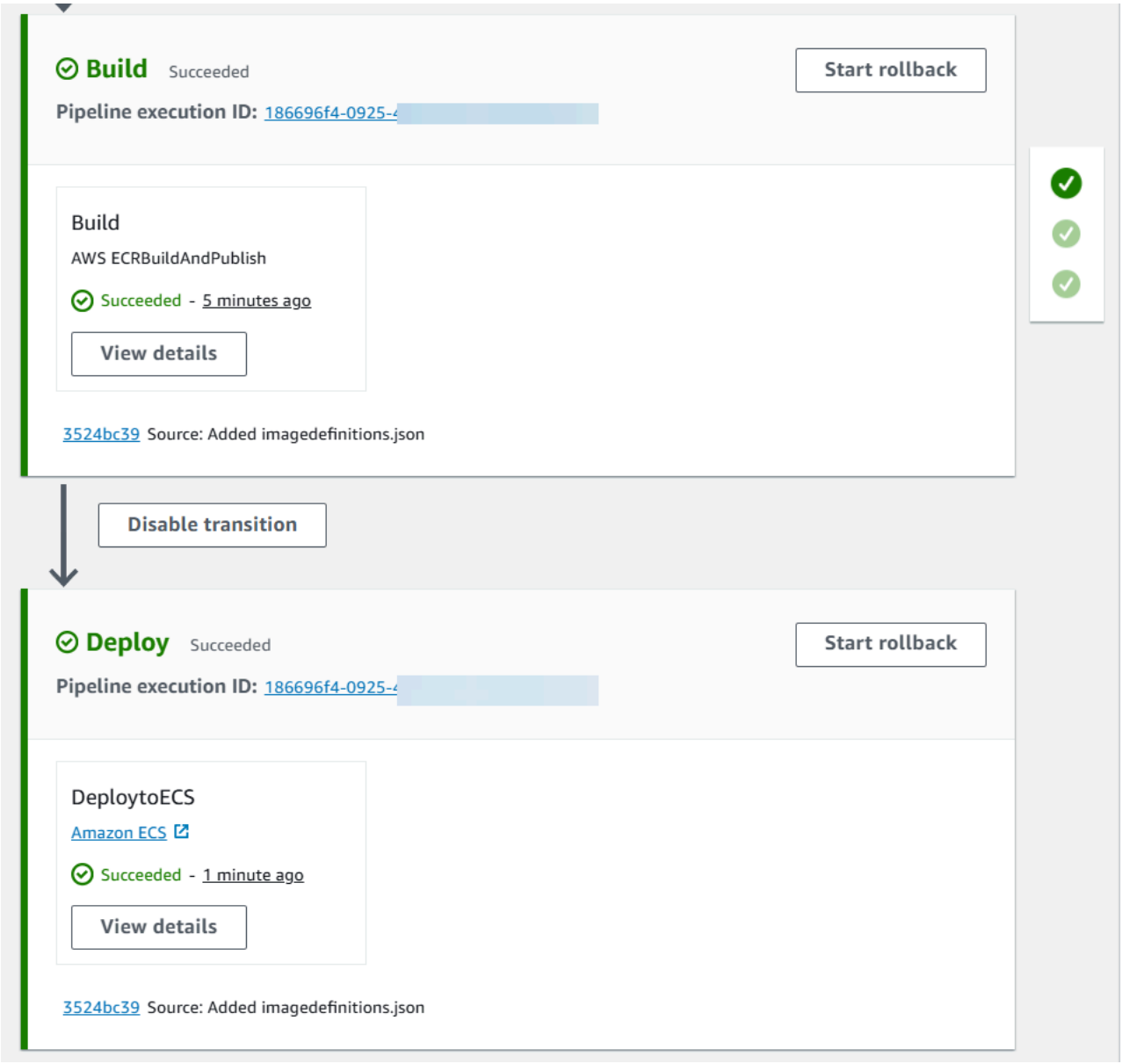
h. En Nombre de servicio, elija el servicio que desea actualizar.

i. Seleccione Save.

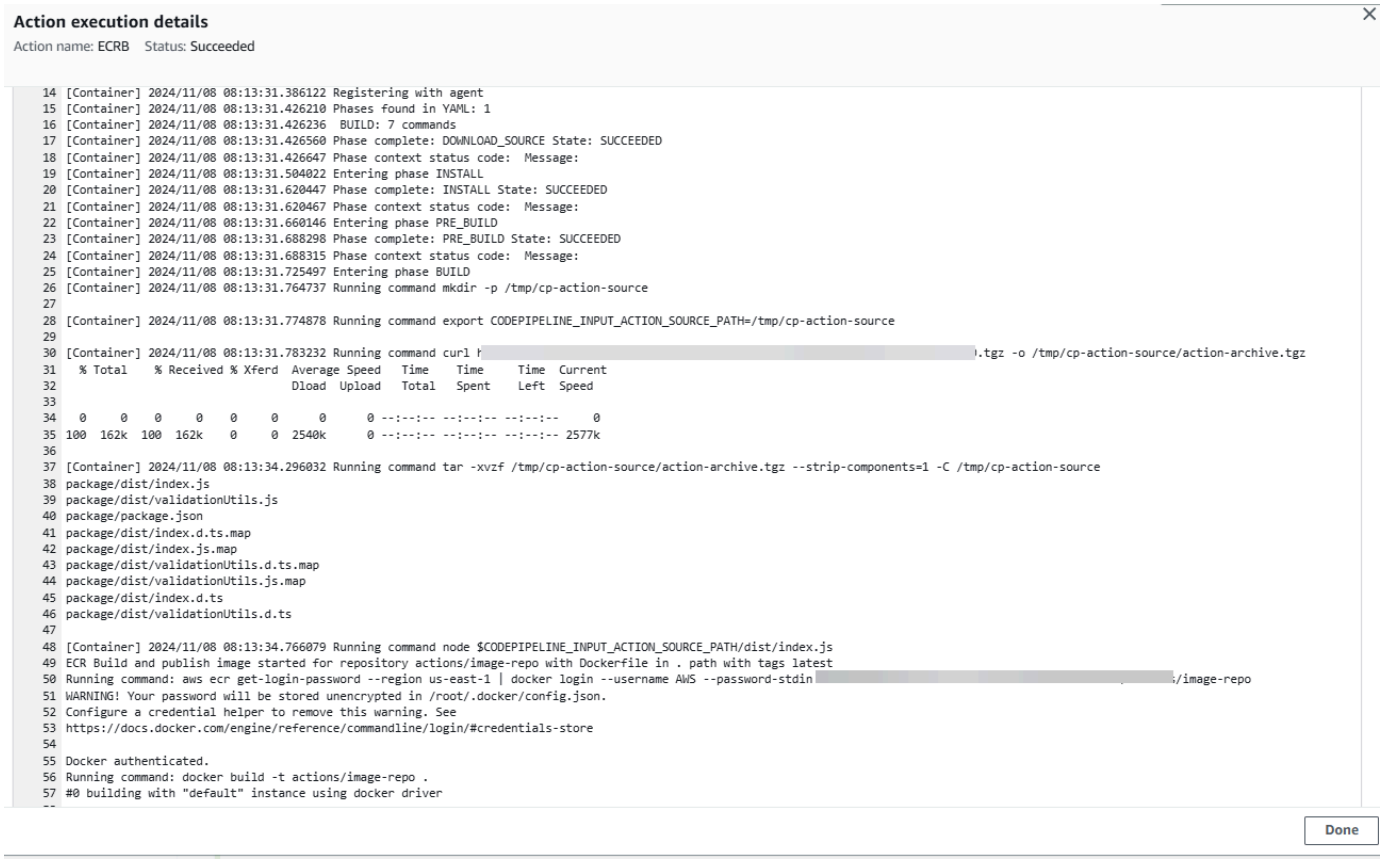
j. En la etapa que está editando, elija Done (Listo). En el panel de AWS CodePipeline , elija Save (Guardar) y, a continuación, elija Save (Guardar) cuando aparezca el mensaje de advertencia.

k. Para enviar los cambios y comenzar una compilación de canalización, seleccione Publicar modificación y, a continuación, Publicar.

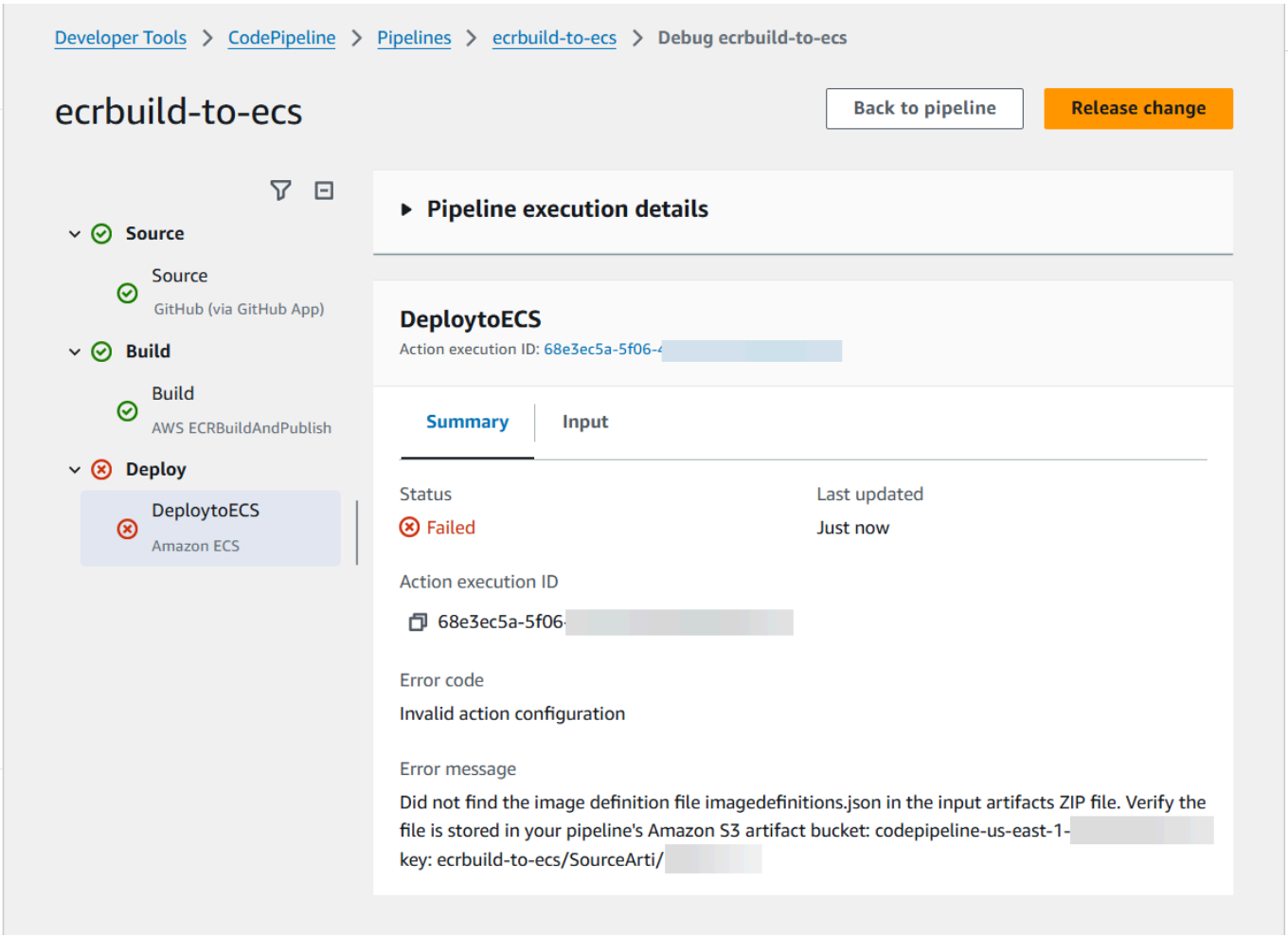
14. Una vez ejecutada la canalización, consulte la estructura y el estado de la canalización.




15. Cuando la canalización se haya ejecutado correctamente, elija Ver detalles para consultar los registros de la acción y ver la salida de la acción de computación administrada.




16. Solucione los problemas de cualquier acción fallida. Por ejemplo, la acción de implementación de ECS puede fallar si el archivo imagedefinitions.json no está en el repositorio de origen. A continuación, se muestra un ejemplo del mensaje de error que aparece cuando falta el archivo imagedefinitions.json.




## Paso 4: Probar la canalización

Tu canalización debe tener todo lo necesario para ejecutar una implementación AWS continua nativa de extremo a extremo. Ahora, pruebe su funcionalidad enviando un cambio de código al repositorio de código fuente.

Para probar la canalización

1. Realice una modificación del código en el repositorio de código fuente configurado, valide y envíe el cambio.
2. Abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
3. Seleccione su canalización de la lista.

4. Vea el progreso en la canalización a través de sus etapas. Su canalización debería completarse y su acción insertar la imagen de Docker en ECR que se creó a partir de la modificación del código.

# Tutorial: Implemente en Amazon EKS con CodePipeline

Este tutorial le ayuda a crear una acción de implementación para implementar el código en un clúster CodePipeline que haya configurado en Amazon EKS.

La acción de EKS admite clústeres de EKS públicos y privados. Los clústeres privados son el tipo recomendado por EKS, pero se admiten ambos.

> **Note**
>
> Como parte de la creación de una canalización en la consola, los artefactos utilizarán un depósito de artefactos de CodePipeline S3. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

> **Note**
>
> Esta acción utiliza la CodeBuild informática CodePipeline gestionada para ejecutar comandos en un entorno de compilación. Si ejecuta la acción de Comandos, se le cobrarán cargos por separado en AWS CodeBuild.

> **Note**
>
> La acción de implementación de EKS solo está disponible para canalizaciones de tipo V2.

## Requisitos previos

Para poder usar este tutorial para crear su propia canalización de implementación continua debe tener instalados algunos recursos. Esto es lo que necesita para empezar:

> **Note**
>
> Todos estos recursos deben crearse en la misma AWS región.

- Un repositorio de control de código fuente (se utiliza en este tutorial GitHub) en el que añadirá un `deployment.yaml` archivo de muestra.
- Debes usar un rol de CodePipeline servicio existente que actualizarás con los permisos para esta acción que Paso 3: Actualiza la política de roles de CodePipeline servicio en IAM se indican a continuación. Los permisos necesarios dependen del tipo de clúster que cree. Para obtener más información, consulte Permisos para las políticas de roles de servicio.
- Una imagen funcional y una etiqueta de repositorio que haya insertado en ECR o en su repositorio de imágenes.

Una vez satisfechos estos requisitos previos, puede continuar con el tutorial y crear su canalización de implementación continua.

## Paso 1: creación de un clúster en Amazon EKS (opcional)

Si lo desea, puede crear un clúster de EKS con un punto de conexión público o privado.

En los siguientes pasos, se crea un clúster público o privado en EKS. Este paso es opcional si ya ha creado el clúster.

### Creación de un clúster público en Amazon EKS

En este paso, se crea un clúster en EKS.

Creación de un clúster público

1. Abra la consola de EKS y elija Crear clúster.
2. En Nombre, asigne un nombre al clúster. Elija Next (Siguiente).
3. Seleccione Create (Crear).

### Creación de un clúster privado en Amazon EKS

En caso de que decidas crear un clúster con un punto final privado, asegúrate de adjuntar únicamente las subredes privadas y de que tengan una conexión a Internet.

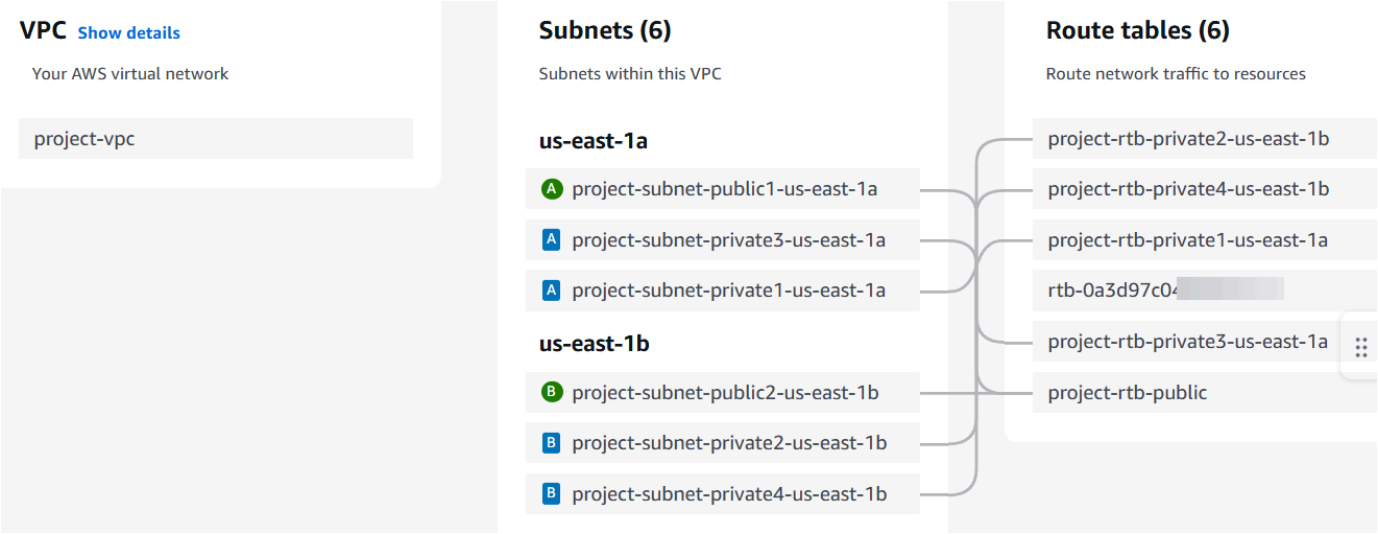
Realice los cinco pasos secundarios siguientes para crear un clúster con un punto de conexión privado.

Creación de una VPC en la consola

1. Abra la consola de VPC y elija Crear VPC.
2. En Configuración de VPC, seleccione VPC y más.
3. Elija crear una subred pública y cuatro subredes privadas. Seleccione Creación de VPC.
4. En la página de subredes, elija Privada.

Determinación de las subredes privadas en la VPC

1. Navegue hasta la VPC y seleccione el ID de VPC para abrir la página de detalles de la VPC.
2. En la página de detalles de la VPC, elija la pestaña Mapa de recursos.
3. Vea el diagrama y anote sus subredes privadas. Las subredes se muestran con etiquetas que indican su estado público o privado, y cada subred está asignada a una tabla de enrutamiento.




   Tenga en cuenta que en un clúster privado todas las subredes serán privadas.
4. Cree una subred pública para alojar la puerta de enlace NAT. Solo puede adjuntar una gateway de Internet a una VPC a la vez.

Creación de una puerta de enlace NAT en la subred pública

1. En la subred pública, cree una puerta de enlace NAT. Navegue hasta la consola de VPC y, a continuación, elija Puertas de enlace de Internet. Elija Crear puerta de enlace de Internet.

2. En Nombre, escriba el nombre de la puerta de enlace de Internet. Elija Crear puerta de enlace de Internet.

Actualice la tabla de enrutamiento de la subred privada para que dirija el tráfico a la puerta de enlace NAT.

Agregación de la puerta de enlace NAT a las tablas de enrutamiento de las subredes privadas

1. Navegue hasta la consola de VPC y, a continuación, elija Subredes.
2. Para cada subred privada, elíjala y, a continuación, elige la tabla de rutas para esa subred en la página de detalles. Elige  Editar tabla de rutas.
3. Actualice la tabla de enrutamiento de la subred privada para que dirija el tráfico de Internet a la puerta de enlace NAT. Seleccione Añadir ruta. Elija Puerta de enlace NAT entre las opciones para agregar. Elija la puerta de enlace de Internet que creó.
4. Para la subred pública, cree una tabla de enrutamiento personalizada. Verifique que la lista de control de acceso (ACL) de red de la subred pública permita el tráfico de entrada de la subred privada.
5. Seleccione Save changes (Guardar cambios).

En este paso, se crea un clúster en EKS.

Creación de un clúster privado

1. Abra la consola de EKS y elija Crear clúster.
2. En Nombre, asigne un nombre al clúster. Elija Siguiente.
3. Especifique su VPC y otra información de configuración. Seleccione Crear.

El clúster de EKS puede ser público o privado. Este paso es para clústeres que tienen SOLO un punto de conexión privado. Asegúrese de que el clúster sea privado.

## Paso 2: configuración de un clúster privado en Amazon EKS

Este paso solo es aplicable si ha creado un clúster privado. Este paso es para clústeres que tienen SOLO un punto de conexión privado.

Configuración de su clúster

1. Asocie las subredes privadas únicamente en el clúster de EKS bajo la pestaña Redes. Asocie las subredes privadas capturadas en la sección Determinación de las subredes privadas en la VPC de Paso 1: creación de un clúster en Amazon EKS (opcional).
2. Asegúrate de que las subredes privadas tengan acceso a Internet, ya que CodePipeline almacenan y recuperan los artefactos del depósito de artefactos de S3 para tu canalización.

## Paso 3: Actualiza la política de roles de CodePipeline servicio en IAM

En este paso, actualizará un rol de CodePipeline servicio existente, por ejemplo**`cp-service-role`**, con los permisos necesarios CodePipeline para conectarse a su clúster. Si no tiene un rol existente, cree uno nuevo.

Actualice su rol de CodePipeline servicio con los siguientes pasos.

Para actualizar su política CodePipeline de funciones de servicio

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de la consola, elija Roles.
3. Busque su rol CodePipeline de servicio, por ejemplo**`cp-service-role`**.
4. Agregue una nueva política insertada.
5. En el Editor de políticas, especifique lo siguiente.

   - Para un clúster público, agregue los siguientes permisos.

     JSON

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

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

```
            "Sid": "EksVpcClusterPolicy",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcs"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

- Para un clúster privado, agregue los siguientes permisos. Los clústeres privados requerirán permisos adicionales para su VPC, según corresponda.

  JSON

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

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

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

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

6. Elija Actualizar política.

## Paso 4: Cree una entrada de acceso para la función CodePipeline de servicio

En este paso, creará una entrada de acceso en su clúster que agregará el rol de CodePipeline servicio que actualizó en el paso 3, junto con una política de acceso administrado.

1. Abra la consola de EKS y navegue hasta el clúster.
2. Elija la pestaña Acceso.
3. En Entradas de acceso de IAM, elija Crear entrada de acceso.
4. En ARN de entidad principal de IAM, introduzca el rol que acaba de actualizar para la acción, como **`cp-service-role`**. Elija Siguiente.
5. En la página Paso 2: agregación de una política de acceso, en Nombre de la política, elija la política administrada para el acceso, como `AmazonEKSClusterAdminPolicy`. Elija Add Policy (Agregar política). Elija Siguiente.

> **Note**
>
> Esta es la política que usa la CodePipeline acción para comunicarse con Kubernetes. Como práctica recomendada, para limitar los permisos de la política con privilegio mínimo, en lugar de la política administrativa, asocie una política personalizada.

6. En la página Revisar, elija Crear.

## Paso 5: Crea un repositorio de código fuente y agrega los archivos de configuración del gráfico de `Helm`

En este paso, se crea un archivo de configuración adecuado para la acción (archivo de manifiesto de Kubernetes o gráfico de Helm) y luego se almacena en el repositorio de origen. Utilice el archivo de configuración adecuado. Para obtener más información, consulte https://kubernetes.io/docs/reference/kubectl/quick-reference/ o https://helm.sh/docs/topics/charts/.

- Para Kubernetes, utilice un archivo de manifiesto.
- Para Helm, utilice un gráfico de Helm.

1. Crea o usa un GitHub repositorio existente.
2. Crea una nueva estructura en tu repositorio para tus archivos de gráficos de Helm, como se muestra en el siguiente ejemplo.

```
mychart
|-- Chart.yaml
|-- charts
|-- templates
|   |-- NOTES.txt
|   |-- _helpers.tpl
|   |-- deployment.yaml
|   |-- ingress.yaml
|   `-- service.yaml
`-- values.yaml
```

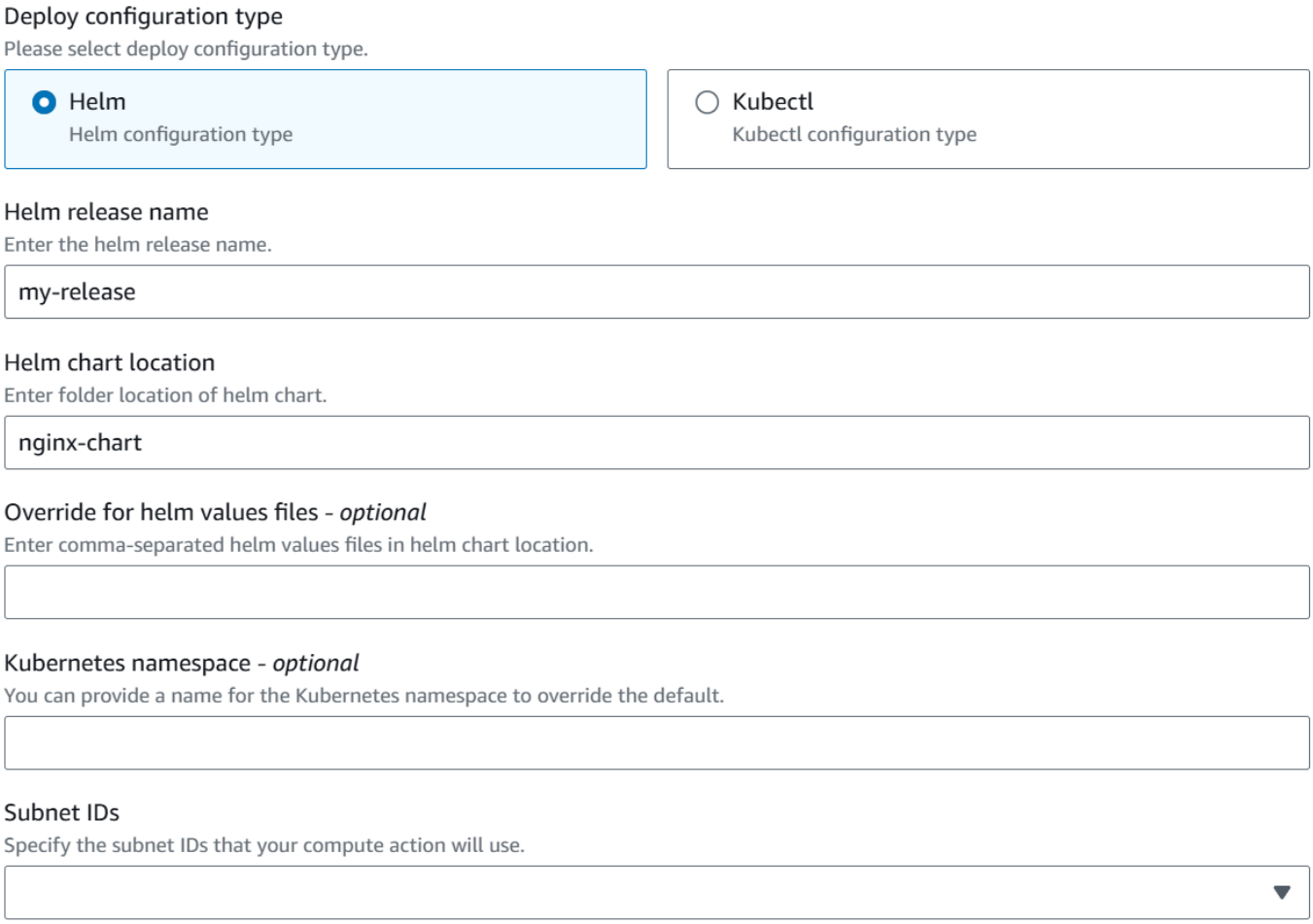
3. Agregue el archivo en el nivel raíz del repositorio.

# Paso 6: creación de una canalización

Usa el CodePipeline asistente para crear las etapas de tu canalización y conectar tu repositorio de origen.

Para crear la canalización

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyEKSPipeline`**.
5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.
6. En Rol de servicio, seleccione el rol de servicio que actualizó en el paso 3.
7. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
8. En la página Paso 3: Añadir fase de código fuente, en el caso del proveedor de código fuente, elige crear una conexión con tu GitHub repositorio.
9. En la página Paso 4: agregación de la etapa de compilación, seleccione Omitir.
10. En la página Paso 5: agregación de la etapa de implementación, elija Amazon EKS.

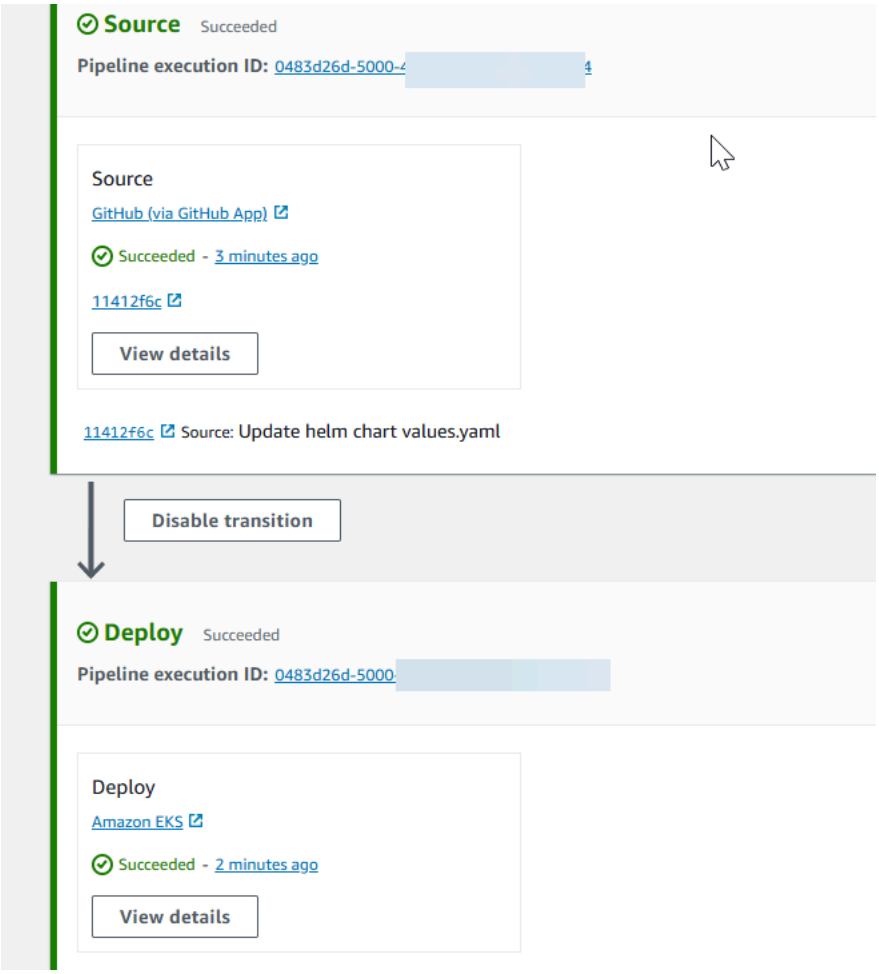



a. En Tipo de configuración de implementación, elija Helm.

b. En Ubicación del gráfico de Helm, introduzca el nombre de la versión, como `my-release`. Para ver la ubicación del gráfico de Helm, introduzca la ruta de los archivos del gráfico de Helm, por ejemplo`mychart`.

c. Elija Siguiente.

11. En la página Step 6: Review, revise la configuración de la canalización y elija Create pipeline para crear la canalización.




12. Cuando la canalización se haya ejecutado correctamente, elija Ver detalles para ver los registros de la acción y su salida.

# Tutorial: creación de una canalización que ejecute comandos mediante computación (tipo V2)

En este tutorial, va a configurar una canalización que ejecutará constantemente comandos de compilación mediante la acción de Comandos en una etapa de compilación. Para obtener más información acerca de la acción de Comandos, consulte Referencia de la acción de Comandos.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará para los artefactos un depósito de artefactos de S3 proporcionado CodePipeline por el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una

cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

## Requisitos previos

Debe disponer de lo siguiente:

- UN REPOSITORIO. GitHub Puedes usar el GitHub repositorio en el que creasteTutorial: Utilice un clon completo con una fuente de GitHub canalización.

## Paso 1: Crea los archivos fuente y envíalos a tu GitHub repositorio

En esta sección, debe crear e insertar los archivos de fuente al repositorio que utiliza la canalización para la etapa de origen. Para este ejemplo, se produce y se inserta lo siguiente:

- Un archivo `README.txt`.

Para crear archivos de fuente

1. Cree un archivo con el siguiente texto:

   ```
   Sample readme file
   ```

2. Guarde el archivo como `README.txt`.

Para subir archivos a tu GitHub repositorio

1. Envíe o cargue los archivos en el repositorio de . Estos archivos son el artefacto de código fuente creado por el asistente Create Pipeline (Crear canalización) para la acción de implementación en AWS CodePipeline. Sus archivos deberían ser parecidos a estos en su directorio local:

   ```
   README.txt
   ```

2. Para utilizar la línea de comandos de Git desde un repositorio clonado en el equipo local:

   a. Ejecute el siguiente comando para preparar todos los archivos de una vez:

```
git add -A
```

b. Ejecute el siguiente comando para confirmar los archivos con un mensaje de confirmación.

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

c. Ejecute el siguiente comando para enviar los archivos de su repositorio local al repositorio de :

```
git push
```

## Paso 2: Crear la canalización

En esta sección, debe crear una canalización con las siguientes acciones:

- Una fase de código fuente con una acción GitHub (a través de la GitHub aplicación) para el repositorio en el que se almacenan los archivos de origen.
- Una etapa de compilación con la acción de Comandos.

Para crear una canalización con el asistente

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyCommandsPipeline`**.
5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.
6. En Función de servicio, seleccione Nueva función de servicio CodePipeline para poder crear una función de servicio en IAM.

> **Note**
>
> Si utiliza un rol de servicio existente, para utilizar la acción de Comandos tendrá que agregar los siguientes permisos para el rol de servicio. Reduzca los permisos al nivel de recursos de la canalización a través de los permisos basados en recursos de la declaración de la política del rol de servicio. Para obtener más información, consulte la política de ejemplo en Permisos para las políticas de roles de servicio.
>
> - registros: CreateLogGroup
> - registros: CreateLogStream
> - registros: PutLogEvents

7. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
8. En la página Paso 3: agregar la etapa de origen, agregue una etapa de origen:

   a. En el proveedor de  origen, elige  GitHub (a través de GitHub la aplicación).

   b. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para tu acción GitHub de origen, consultaGitHub conexiones.

   c. En Repository name (Nombre de repositorio), elija el nombre de su repositorio de GitHub.com .

   d. En la Ramificación predeterminada, elige la ramificación que quiere especificar cuando la canalización se inicie manualmente o con un evento de origen que no sea una etiqueta de Git. Si la fuente del cambio no es el desencadenador o si la ejecución de una canalización se inició manualmente, el cambio utilizado será la confirmación de HEAD desde la ramificación predeterminada. Opcionalmente, también puede especificar webhooks con filtrado (desencadenadores). Para obtener más información, consulte Automatización del inicio de las canalizaciones mediante desencadenadores y filtrado.

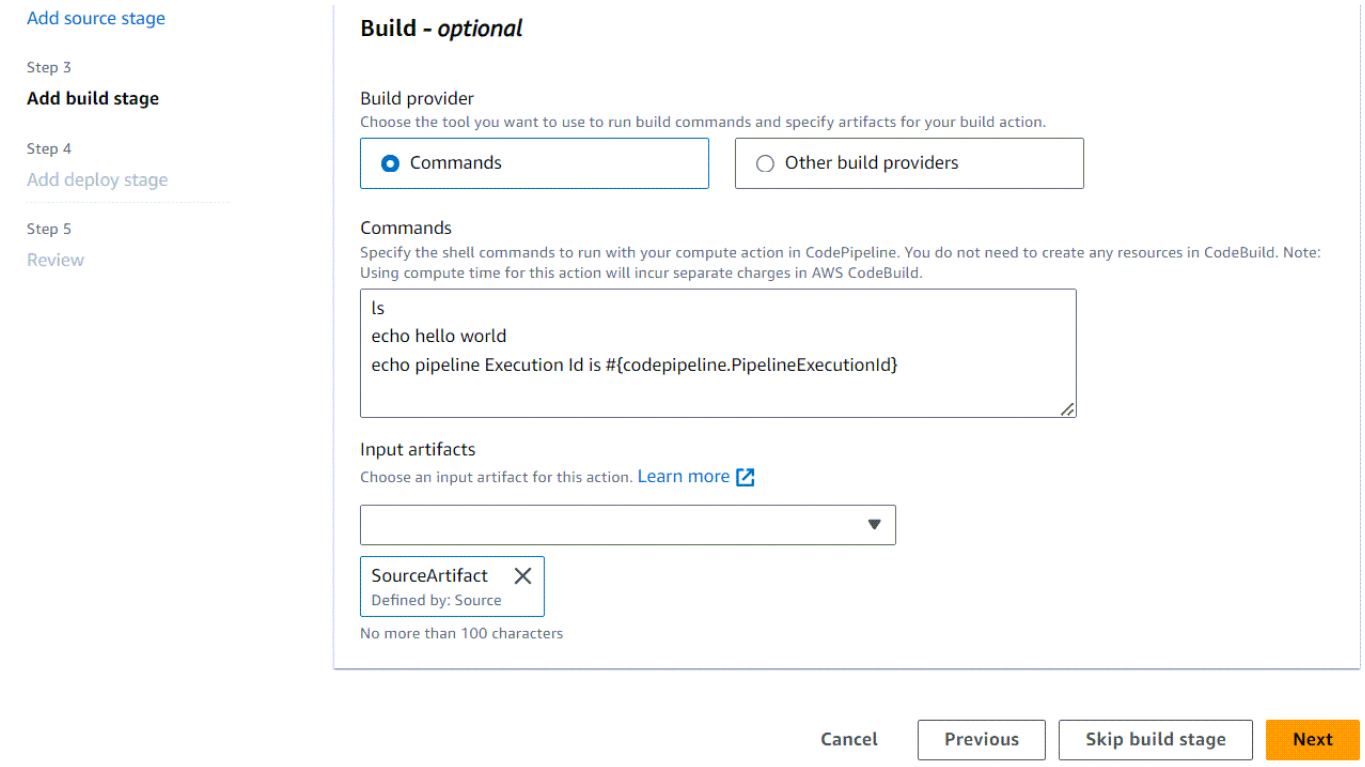
   Elija Siguiente.
9. En Paso 4: agregar la etapa de compilación, seleccione Comandos.

> **Note**
>
> Si ejecuta la acción de Comandos, se le cobrarán cargos por separado en AWS CodeBuild.

Ejecute los comandos siguientes:

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

Elija Siguiente.




10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

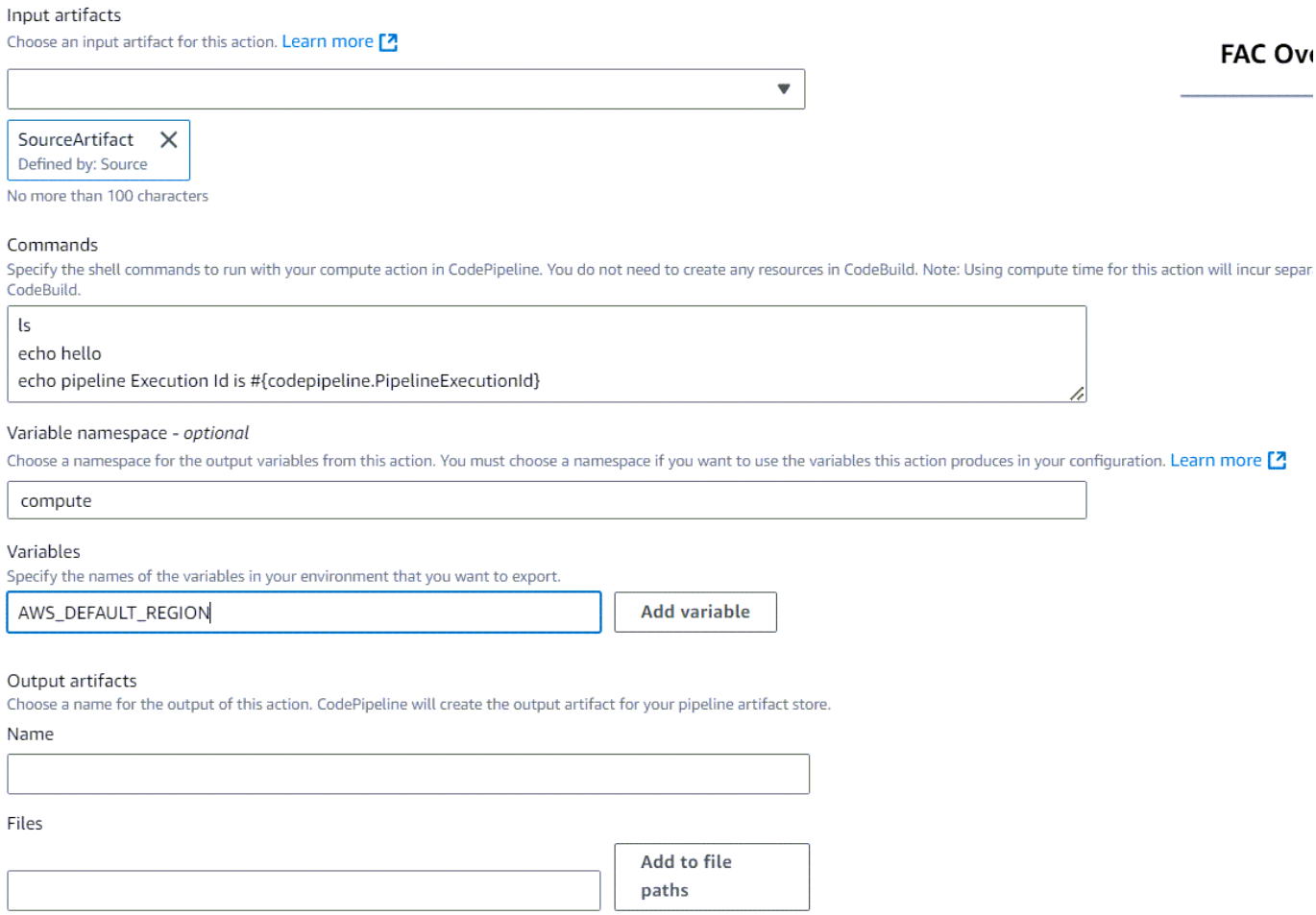
    Elija Siguiente.

11. En Paso 6: agregación de la etapa de implementación, elija Omitir la etapa de implementación y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

Elija Siguiente.

12. En el  paso 7: Revisa, revisa la información y, a continuación, selecciona  Crear canalización.
13. Como último paso para crear la acción, agregue una variable de entorno a la acción que dé como resultado una variable de salida para la acción. En la acción de Comandos, elija Editar. En la pantalla Editar, especifique un espacio de nombres de variables para la acción introduciendo `compute` en el campo Espacio de nombres de variables.

    Agregue la variable CodeBuild de salida y`AWS_Default_Region`, a continuación, elija  Agregar variable.



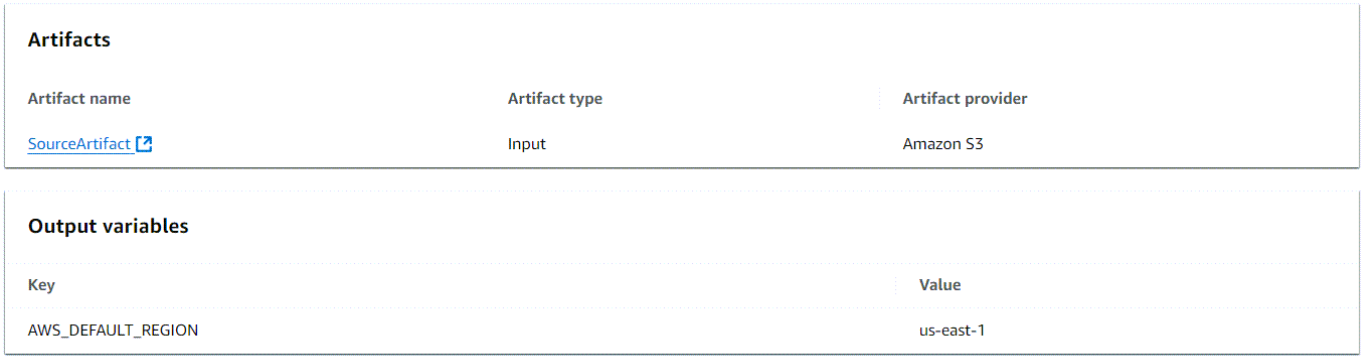
# Paso 3: ejecución de una canalización y verificación de los comandos de compilación

Publique algún cambio para ejecutar su canalización. Compruebe que los comandos de compilación se hayan ejecutado; para ello, consulte el historial de ejecución, los registros de compilación y las variables de salida.

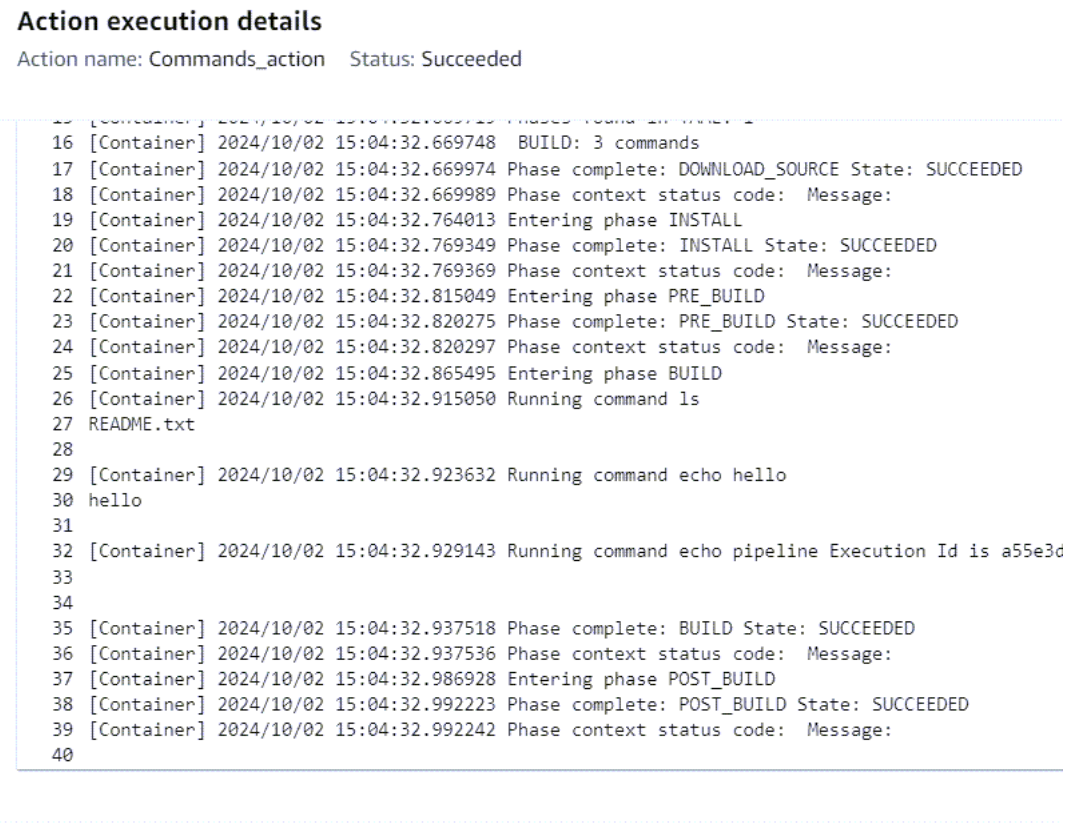
Para ver los registros de acciones y las variables de salida

1. Después de que la canalización se ejecute correctamente, podrá ver los registros y el resultado de la acción.
2. Para ver las variables de salida de la acción, seleccione Historial y, a continuación, seleccione Línea temporal.

   Vea la variable de salida que se agregó a la acción. El resultado de la acción de Comandos muestra la variable de salida resuelta en la región de la acción.




3. Para ver los registros de la acción, seleccione Ver detalles de la acción de Comandos realizada correctamente. Vea los registros de la acción de Comandos.




# Tutorial: Usar etiquetas de Git para iniciar la canalización

En este tutorial, crearás una canalización que se conecte a tu GitHub repositorio donde la acción de origen esté configurada para el tipo de activación de las etiquetas de Git. Cuando se crea una etiqueta Git en una confirmación, se inicia su canalización. En este ejemplo, se muestra cómo crear una canalización que permita filtrar las etiquetas en función de la sintaxis del nombre de la etiqueta. Para obtener más información acerca de filtros con patrones de glob, consulte Trabajar con patrones glob en la sintaxis.

> **Important**
>
> Como parte de la creación de una canalización, para los artefactos se utilizará un depósito de artefactos de S3 proporcionado CodePipeline por el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

Este tutorial se conecta a GitHub través del tipo de acción. `CodeStarSourceConnection`

> **Note**
>
> Esta característica no está disponible en las regiones de Asia-Pacífico (Hong Kong), África (Ciudad del Cabo), Medio Oriente (Baréin), Europa (Zúrich). Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline. Para ver consideraciones sobre esta acción en la región de Europa (Milán), consulte la nota que aparece en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

Temas


- Requisitos previos
- Paso 1: Abre CloudShell y clona tu repositorio
- Paso 2: Crear una canalización para activarla en las etiquetas de Git
- Paso 3: Etiquetar las confirmaciones para publicarlas
- Paso 4: Publicar los cambios y ver los registros


## Requisitos previos

Antes de empezar, debe hacer lo siguiente:

- Crea un GitHub repositorio con tu GitHub cuenta.
- Ten tus GitHub credenciales listas. Cuando utilice el Consola de administración de AWS para configurar una conexión, se le pedirá que inicie sesión con sus GitHub credenciales.

## Paso 1: Abre CloudShell y clona tu repositorio

Puede usar una interfaz de línea de comandos para clonar su repositorio, realizar confirmaciones y añadir etiquetas. Este tutorial lanza una CloudShell instancia para la interfaz de línea de comandos.

1. Inicie sesión en Consola de administración de AWS.
2. En la barra de navegación superior, selecciona el AWS icono. Se muestra la página principal de la Consola de administración de AWS .

3. En la barra de navegación superior, selecciona el AWS CloudShell icono. CloudShell se abre. Espere mientras se crea el CloudShell entorno.

> **Note**
>
> Si no ve el CloudShell icono, asegúrese de que se encuentra en una región compatible con CloudShell. En este tutorial se da por sentado que está en la región de Oeste de EE. UU. (Oregón).

4. En GitHub, navega hasta tu repositorio. Elija Código, y a continuación, elija HTTPS. Copie la ruta. La dirección para clonar el repositorio Git se copia en el portapapeles.
5. Ejecute el siguiente comando para clonar el repositorio.

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

6. Introduzca su GitHub cuenta `Username` y `Password` cuando se le solicite. Para la entrada `Password`, debe usar un token creado por el usuario en lugar de la contraseña de su cuenta.

## Paso 2: Crear una canalización para activarla en las etiquetas de Git

En esta sección, debe crear una canalización con las siguientes acciones:

- Una fase fuente con una conexión a tu GitHub repositorio y a tu acción.
- Una etapa de compilación con una acción de AWS CodeBuild compilación.

Para crear una canalización con el asistente

1. Inicia sesión en la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página Bienvenido, Introducción o en la página Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyGitHubTagsPipeline`**.
5. En Tipo de canalización, mantenga la selección predeterminada en V2. Los tipos de canalización difieren en características y precio. Para obtener más información, consulte Tipos de canalización.

6. En Service role (Rol de servicio), elija New service role (Nuevo rol de servicio).

> **Note**
>
> Si, en cambio, optas por usar tu rol de CodePipeline servicio actual, asegúrate de haber agregado el permiso de `codestar-connections:UseConnection` IAM a tu política de roles de servicio. Para obtener instrucciones sobre la función CodePipeline de servicio, consulta Añadir permisos a la función CodePipeline de servicio.

7. Para Configuración avanzada deje los valores predeterminados. En Artifact store (Almacén de artefactos), elija Default location (Ubicación predeterminada) para utilizar el almacén de artefactos predeterminado, como el bucket de artefacto de Amazon S3 que se estableció como predeterminado, para la canalización en la región que seleccionó para esta.

> **Note**
>
> Este no es el bucket de origen para su código fuente. Este es el almacén de artefactos de la canalización. Cada canalización debe tener su propio almacén de artefactos independiente, como un bucket de S3.

Elija Siguiente.

8. En la página Paso 3: agregar la etapa de origen, agregue una etapa de origen:

   a. En Proveedor de origen, elija GitHub (a través de GitHub la aplicación).

   b. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para tu acción GitHub de origen, consultaGitHub conexiones.

   c. En Repository name (Nombre de repositorio), elija el nombre de su repositorio de GitHub.

   d. En la Ramificación predeterminada, elige la ramificación que quiere especificar cuando la canalización se inicie manualmente o con un evento de origen que no sea una etiqueta de Git. Si la fuente del cambio no es el desencadenador o si la ejecución de una canalización se inició manualmente, el cambio utilizado será la confirmación de HEAD desde la ramificación predeterminada.

   e. En Eventos de webhook, en Tipo de filtro, elija Etiquetas.

      En el campo Etiquetas o patrones, especifique `release*`.

> **Important**
>
> Las canalizaciones que comiencen con un tipo de desencadenador de etiquetas Git se configurarán para los eventos de WebHookV2 y no utilizarán el evento Webhook (detección de cambios en todos los eventos push) para iniciar la canalización.

Elija Siguiente.

9. En Add build stage (Añadir etapa de compilación), añada una etapa de compilación:

   a. En Build provider (Proveedor de compilación), elija AWS CodeBuild. En el campo Region (Región) conserve el valor predeterminado de la región de la canalización.

   b. Elija Crear proyecto.

   c. En Project name (Nombre de proyecto), escriba un nombre para este proyecto de compilación.

   d. En Environment image (Imagen de entorno), elija Managed image (Imagen administrada). En Operating system (Sistema operativo), elija Ubuntu.

   e. En Runtime, elija Standard (Estándar). En  Image, elija  aws/codebuild  /standard:5.0.

   f. En Service role (Rol de servicio), elija New service role (Nuevo rol de servicio).

   > **Note**
   >
   > Anote el nombre de su rol de servicio. CodeBuild Necesitará el nombre del rol para el paso final de este tutorial.

   g. En Buildspec, para Build specifications (Especificaciones de la compilación), elija Insert build commands (Insertar comandos de compilación). Elija Cambiar a editor y pegue lo siguiente en Comandos de compilación:

```
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. Elige Continuar hasta CodePipeline. Esto vuelve a la CodePipeline consola y crea un CodeBuild proyecto que usa tus comandos de compilación para la configuración. El proyecto de compilación usa un rol de servicio para administrar Servicio de AWS los permisos. Es posible que este paso tarde un par de minutos.

i. Elija Siguiente.

10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

Elija Siguiente.

11. En la página Paso 6: Agregar la fase de implementación, elija Omitir la etapa de implementación y, a continuación, acepte el mensaje de advertencia seleccionando nuevamente Omitir. Elija Siguiente.
12. En el paso 7: revisar, selecciona Crear canalización.

## Paso 3: Etiquetar las confirmaciones para publicarlas

Tras crear tu canalización y especificar las etiquetas de Git, puedes etiquetar las confirmaciones en tu GitHub repositorio. En estos pasos, etiquetará una confirmación con la etiqueta `release-1`. Cada confirmación de un repositorio de Git debe tener una etiqueta Git única. Si elige la confirmación y la etiqueta, podrá incorporar los cambios de distintas ramificaciones a la implementación de su canalización. Ten en cuenta que el nombre de la etiqueta release no se aplica al concepto de lanzamiento en GitHub.

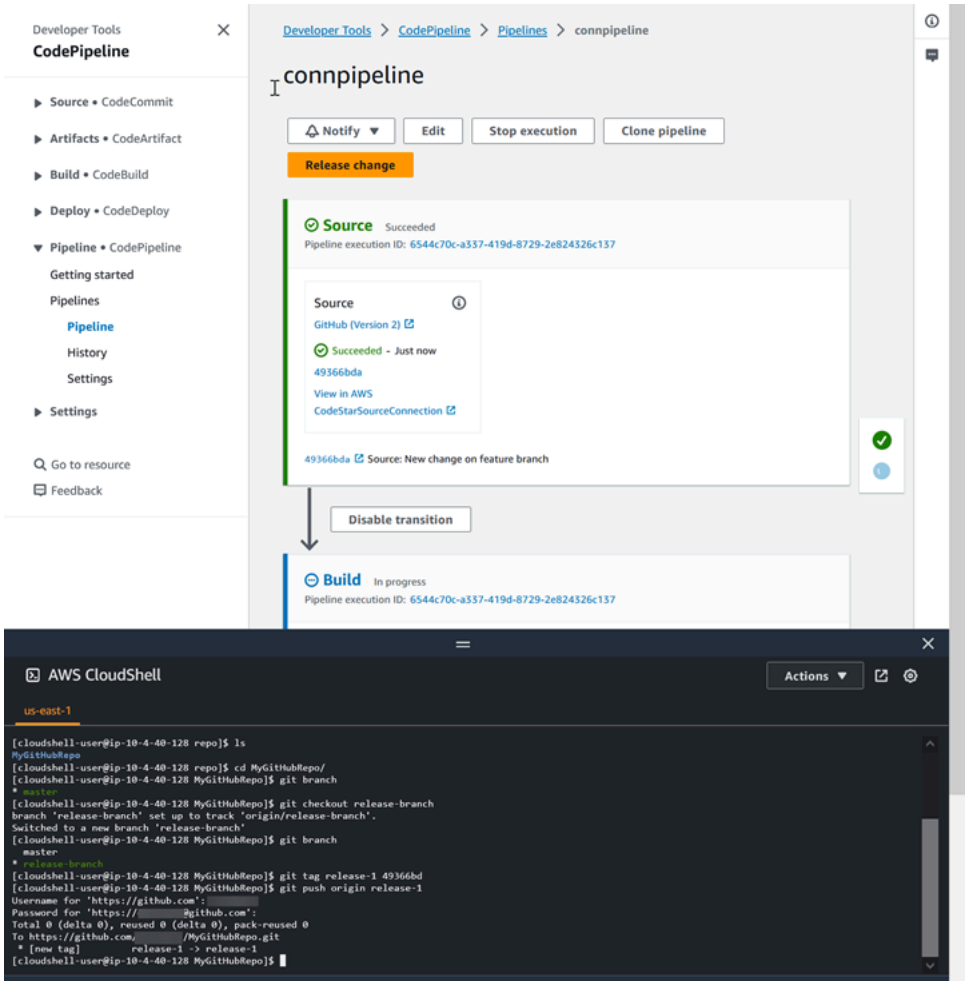
1. Haga referencia a los ID de confirmación copiados que desea etiquetar. Para ver las confirmaciones en cada rama, en la CloudShell terminal, introduce el siguiente comando para capturar los ID de confirmación que quieres etiquetar:

   ```
   git log
   ```

2. En la CloudShell terminal, introduce el comando para etiquetar tu confirmación y enviarla al origen. Después de etiquetar la confirmación, use el comando de inclusión git para enviar la etiqueta al origen. En el siguiente ejemplo, introduzca el siguiente comando para usar la etiqueta `release-1` en la segunda confirmación con ID `49366bd`. Esta etiqueta se filtrará mediante el filtro de etiquetas `release*` de la canalización e iniciará la canalización.

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

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




## Paso 4: Publicar los cambios y ver los registros

1. Después de que la canalización se ejecute correctamente, en la etapa de implementación, elija Ver registro.

   En  Registros, consulta el resultado de la CodeBuild compilación. Los comandos muestran el valor de la variable introducida.

2. En la página Historial, consulte la columna Desencadenadores. Consulta el tipo de desencadenador GitTag : release-1.

# Tutorial: filtrado de solicitudes de extracción por nombres de ramificación para iniciar una canalización (tipo V2)

En este tutorial, crearás una canalización que se conecte a tu GitHub.com repositorio, donde la acción de origen está configurada para iniciar tu canalización con una configuración de activación que filtra las solicitudes de extracción. Cuando se produce un evento de solicitud de extracción específico para una ramificación específica, se inicia la canalización. En este ejemplo, se muestra cómo crear una canalización que permita filtrar por nombres de ramificación. Para obtener más información acerca de cómo trabajar con desencadenadores, consulte Agregación de filtros para los tipos de eventos de solicitud de inserción y extracción (CLI). Para obtener más información acerca del filtrado con patrones regex en formato glob, consulte Trabajar con patrones glob en la sintaxis.

> **Important**
>
> Como parte de la creación de una canalización, para los artefactos se utilizará un depósito de artefactos de S3 proporcionado CodePipeline por el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

Este tutorial se conecta a GitHub.com través del tipo de acción. `CodeStarSourceConnection`

Temas


- Requisitos previos
- Paso 1: creación de una canalización que iniciar con una solicitud de extracción de ramificaciones específicas
- Paso 2: Crea y fusiona una pull request GitHub.com para iniciar la ejecución de tu canalización


## Requisitos previos

Antes de empezar, debe hacer lo siguiente:

- Crea un GitHub.com repositorio con tu GitHub.com cuenta.
- Ten tus GitHub credenciales listas. Cuando utilice el Consola de administración de AWS para configurar una conexión, se le pedirá que inicie sesión con sus GitHub credenciales.

# Paso 1: creación de una canalización que iniciar con una solicitud de extracción de ramificaciones específicas

En esta sección, debe crear una canalización con las siguientes acciones:

- Una etapa de origen con una conexión a tu GitHub.com repositorio y a tu acción.
- Una etapa de compilación con una acción de AWS CodeBuild compilación.

Para crear una canalización con el asistente

1. Inicia sesión en la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página Bienvenido, Introducción o en la página Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyFilterBranchesPipeline`**.
5. En Tipo de canalización, mantenga la selección predeterminada en V2. Los tipos de canalización difieren en características y precio. Para obtener más información, consulte Tipos de canalización.
6. En Service role (Rol de servicio), elija New service role (Nuevo rol de servicio).

   > **Note**
   >
   > Si, en cambio, optas por usar tu rol de CodePipeline servicio actual, asegúrate de haber agregado el permiso de `codeconnections:UseConnection` IAM a tu política de roles de servicio. Para obtener instrucciones sobre la función CodePipeline de servicio, consulta Añadir permisos a la función CodePipeline de servicio.

7. Para Configuración avanzada deje los valores predeterminados. En Artifact store (Almacén de artefactos), elija Default location (Ubicación predeterminada) para utilizar el almacén de artefactos predeterminado, como el bucket de artefacto de Amazon S3 que se estableció como predeterminado, para la canalización en la región que seleccionó para esta.

> **Note**
>
> Este no es el bucket de origen para su código fuente. Este es el almacén de artefactos de la canalización. Cada canalización debe tener su propio almacén de artefactos independiente, como un bucket de S3.

Elija Siguiente.

8. En la página Paso 3: agregar la etapa de origen, agregue una etapa de origen:

   a. En Proveedor de origen, elija GitHub (a través de GitHub la aplicación).

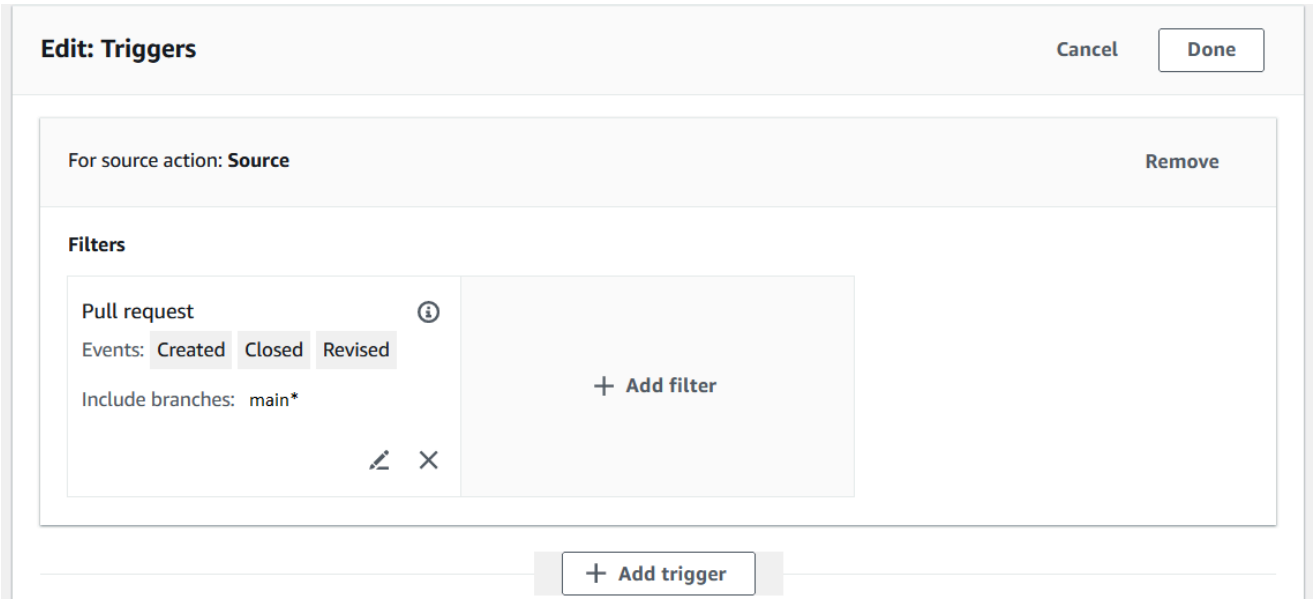
   b. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para tu acción GitHub de origen, consultaGitHub conexiones.

   c. En Repository name (Nombre de repositorio), elija el nombre de su repositorio de GitHub.com.

   d. En Tipo de desencadenador, seleccione Especificar filtro.

      En Tipo de evento, seleccione Solicitud de extracción. Seleccione todos los eventos de la solicitud de extracción para que el evento se produzca para las solicitudes de extracción creadas, actualizadas o cerradas.

      En Ramificaciones, en el campo Incluir, introduzca `main*`.



> **Important**
>
> Las canalizaciones que se inicien con este tipo de desencadenador se configurarán para los eventos de WebhookV2 y no utilizarán el evento de Webhook (detección de cambios en todos los eventos de inserción) para iniciar la canalización.

Elija Siguiente.

9. En Paso 4: agregación de la etapa de compilación, en Proveedor de compilación, elija AWS CodeBuild. En el campo Region (Región) conserve el valor predeterminado de la región de la canalización. Elija o cree el proyecto de compilación como se indica en Tutorial: Usar etiquetas de Git para iniciar la canalización. Esta acción solo se usará en este tutorial como la segunda etapa necesaria para crear la canalización.
10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

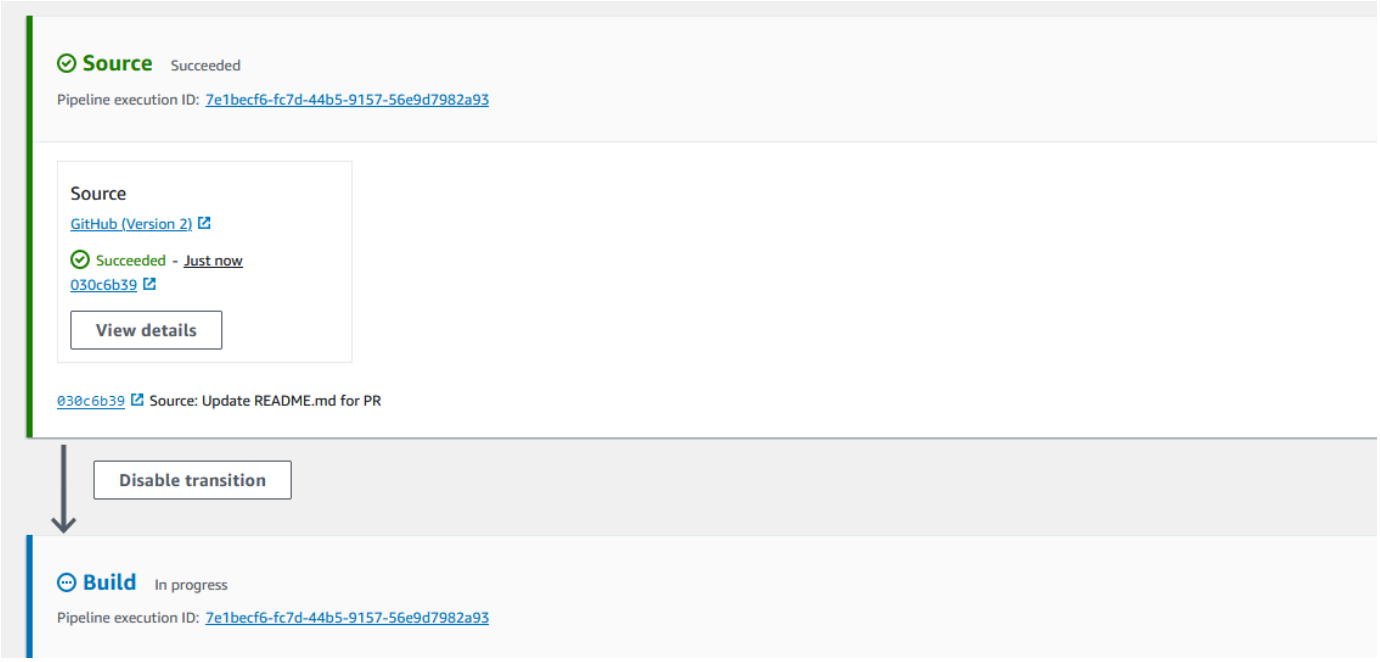
    Elija Siguiente.
11. En la página Paso 6: Agregar fase de despliegue, seleccione Omitir fase de despliegue y, a continuación, acepte el mensaje de advertencia seleccionando de nuevo Omitir. Elija Siguiente.
12. En el paso 7: revisar, selecciona Crear canalización.

## Paso 2: Crea y fusiona una pull request GitHub.com para iniciar la ejecución de tu canalización

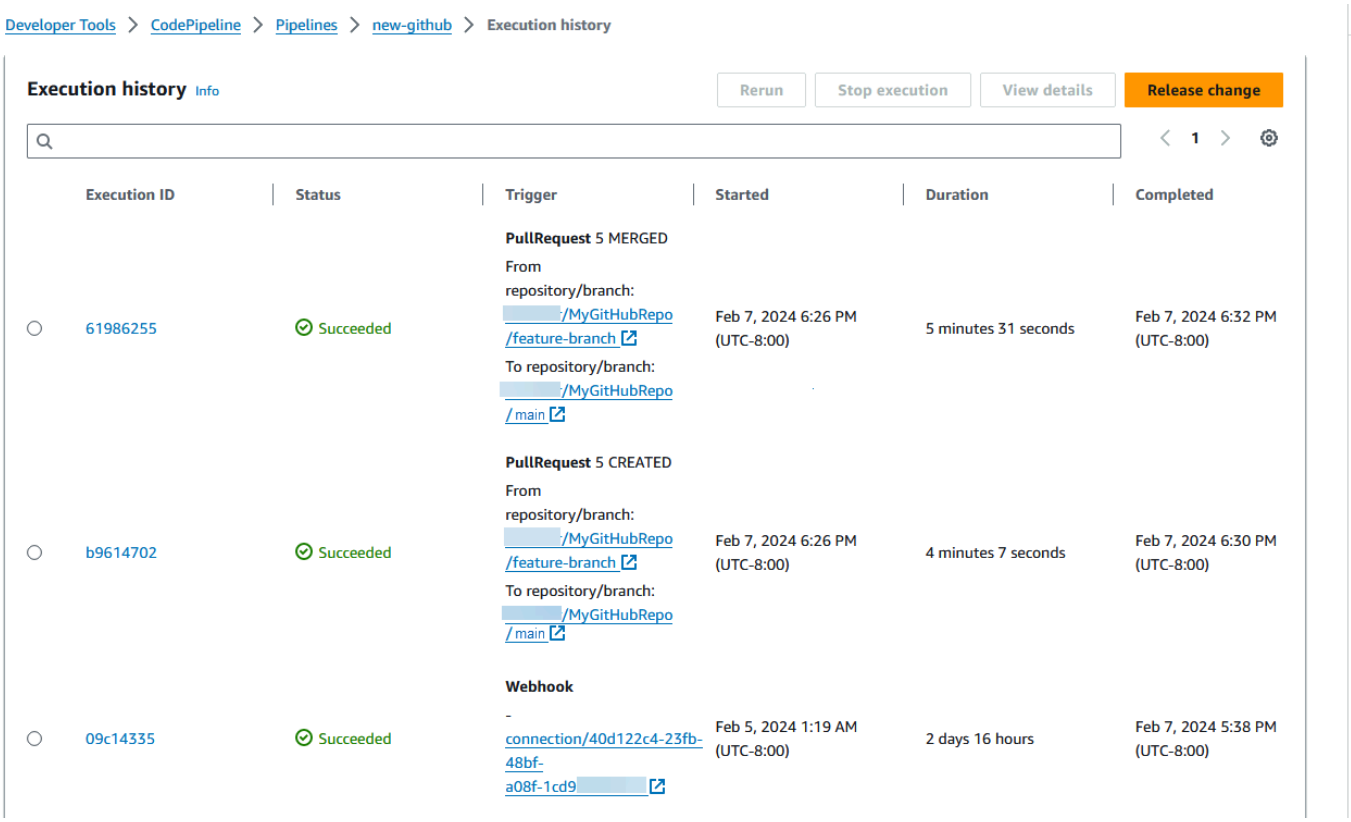
En esta sección, va a crear y combinar una solicitud de extracción. Esto inicia la canalización, con una ejecución para la solicitud de extracción abierta y una ejecución para la solicitud de extracción cerrada.

Para crear una solicitud de extracción e iniciar la canalización

1. En GitHub.com primer lugar, crea una solicitud de extracción realizando un cambio README.md en la rama de una función y enviando una solicitud de extracción a la `main` rama. Confirme el cambio con un mensaje como `Update README.md for PR`.
2. La canalización comienza con la revisión de la fuente que muestra el mensaje de origen de la solicitud de extracción como actualización README.md para PR.




3. Elija Historial. En el historial de ejecución de canalizaciones, consulte los eventos de estado de las solicitudes de extracción CREATED y MERGED que iniciaron las ejecuciones de canalización.



# Tutorial: Uso de variables a nivel de canalización

En este tutorial, crearás una canalización en la que añadirás una variable a nivel de canalización y ejecutarás una acción de CodeBuild compilación que generará el valor de la variable.

> **Important**
>
> Como parte de la creación de una canalización, para los artefactos se utilizará un depósito de artefactos S3 proporcionado CodePipeline por el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

Temas


- Requisitos previos
- Paso 1: Crear la canalización y compilar el proyecto
- Paso 2: Publicar los cambios y ver los registros


## Requisitos previos

Antes de empezar, debe hacer lo siguiente:

- Crea un repositorio. CodeCommit
- Añada un archivo .txt al repositorio.

## Paso 1: Crear la canalización y compilar el proyecto

En esta sección, debe crear una canalización con las siguientes acciones:

- Una etapa de origen con una conexión a tu CodeCommit repositorio.
- Una etapa de compilación con una acción de AWS CodeBuild compilación.

Para crear una canalización con el asistente

1. Inicia sesión en la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

2. En la página Bienvenido, Introducción o en la página Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyVariablesPipeline`**.
5. En Tipo de canalización, mantenga la selección predeterminada en V2. Los tipos de canalización difieren en características y precio. Para obtener más información, consulte Tipos de canalización.
6. En Service role (Rol de servicio), elija New service role (Nuevo rol de servicio).

   > **Note**
   >
   > Si, en cambio, optas por usar tu rol de CodePipeline servicio actual, asegúrate de haber agregado el permiso de `codeconnections:UseConnection` IAM a tu política de roles de servicio. Para obtener instrucciones sobre la función CodePipeline de servicio, consulta Añadir permisos a la función CodePipeline de servicio.

7. En Variables, seleccione Añadir variable. En Name (Nombre), escriba `timeout`. En Predeterminado, escriba 1000. En la descripción, introduzca la siguiente descripción: **`Timeout`**.

   Esto creará una variable en la que podrá declarar el valor cuando comience la ejecución de la canalización. Los nombres de las variables deben coincidir con `[A-Za-z0-9@\-_]+` y pueden ser cualquier cosa excepto una cadena vacía.
8. Para Configuración avanzada deje los valores predeterminados. En Artifact store (Almacén de artefactos), elija Default location (Ubicación predeterminada) para utilizar el almacén de artefactos predeterminado, como el bucket de artefacto de Amazon S3 que se estableció como predeterminado, para la canalización en la región que seleccionó para esta.

   > **Note**
   >
   > Este no es el bucket de origen para su código fuente. Este es el almacén de artefactos de la canalización. Cada canalización debe tener su propio almacén de artefactos independiente, como un bucket de S3.

   Elija Siguiente.

9. En la página Paso 3: agregar la etapa de origen, agregue una etapa de origen:

   a. En Source provider (Proveedor de código fuente), elija AWS CodeCommit.

   b. En Nombre del repositorio y Nombre de la sucursal, elige tu repositorio y sucursal.

   Elija Siguiente.

10. En Paso 4: agregación de la etapa de compilación, agregue una etapa de compilación:

    a. En Build provider (Proveedor de compilación), elija AWS CodeBuild. En el campo Region (Región) conserve el valor predeterminado de la región de la canalización.

    b. Elija Crear proyecto.

    c. En Project name (Nombre de proyecto), escriba un nombre para este proyecto de compilación.

    d. En Environment image (Imagen de entorno), elija Managed image (Imagen administrada). En Operating system (Sistema operativo), elija Ubuntu.

    e. En Runtime, elija Standard (Estándar). En Imagen, elige aws/codebuild /standard:5.0.

    f. En Service role (Rol de servicio), elija New service role (Nuevo rol de servicio).

    > **Note**
    >
    > Anote el nombre de su rol de servicio. CodeBuild Necesitará el nombre del rol para el paso final de este tutorial.

    g. En Buildspec, para Build specifications (Especificaciones de la compilación), elija Insert build commands (Insertar comandos de compilación). Elija Cambiar a editor y pegue lo siguiente en Comandos de compilación: En la especificación de compilación, se utilizará la variable de cliente `$CUSTOM_VAR1` para generar la variable de canalización en el registro de compilación. En el siguiente paso, creará la variable de salida `$CUSTOM_VAR1` como variable de entorno.

```
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. Elige Continuar hasta CodePipeline. Esto vuelve a la CodePipeline consola y crea un CodeBuild proyecto que usa tus comandos de compilación para la configuración. El proyecto de compilación usa un rol de servicio para administrar Servicio de AWS los permisos. Es posible que este paso tarde un par de minutos.

i. En Variables de entorno (opcional), para crear una variable de entorno como variable de entrada para la acción de creación que resolverá la variable a nivel de canalización, elija Añadir variable de entorno. Esto creará la variable especificada en la especificación de

compilación como `$CUSTOM_VAR1`. En Name (Nombre), escriba `CUSTOM_VAR1`. En Valor, escriba `#{variables.timeout}`. En Tipo, elija `Plaintext`.

El `#{variables.timeout}` valor de la variable de entorno se basa en el espacio de nombres de la variable a nivel de canalización `variables` y en la variable a nivel de canalización `timeout` creada para la canalización en el paso 7.

j. Elija Siguiente.

11. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.

12. En la página Paso 6: Añadir fase de despliegue, selecciona Omitir fase de despliegue y, a continuación, vuelve a seleccionar Omitir fase de despliegue para aceptar el mensaje de advertencia. Elija Siguiente.

13. En el paso 7: revisar, selecciona Crear canalización.

## Paso 2: Publicar los cambios y ver los registros

1. Después de que la canalización se ejecute correctamente, en la etapa de implementación, elija Ver detalles.

   En la página de detalles, elija la pestaña Registros. Ve el resultado de la CodeBuild compilación. Los comandos muestran el valor de la variable introducida.

2. En el panel de navegación de la izquierda, elija Historial.

   Elija la ejecución reciente y, a continuación, elija la pestaña Variables. Vea el valor resuelto de la variable de canalización.

# Tutorial: Crear una canalización simple (bucket de S3)

La forma más sencilla de crear una canalización es utilizar el asistente de creación de canalizaciones de la AWS CodePipeline consola.

En este tutorial, crearás una canalización en dos etapas que utilice un bucket de origen de S3 versionado y CodeDeploy lanzarás una aplicación de muestra.

> **Note**
>
> Si Amazon S3 es el proveedor de origen de la canalización, debe comprimir los archivos de origen en un solo archivo.zip y cargarlo en el bucket de origen. También puede cargar un archivo sin comprimir; sin embargo, se producirá un error en las acciones posteriores que esperan un archivo.zip.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará para los artefactos un depósito de artefactos de S3 proporcionado por CodePipeline el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

Una vez creada esta canalización simple, le agregará otra etapa para después deshabilitar y volver a habilitar la transición entre etapas.

> **Important**
>
> Muchas de las acciones que añades a tu canalización en este procedimiento implican AWS recursos que debes crear antes de crear la canalización. AWS Los recursos para tus acciones de origen siempre deben crearse en la misma AWS región en la que creas tu canalización. Por ejemplo, si creas tu canalización en la región Este de EE. UU. (Ohio), tu CodeCommit repositorio debe estar en la región Este de EE. UU. (Ohio).
> Puedes añadir acciones interregionales al crear tu canalización. AWS Los recursos para las acciones interregionales deben estar en la misma AWS región en la que planeas ejecutar la acción. Para obtener más información, consulte Agregue una acción interregional en CodePipeline.

Antes de comenzar, debe completar los requisitos previos de Empezar con CodePipeline.

Temas

- Paso 1: creación de un bucket de origen de S3 para la aplicación

- Paso 2: Cree instancias Windows de Amazon EC2 e instale el agente CodeDeploy
- Paso 3: Crea una aplicación en CodeDeploy
- Paso 4: Crea tu primera canalización en CodePipeline
- (Opcional) Paso 5: Agregar otra etapa a la canalización
- (Opcional) Paso 6: Deshabilite y habilite las transiciones entre las etapas de CodePipeline
- Paso 7: Limpiar recursos


## Paso 1: creación de un bucket de origen de S3 para la aplicación

Puede almacenar sus aplicaciones o archivos de código fuente en cualquier ubicación con control de versiones. En este tutorial, va a crear un bucket de S3 para los archivos de la aplicación de ejemplo y a habilitar en él el control de versiones. Una vez habilitado el control de versiones, copiará las aplicaciones de ejemplo a ese bucket.

Para crear un bucket de S3

1. Inicia sesión en la consola en Consola de administración de AWS. Abra la consola de S3.
2. Elija Crear bucket.
3. En Bucket Name (Nombre del bucket), escriba el nombre del bucket (por ejemplo: **`awscodepipeline-demobucket-example-date`**).

   > **Note**
   >
   > Como todos los nombres de bucket en Amazon S3 deben ser únicos, elija un nombre que no sea el que aparece en el ejemplo. Puede cambiar el nombre del ejemplo simplemente añadiéndole la fecha. Anote este nombre, ya que lo necesitará durante el resto de este tutorial.

   En Región, elija la región en la que quiere crear la canalización (por ejemplo, Oeste de EE. UU. (Oregón)) y elija Crear bucket.
4. Una vez creado el bucket, aparecerá un banner donde se indicará que la operación se ha realizado correctamente. Elija Go to bucket details (Acceder a los detalles del bucket).
5. En la pestaña Properties (Propiedades), elija Versioning (Control de versiones). Elija Enable versioning (Habilitar control de versiones) y haga clic en Save (Guardar).

Cuando se habilita este control de versiones, Amazon S3 guarda todas las versiones de cada objeto en el bucket.

6. En la pestaña Permissions (Permisos), deje los valores predeterminados. Para obtener más información sobre los permisos de los objetos y los buckets de S3, consulte Especificación de permisos en una política.
7. A continuación, descargue un ejemplo y guárdelo en una carpeta o directorio del equipo local.

   a. Elija una de las siguientes opciones. Elija `SampleApp_Windows.zip` si desea seguir los pasos de este tutorial en instancias de Windows Server.

      - Si desea realizar la implementación en instancias de Amazon Linux utilizando CodeDeploy, descargue la aplicación de muestra aquí: SampleApp _ Linux.zip.
      - Si desea realizar la implementación en instancias de Windows Server utilizando CodeDeploy, descargue la aplicación de muestra aquí: SampleApp _ Windows.zip.

      La aplicación de ejemplo contiene los siguientes archivos para implementarlos con CodeDeploy:

      - `appspec.yml`— El archivo de especificación de la aplicación (AppSpecarchivo) es un archivo con formato YAML que se utiliza CodeDeploy para gestionar una implementación. Para obtener más información sobre el AppSpec archivo, consulte la referencia al CodeDeploy AppSpec archivo en la Guía del AWS CodeDeploy usuario.
      - `index.html`: el archivo de índice contiene la página de inicio de la aplicación de ejemplo implementada.
      - `LICENSE.txt`: el archivo de licencia contiene la información de licencia de la aplicación de ejemplo.
      - Archivos para scripts: la aplicación de ejemplo usa scripts para escribir archivos de texto en una ubicación de la instancia. Se escribe un archivo para cada uno de los varios eventos del ciclo de vida de la CodeDeploy implementación de la siguiente manera:
        - (Solo ejemplos de Linux) Carpeta `scripts`: la carpeta contiene los siguientes scripts de intérprete de comandos para instalar las dependencias e iniciar y detener la aplicación de ejemplo para la implementación automatizada: `install_dependencies`, `start_server` y `stop_server`.
        - (Solo ejemplo de Windows) `before-install.bat`: se trata de un script por lotes para el evento de ciclo de vida de implementación, que se ejecutará para eliminar los

archivos antiguos escritos durante implementaciones anteriores de este ejemplo y crear una ubicación en su instancia en la que escribir los archivos nuevos.

b. Descargue el archivo comprimido (en zip). No descomprima el archivo.

8. En la consola de Amazon S3, para su bucket suba el archivo:

   a. Seleccione Cargar.

   b. Arrastre y suelte el archivo o elija Add files (Agregar archivos) y busque el archivo.

   c. Seleccione Cargar.

## Paso 2: Cree instancias Windows de Amazon EC2 e instale el agente CodeDeploy

> **Note**
>
> Este tutorial contiene ejemplos de los pasos necesarios para crear instancias de Amazon EC2 de Windows. Si desea ver los pasos de ejemplo para crear instancias de Amazon EC2 para Linux, consulte Paso 3: Cree una instancia Linux de Amazon EC2 e instale el agente CodeDeploy. Cuando le pidan el número de instancias que se van a crear, especifique 2 instancias.

En este paso, va a crear instancias de Amazon EC2 de Windows Server en las que implementará una aplicación de ejemplo. Como parte de este proceso, se crea un rol de instancia con políticas que permiten la instalación y la administración del CodeDeploy agente en las instancias. El CodeDeploy agente es un paquete de software que permite usar una instancia en CodeDeploy las implementaciones. También se adjuntan políticas que permiten a la instancia recuperar los archivos que el CodeDeploy agente usa para implementar la aplicación y permitir que SSM administre la instancia.

Para crear un rol de instancia

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de la consola, elija Roles.
3. Elija Crear rol.
4. En Seleccionar el tipo de entidad de confianza, seleccione Servicio de AWS. En Choose a use case (Elija un caso de uso), seleccione EC2 y Next: Permissions (Siguiente: Permisos).

5. Busque y seleccione la política denominada **AmazonEC2RoleforAWSCodeDeploy**.
6. Busque y seleccione la política denominada **AmazonSSMManagedInstanceCore**. Elija Siguiente: Etiquetas.
7. Elija Siguiente: Revisar. Escriba el nombre del rol (por ejemplo, **EC2InstanceRole**).

> **Note**
>
> Anote el nombre del rol para utilizarlo en el siguiente paso. Tendrá que elegir este rol cuando cree la instancia.

Elija Crear rol.

Para lanzar las instancias

1. Abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En la barra de navegación lateral, elija Instancias y seleccione Lanzar instancias en la parte superior de la página.
3. En Nombre y etiquetas, en Nombre, introduzca **MyCodePipelineDemo**. Esto asigna a las instancias una clave de etiqueta **Name** y un valor de etiqueta **MyCodePipelineDemo**. Más adelante, creas una CodeDeploy aplicación que implementa la aplicación de ejemplo en las instancias. CodeDeployselecciona las instancias que se van a implementar en función de las etiquetas.
4. En Imágenes de aplicaciones y sistema operativo (imagen de máquina de Amazon), elija Windows. (Esta AMI se describe como  Microsoft Windows Server 2019 Base  y está etiquetada como «apta para el nivel gratuito» y se puede encontrar en Inicio  rápido).
5. En Tipo de instancia, elija el tipo `t2.micro` apto para la capa gratuita como configuración de hardware de la instancia.
6. En Par de claves (inicio de sesión), seleccione un par de claves o cree uno.

   Puede elegir Continuar sin un par de claves.

> **Note**
>
> A efectos de este tutorial, puede continuar sin utilizar un par de claves. Si desea usar SSH para conectarse a sus instancias, cree o use un par de claves.

7. En Configuración de red, haga lo siguiente:

   En IP Auto-assign pública, asegúrese de que el estado sea Habilitado.

   - Junto a Assign a security group (Asignar un grupo de seguridad), elija Create a new security group (Crear un nuevo grupo de seguridad).
   - En la fila de SSH, en Tipo de origen, elija Mi IP.
   - Elija Añadir grupo de seguridad, HTTP y en Tipo de origen, elija Mi IP.
8. Amplíe Advanced details (Detalles avanzados). En Perfil de instancia de IAM, elija el rol de IAM que creó en el procedimiento anterior (por ejemplo, **`EC2InstanceRole`**).
9. En Resumen, en Número de instancias, especifique 2.
10. Seleccione Iniciar instancia.
11. Elija Ver todas las instancias para cerrar la página de confirmación y volver a la consola.
12. Puede ver el estado del lanzamiento en la página Instances. Al lanzar una instancia, su estado inicial es `pending`. Una vez iniciada la instancia, el estado cambia a `running` y recibe un nombre de DNS público. (Si no aparece la columna DNS público, elija el Show/Hide icono y, a continuación, seleccione DNS público).
13. Puede que transcurran unos minutos hasta que la instancia esté lista para conectarse. Verifique que su instancia ha pasado las comprobaciones de estado. Puede ver esta información en la columna Status Checks (Comprobaciones de estado).

## Paso 3: Crea una aplicación en CodeDeploy

En CodeDeploy, una aplicación es un identificador, en forma de nombre, del código que desea implementar. CodeDeploy usa este nombre para garantizar que se haga referencia a la combinación correcta de revisión, configuración de implementación y grupo de implementación durante una implementación. Seleccione el nombre de la CodeDeploy aplicación que cree en este paso cuando cree la canalización más adelante en este tutorial.

Primero debes crear un rol de servicio CodeDeploy para usarlo. Si ya ha creado un rol de servicio, no necesita crear otro.

Para crear un rol CodeDeploy de servicio

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de la consola, elija Roles.
3. Elija Crear rol.
4. En Seleccionar entidad de confianza, elija Servicio de AWS. En Use case (Caso de uso), elija CodeDeploy. Elija una CodeDeploy de las opciones de la lista. Elija Siguiente. La política administrada `AWSCodeDeployRole` ya está asociada al rol.
5. Elija Siguiente.
6. Especifique un nombre para el rol (por ejemplo, **`CodeDeployRole`**) y elija Create role (Crear rol).

Para crear una aplicación en CodeDeploy

1. Abra la CodeDeploy consola en https://console.aws.amazon.com/codedeploy.
2. Si la página Aplicaciones no aparece, en el AWS CodeDeploy menú, seleccione Aplicaciones.
3. Elija Creación de aplicación.
4. En Application name (Nombre de aplicación), escriba `MyDemoApplication`.
5. En Compute Platform, elige EC2/On-premises.
6. Elija Creación de aplicación.

Para crear un grupo de implementación en CodeDeploy

1. En la página que muestra su aplicación, elija Create deployment group (Crear grupo de implementaciones).
2. En Nombre de grupo de implementación, escriba **`MyDemoDeploymentGroup`**.
3. En Rol de servicio, elija el rol de servicio de que creó anteriormente. Debe usar un rol de servicio que confíe AWS CodeDeploy , como mínimo, en la confianza y los permisos descritos en Crear un rol de servicio para CodeDeploy. Para obtener el ARN del rol de servicio, consulte Obtención del ARN del rol de servicio (consola).
4. En Tipo de implementación, elija In-place.

5. En Environment configuration (Configuración de entorno), elija Amazon EC2 Instances (Instancias de Amazon EC2). Elija Name (Nombre) en el campo Key (Clave) y, en el campo Value (Valor), escriba **`MyCodePipelineDemo`**.

   > **Important**
   >
   > En la clave Nombre, debe elegir el mismo valor que asignó a la instancia EC2 cuando la creó. Si etiquetó la instancia con un término distinto de **`MyCodePipelineDemo`**, asegúrese de usarlo aquí.

6. En Configuración del  agente con AWS Systems Manager, elija  Ahora y programe las actualizaciones. Primero, instale el agente en la instancia. La instancia de Windows ya está configurada con el agente SSM y ahora se actualizará con el CodeDeploy agente.
7. En Configuración de implementación, elija `CodeDeployDefault.OneAtaTime`.
8. En Equilibrador de carga, asegúrese de que no esté seleccionado el cuadro Habilitar equilibrio de carga. No es necesario configurar un balanceador de carga ni elegir un grupo de destino para este ejemplo. Tras anular la selección de la casilla de verificación, no se muestran las opciones del equilibrador de carga.
9. En la sección Avanzado, deje los valores predeterminados.
10. Elija Crear grupo de implementación.

## Paso 4: Crea tu primera canalización en CodePipeline

En esta parte del tutorial se crea la canalización. El ejemplo se ejecuta automáticamente en la canalización.

Para crear un proceso de publicación CodePipeline automatizado

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en  http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyFirstPipeline`**.

> **Note**
>
> Si elige otro nombre para la canalización, utilícelo en lugar de **`MyFirstPipeline`** durante el resto de este tutorial. Después de crear una canalización, no podrá cambiar el nombre. Los nombres de canalizaciones están sujetos a algunas limitaciones. Para obtener más información, consulte Cuotas en AWS CodePipeline.

5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.
6. En Service role (Rol de servicio), realice una de las operaciones siguientes:
   - Seleccione  Nueva función de servicio  para CodePipeline poder crear una nueva función de servicio en IAM.
   - Elija Existing service role (Rol de servicio existente) para utilizar un rol de servicio que ya se ha creado en IAM. En Role name (Nombre del rol), elija el nombre del rol de servicio en la lista.
7. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
8. En Paso 3: agregar la etapa de origen, en Proveedor de origen, elija Amazon S3. En Bucket, escriba el nombre del bucket de S3·que creó en Paso 1: creación de un bucket de origen de S3 para la aplicación. En la clave del objeto de S3, escriba la clave del objeto con o sin una ruta de archivo y recuerde incluir la extensión de archivo. Por ejemplo, en `SampleApp_Windows.zip`, escriba el nombre del archivo de ejemplo tal y como aparece en este ejemplo:

   ```
   SampleApp_Windows.zip
   ```

   Elija Next Step (Paso siguiente).

   En Change detection options, deje los valores predeterminados. Esto permite CodePipeline usar Amazon CloudWatch Events para detectar cambios en tu bucket de origen.

   Elija Siguiente.
9. En Paso 4: agregar la etapa de compilación, elija Omitir la etapa de compilación y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más. Elija Siguiente.

10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.

11. En el paso 6: Agregue la etapa de implementación y, en el proveedor de implementación, elija CodeDeploy . El valor predeterminado del campo Región es el mismo Región de AWS que el de tu canalización. En Application name (Nombre de la aplicación), escriba `MyDemoApplication` o haga clic en el botón Refresh (Actualizar) y, a continuación, elija el nombre de la aplicación en la lista. En Deployment group (Grupo de implementación), escriba **`MyDemoDeploymentGroup`** o elija un grupo de la lista. A continuación, elija Next (Siguiente).

    > **Note**
    >
    > El nombre "Deploy" (Implementación) es el que se asigna de forma predeterminada a la etapa creada en Step 4: Add deploy stage (Paso 4: Agregar la etapa de implementación), mientras que "Source" (Código fuente) es el nombre que recibe la primera etapa de la canalización.

12. En el paso 7: Revisa, revisa la información y, a continuación, selecciona Crear canalización.
13. La canalización comienza a ejecutarse. Puede ver los mensajes de progreso y éxito y fracaso a medida que el CodePipeline ejemplo implementa una página web en cada una de las instancias de Amazon EC2 de la CodeDeploy implementación.

¡Enhorabuena! Acaba de crear una canalización sencilla en. CodePipeline La canalización tiene dos etapas:

- Una etapa denominada Source (Código fuente), que detecta los cambios en la aplicación de ejemplo con control de versiones almacenada en el bucket de S3 y los transfiere a la canalización.
- Una etapa Deploy (Implementación) que implementa dichos cambios en las instancias EC2 con CodeDeploy.

Ahora compruebe los resultados.

Para comprobar que la canalización se ha ejecutado correctamente

1. Vea el progreso inicial de la canalización. El estado de cada etapa cambia de No executions yet (Sin ejecuciones) a In Progress (En curso) y después a Succeeded (Realizado correctamente) o Failed (Error). La canalización debería completar la primera ejecución en unos minutos.
2. Cuando el estado de la acción aparezca como Realizado correctamente, en el área de estado de la etapa Implementar, elija Detalles. Esto abre la CodeDeploy consola.
3. En la pestaña Grupo de implementaciones de la sección Eventos del ciclo de vida de la implementación, elija el ID de la instancia. Se abrirá la consola de EC2.
4. En la pestaña Description (Descripción), en Public DNS (DNS público), copie la dirección y después péguela en la barra de direcciones de su explorador web. Vea la página de índice de la aplicación de ejemplo que cargó en el bucket de S3.

   La página web muestra la aplicación de ejemplo que cargó en el bucket de S3.

Para obtener más información sobre las etapas, las acciones y cómo funcionan las canalizaciones, consulte CodePipeline conceptos.

## (Opcional) Paso 5: Agregar otra etapa a la canalización

Ahora agregaremos otra etapa a la canalización para implementar desde los servidores de ensayo a los servidores de producción utilizando CodeDeploy. En primer lugar, debe crear otro grupo de implementación CodePipelineDemoApplication en la ventana CodeDeploy. A continuación tiene que añadir una etapa con una acción que utilice ese grupo de implementaciones. Para añadir otra etapa, utiliza la CodePipeline consola o la AWS CLI para recuperar y editar manualmente la estructura de la canalización en un archivo JSON y, a continuación, ejecutar el update-pipeline comando para actualizar la canalización con los cambios.

Temas

- Crea un segundo grupo de implementación en CodeDeploy
- Agregar el grupo de implementación como otra etapa de la canalización

## Crea un segundo grupo de implementación en CodeDeploy

> **Note**
>
> En esta parte del tutorial, va a crear un segundo grupo de implementaciones, pero la implementación se hará en las mismas instancias de Amazon EC2 usadas anteriormente. Esto solo se hace con fines ilustrativos. Está diseñado a propósito para no mostrarle cómo se muestran los errores en CodePipeline.

Para crear un segundo grupo de implementación en CodeDeploy

1. Abra la CodeDeploy consola en https://console.aws.amazon.com/codedeploy.
2. Elija Applications (Aplicaciones) y, en la lista de aplicaciones, elija `MyDemoApplication`.
3. Elija la pestaña Deployment groups (Grupos de implementaciones) y , a continuación, Create deployment group (Crear grupo de implementaciones).
4. En la página Create deployment group (Crear grupo de implementaciones), en Deployment group name (Nombre de grupo de implementaciones), escriba un nombre para el segundo grupo de implementaciones, como **`CodePipelineProductionFleet`**.
5. En Función de servicio, elija la misma función de CodeDeploy servicio que utilizó para la implementación inicial (no la función de CodePipeline servicio).
6. En Tipo de implementación, elija In-place.
7. En Environment configuration (Configuración de entorno), elija Amazon EC2 Instances (Instancias de Amazon EC2). Elija Nombre en el cuadro Clave y, en el cuadro Valor, seleccione `MyCodePipelineDemo` en la lista. Mantenga la configuración predeterminada de Deployment settings (Ajustes de implementación).
8. En Deployment configuration (Configuración de implementación), elija `CodeDeployDefault.OneAtaTime`.
9. En Load Balancer (Balanceador de carga), anule la selección de Enable load balancing (Habilitar balanceo de carga).
10. Elija Crear grupo de implementación.

## Agregar el grupo de implementación como otra etapa de la canalización

Ahora que tiene otro grupo de implementación, puede agregar una etapa que lo utilice para realizar la implementación en las mismas instancias EC2 que usamos antes. Puede usar la CodePipeline consola o la AWS CLI para agregar esta etapa.

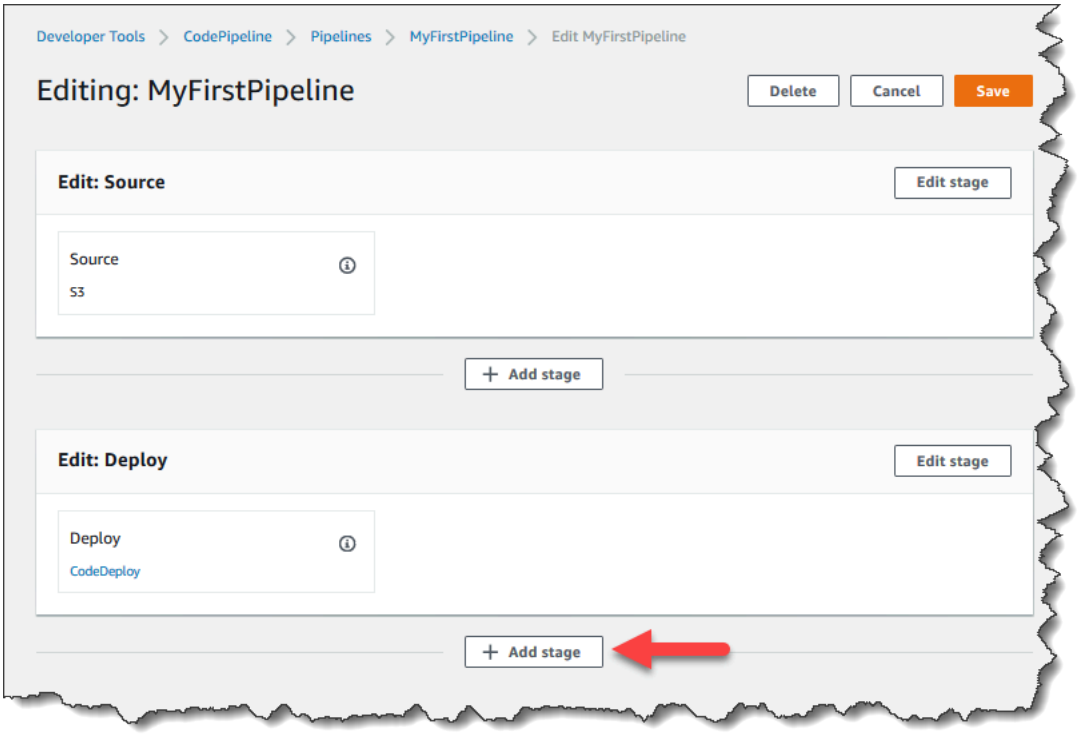
Temas


- Crear una tercera etapa (consola)
- Crear una tercera etapa (CLI)


### Crear una tercera etapa (consola)

Puede usar la CodePipeline consola para agregar una nueva etapa que use el nuevo grupo de implementación. Como este grupo de implementación utiliza las instancias EC2 que ya habíamos empleado, la acción de implementación en esta etapa produce un error.

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En Name (Nombre), elija el nombre de la canalización que ha creado, MyFirstPipeline.
3. En la página de detalles de la canalización, elija Edit.
4. En la página Edit (Editar), elija + Add stage (Añadir etapa) para añadir una etapa inmediatamente después de la etapa Deploy (Implementar).




5. En Add stage (Añadir etapa), in Stage name (Nombre de etapa), escriba **`Production`**. Elija Add stage (Añadir etapa).
6. En la nueva etapa, elija +Add action group (+Añadir grupo de acciones).
7. En Edit action (Editar acción), en Action name (Nombre de acción), escriba **`Deploy-Second-Deployment`**. En el proveedor de acciones, en Implementar, elija CodeDeploy.
8. En la CodeDeploy sección, en Nombre de la aplicación, elija una opción `MyDemoApplication` de la lista desplegable, tal y como hizo cuando creó la canalización. En Deployment group (Grupo de implementación), elija el grupo de implementaciones que acaba de crear, **`CodePipelineProductionFleet`**. En Input artifacts (Artefactos de entrada), elija el artefacto de entrada de la acción de origen. Seleccione Save.
9. En la página Edit (Editar), elija Save (Guardar). En Save pipeline changes (Guardar los cambios de la canalización), elija Save (Guardar).
10. Aunque se ha añadido la nueva etapa a la canalización, aparece el estado No executions yet (Sin ejecuciones) porque no se han producido cambios que activen otra ejecución de la canalización. Tiene que volver a ejecutar manualmente la última revisión para ver cómo se ejecuta la canalización editada. En la página de detalles de canalización, elija Liberar cambio y, a continuación, elija Liberar cuando se le solicite. Esto ejecuta la revisión más reciente disponible en cada ubicación de código fuente especificada en una acción de código fuente a través de la canalización.

Como alternativa, para volver AWS CLI a ejecutar la canalización, desde una terminal de tu máquina Linux, macOS o Unix local, o desde una línea de comandos de tu máquina Windows local, ejecuta el start-pipeline-execution comando especificando el nombre de la canalización. De este modo se ejecuta por segunda vez la aplicación del bucket de código fuente en la canalización.

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

Este comando devuelve un objeto `pipelineExecutionId`.

11. Vuelve a la CodePipeline consola y, en la lista de canalizaciones, elige MyFirstPipeline abrir la página de visualización.

    La canalización muestra tres etapas y el estado del artefacto en ejecución a lo largo de esas tres etapas. Pueden que transcurran cinco minutos hasta que el proceso de canalización se ejecute en todas las etapas. Verá que la implementación se realiza correctamente en las dos primeras etapas, igual que antes, pero la etapa de producción muestra que la Deploy-Second-Deployment acción falló.

12. En la Deploy-Second-Deployment acción, elige Detalles. Se le redirigirá a la página de la implementación CodeDeploy. En este caso, el error se ha producido porque el primer grupo de instancias se ha implementado en todas las instancias EC2 y no han quedado instancias para el segundo grupo de implementación.

    > **Note**
    >
    > Se trata de un error de diseño, para mostrar lo que ocurre cuando se produce un error en una etapa de la canalización.

### Crear una tercera etapa (CLI)

Aunque usar la AWS CLI para añadir una etapa a tu canalización es más complejo que usar la consola, proporciona más visibilidad de la estructura de la canalización.

Para crear una tercera etapa en la canalización

1. Abra una sesión de terminal en su máquina Linux, macOS o Unix local o a través del símbolo del sistema en su equipo Windows local y ejecute el comando para mostrar la estructura de

la canalización que acaba de crear. Para **MyFirstPipeline**, debería escribir el siguiente comando:

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

Este comando devuelve la estructura de MyFirstPipeline. La primera parte del resultado debería tener un aspecto similar al siguiente:

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

La última parte de la salida incluye los metadatos de la canalización y debería tener un aspecto similar al siguiente:

```
    ...
        ],
        "artifactStore": {
            "type": "S3"
            "location": "amzn-s3-demo-bucket",
        },
        "name": "MyFirstPipeline",
        "version": 4
    },
    "metadata": {
        "pipelineArn": "arn:aws:codepipeline:us-
east-2:80398EXAMPLE:MyFirstPipeline",
        "updated": 1501626591.112,
        "created": 1501626591.112
    }
}
```

2. Copie y pegue esta estructura en un editor de texto sin formato y guarde el archivo como **pipeline.json**. Para su comodidad, guarde este archivo en el mismo directorio en el que ejecuta los comandos aws codepipeline.

> **Note**
>
> Puede enviar JSON directamente a un archivo con el comando get-pipeline del modo siguiente:
>
> ```
> aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
> ```

3. Copie la sección de la etapa Implementar y péguela después de las primeras dos etapas. Al ser una etapa de implementación, al igual que la etapa Implementar, la utilizará como plantilla de la tercera etapa.
4. Cambie el nombre de la etapa y los detalles del grupo de implementación.

   En el siguiente ejemplo, se muestra el archivo JSON que se va a agregar al archivo pipeline.json después de la etapa Implementar. Edite los elementos resaltados con valores nuevos. No olvide incluir una coma para separar las definiciones de las etapas Implementar y Producción.

```
,
{
    "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. Si está trabajando con la estructura de la canalización recuperada mediante el comando get-pipeline, debe eliminar las líneas `metadata` del archivo JSON. De lo contrario, el comando update-pipeline no puede utilizarlo. Elimine las líneas `"metadata": { }` y los campos `"updated"`, `"created"` y `"pipelineARN"`.

   Por ejemplo, quite las siguientes líneas de la estructura:

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

   Guarde el archivo.

6. Ejecute el comando update-pipeline, especificando el archivo JSON de la canalización, de forma similar a como se muestra a continuación:

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

   Este comando devuelve la estructura completa de la canalización actualizada.

   > **Important**
   >
   > Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

7. Ejecute el comando start-pipeline-execution y especifique el nombre de la canalización. De este modo se ejecuta por segunda vez la aplicación del bucket de código fuente en la canalización.

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

   Este comando devuelve un objeto `pipelineExecutionId`.

8. Abra la CodePipeline consola y elija una opción  MyFirstPipeline  de la lista de canalizaciones.

   La canalización muestra tres etapas y el estado del artefacto en ejecución a lo largo de esas tres etapas. Pueden que transcurran cinco minutos hasta que el proceso de canalización se ejecute

en todas las etapas. Si bien la implementación se realiza correctamente en las dos primeras etapas, al igual que antes, la etapa de producción muestra que la Deploy-Second-Deployment acción falló.

9. En la Deploy-Second-Deployment acción, elija Detalles para ver los detalles del error. Se le redirigirá a la página de detalles de la CodeDeploy implementación. En este caso, el error se ha producido porque el primer grupo de instancias se ha implementado en todas las instancias EC2 y no han quedado instancias para el segundo grupo de implementación.

> **Note**
>
> Se trata de un error de diseño, para mostrar lo que ocurre cuando se produce un error en una etapa de la canalización.

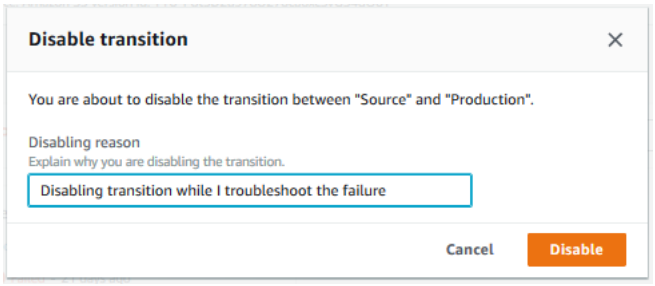
## (Opcional) Paso 6: Deshabilite y habilite las transiciones entre las etapas de CodePipeline

La transición entre las etapas de una canalización puede habilitarse y deshabilitarse. Deshabilitar una transición entre etapas permite controlar manualmente las transiciones entre una etapa y otra. Por ejemplo, puede ser conveniente ejecutar las dos primeras etapas de una canalización, pero deshabilitar las transiciones a la tercera hasta que esté listo para implementar en la producción, o mientras investiga un problema o un error en esa etapa.

Para deshabilitar y habilitar las transiciones entre las etapas de una CodePipeline canalización

1. Abre la CodePipeline consola y elige una opción MyFirstPipeline de la lista de canalizaciones.
2. En la página de detalles de la canalización, elija el botón Deshabilitar la transición situado entre la segunda etapa, Implementar y la tercera etapa que agregó en la sección anterior (Producción).
3. En el cuadro de diálogo Disable transition (Deshabilitar transición), escriba un motivo para deshabilitar la transición entre las etapas y después elija Disable (Deshabilitar).

   La flecha entre etapas muestra un icono y cambia de color, y aparece el botón Enable transition (Habilitar transición).




4. Cargue nuevamente la muestra en el bucket de S3. Como el bucket tiene varias versiones, este cambio inicia la canalización.
5. Vuelva a la página de detalles de la canalización y observe el estado de las etapas. La vista de canalización cambia para mostrar el progreso y el éxito de las primeras dos etapas, pero no se realizan cambios en la tercera etapa. Este proceso puede tardar unos minutos.
6. Habilite la transición eligiendo el botón Enable transition (Habilitar transición) entre las dos etapas. En el cuadro de diálogo Enable transition, elija Enable. La etapa comienza a ejecutarse en unos minutos e intenta procesar el artefacto que ya se ha ejecutado en las dos primeras etapas de la canalización.

> **Note**
>
> Si desea que esta tercera etapa tenga éxito, edite el grupo de CodePipelineProductionFleet implementación antes de habilitar la transición y especifique un conjunto diferente de instancias EC2 en las que se implemente la aplicación. Para obtener más información acerca de cómo hacerlo, consulte Cambio en la configuración de un grupo de implementaciones. Si crea más instancias EC2, se podrían generar costos adicionales.

## Paso 7: Limpiar recursos

Puede reutilizar algunos de los recursos que ha creado aquí en Tutorial: Crear una canalización de cuatro etapas. Por ejemplo, puede reutilizar la CodeDeploy aplicación y la implementación. Puedes configurar una acción de compilación con un proveedor como CodeBuild, por ejemplo, un servicio de compilación totalmente gestionado en la nube. También puede configurar una acción de compilación que utilice un proveedor con un servidor o un sistema de compilación, como Jenkins.

Sin embargo, una vez completado este o cualquier otro tutorial, debe eliminar la canalización y los recursos que utiliza para que no se le cobre por el uso continuado de esos recursos. En primer lugar,

elimina la canalización, luego la CodeDeploy aplicación y sus instancias Amazon EC2 asociadas y, por último, el bucket de S3.

Para limpiar los recursos usados en este tutorial

1. Para limpiar sus CodePipeline recursos, siga las instrucciones que aparecen en Eliminar una canalización en AWS CodePipeline.
2. Para limpiar tus CodeDeploy recursos, sigue las instrucciones de Para limpiar los recursos (consola).
3. Para eliminar el bucket de S3, siga las instrucciones que se especifican en Eliminar o vaciar un bucket de S3. Si no tiene intención de crear más canalizaciones, elimine el bucket de S3 creado para almacenar los artefactos de la canalización. Para obtener más información acerca de este bucket, consulte CodePipeline conceptos.

# Tutorial: Crear una canalización simple (CodeCommit repositorio)

En este tutorial, se utiliza CodePipeline para implementar el código mantenido en un CodeCommit repositorio en una única instancia de Amazon EC2. Su canalización se activa cuando introduce un cambio en el CodeCommit repositorio. La canalización implementa los cambios en una instancia de Amazon EC2 que la utiliza CodeDeploy como servicio de implementación.

> **Important**
>
> Como parte de la creación de una canalización, para los artefactos se utilizará un depósito de artefactos de S3 proporcionado por CodePipeline el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

La canalización tiene dos etapas:

- Una etapa de fuente (Source) para tu CodeCommit acción de fuente.
- Una etapa de despliegue (Deploy) para su acción CodeDeploy de despliegue.

La forma más sencilla de empezar AWS CodePipeline es utilizar el asistente Create Pipeline de la CodePipeline consola.

> **Note**
>
> Antes de empezar, asegúrate de haber configurado tu cliente de Git para que funcione con él CodeCommit. Para obtener instrucciones, consulte Configuración para CodeCommit.

## Paso 1: Crea un CodeCommit repositorio

En primer lugar, debe crear un repositorio en CodeCommit. Su canalización obtendrá el código fuente de este repositorio cuando se ejecute. También creas un repositorio local donde mantienes y actualizas el código antes de subirlo al CodeCommit repositorio.

Para crear un CodeCommit repositorio

1. Abra la CodeCommit consola en https://console.aws.amazon.com/codecommit/.
2. En el selector de regiones, elige Región de AWS dónde quieres crear el repositorio y la canalización. Para obtener más información, consulte Puntos de conexión y Regiones de AWS.
3. En la página Repositorios, seleccione Crear repositorio.
4. En la página Create repository (Crear repositorio), en Repository name (Nombre del repositorio), escriba un nombre para el repositorio (por ejemplo, **`MyDemoRepo`**).
5. Seleccione Crear.

> **Note**
>
> Los pasos restantes de este tutorial se utilizan **`MyDemoRepo`** para el nombre de tu CodeCommit repositorio. Si elige otro nombre, asegúrese de utilizarlo durante todo el tutorial.

Para configurar un repositorio local

En este paso, configurará un repositorio local para conectarse a su repositorio remoto de CodeCommit.

> **Note**
>
> No es necesario configurar un repositorio local. También puede usar la consola para cargar archivos, tal y como se describe en Paso 2: Agrega código de muestra a tu CodeCommit repositorio.

1. Con el nuevo repositorio abierto en la consola, elija Clone URL (Clonar URL) en la parte superior derecha de la página y, a continuación, elija Clone SSH (Clonar SSH). La dirección para clonar el repositorio Git se copia en el portapapeles.
2. En el terminal o en la línea de comandos, acceda a un directorio local en el que desea que se almacene el repositorio local. En este tutorial, utilizamos `/tmp`.
3. Ejecute el siguiente comando para clonar el repositorio, sustituyendo la dirección SSH por la que ha copiado en el paso anterior. Este comando crea un directorio denominado `MyDemoRepo`. Copie una aplicación de ejemplo en este directorio.

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

## Paso 2: Agrega código de muestra a tu CodeCommit repositorio

En este paso, descargas el código de una aplicación de ejemplo que se creó para un tutorial de CodeDeploy muestra y lo agregas a tu CodeCommit repositorio.

1. A continuación, descargue un ejemplo y guárdelo en una carpeta o directorio del equipo local.

   a. Elija una de las siguientes opciones. Elija `SampleApp_Linux.zip` si desea seguir los pasos de este tutorial en instancias de Linux.

   - Si desea realizar la implementación en instancias de Amazon Linux utilizando CodeDeploy, descargue la aplicación de muestra aquí: SampleApp _ Linux.zip.
   - Si desea realizar la implementación en instancias de Windows Server utilizando CodeDeploy, descargue la aplicación de muestra aquí: SampleApp _ Windows.zip.

   La aplicación de ejemplo contiene los siguientes archivos para implementarlos con CodeDeploy:

- `appspec.yml`— El archivo de especificación de la aplicación (AppSpecarchivo) es un archivo con formato YAML que se utiliza CodeDeploy para gestionar una implementación. Para obtener más información sobre el AppSpec archivo, consulte la referencia al CodeDeploy AppSpec archivo en la Guía del AWS CodeDeploy usuario.
- `index.html`: el archivo de índice contiene la página de inicio de la aplicación de ejemplo implementada.
- `LICENSE.txt`: el archivo de licencia contiene la información de licencia de la aplicación de ejemplo.
- Archivos para scripts: la aplicación de ejemplo usa scripts para escribir archivos de texto en una ubicación de la instancia. Se escribe un archivo para cada uno de los varios eventos del ciclo de vida de la CodeDeploy implementación de la siguiente manera:
  - (Solo ejemplos de Linux) Carpeta `scripts`: la carpeta contiene los siguientes scripts de intérprete de comandos para instalar las dependencias e iniciar y detener la aplicación de ejemplo para la implementación automatizada: `install_dependencies`, `start_server` y `stop_server`.
  - (Solo ejemplo de Windows) `before-install.bat`: se trata de un script por lotes para el evento de ciclo de vida de implementación, que se ejecutará para eliminar los archivos antiguos escritos durante implementaciones anteriores de este ejemplo y crear una ubicación en su instancia en la que escribir los archivos nuevos.

b. Descargue el archivo comprimido (en zip).

2. Descomprima los archivos de SampleApp _ Linux.zip en el directorio local que creó anteriormente (por ejemplo, `/tmp/MyDemoRepo` `oc:\temp\MyDemoRepo`).

   Asegúrese de colocar los archivos directamente en su repositorio local. No incluya una carpeta `SampleApp_Linux`. En la máquina Linux, macOS o Unix local, por ejemplo, su directorio y la jerarquía de archivos deberían verse así:

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

3. Para cargar archivos en el repositorio, utilice alguno de los métodos siguientes.

   a. Para usar la CodeCommit consola para cargar los archivos:

      i. Abre la CodeCommit consola y elige tu repositorio en la lista de repositorios.

      ii. Elija Add file (Añadir archivo) y, a continuación, Upload file (Cargar archivo).

      iii. Seleccione Choose file (Elegir archivo) y, a continuación, busque el archivo. Para añadir un archivo a una carpeta, seleccione Crear archivo y, a continuación, introduce el nombre de la carpeta con el nombre del archivo, por ejemplo `scripts/install_dependencies`. Pegue los siguientes contenidos en el nuevo archivo.

      Para confirmar el cambio, introduzca su nombre de usuario y la dirección de correo electrónico.

      Seleccione Confirmar cambios.

      iv. Repita este paso para cada archivo.

      El contenido del repositorio debería ser similar al siguiente:

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

   b. Para usar los comandos de git para cargar sus archivos:

      i. Cambiar los directorios a su repositorio local:

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

      ii. Ejecute el siguiente comando para preparar todos los archivos de una vez:

```
git add -A
```

      iii. Ejecute el siguiente comando para confirmar los archivos con un mensaje de confirmación:

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

iv. Ejecute el siguiente comando para enviar los archivos de su repositorio local al repositorio de CodeCommit:

```
git push
```

4. Los archivos que descargaste y agregaste a tu repositorio local ya se han agregado a la `main` rama de tu CodeCommit `MyDemoRepo` repositorio y están listos para ser incluidos en una canalización.

## Paso 3: Cree una instancia Linux de Amazon EC2 e instale el agente CodeDeploy

En este paso, va a crear una instancia de Amazon EC2 en la que va a implementar una aplicación de ejemplo. Como parte de este proceso, cree un rol de instancia que permita instalar y administrar el CodeDeploy agente en la instancia. El CodeDeploy agente es un paquete de software que permite usar una instancia en CodeDeploy las implementaciones. También se adjuntan políticas que permiten a la instancia recuperar los archivos que el CodeDeploy agente usa para implementar la aplicación y permitir que SSM administre la instancia.

Para crear un rol de instancia

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de la consola, elija Roles.
3. Elija Crear rol.
4. En Seleccionar el tipo de entidad de confianza, seleccione Servicio de AWS. En Elegir un caso de uso, seleccione EC2. En Select your use case (Seleccionar su caso de uso), elija EC2. Elija Siguiente: permisos.
5. Busque y seleccione la política denominada **`AmazonEC2RoleforAWSCodeDeploy`**.
6. Busque y seleccione la política denominada **`AmazonSSMManagedInstanceCore`**. Elija Siguiente: Etiquetas.
7. Elija Siguiente: Revisar. Escriba el nombre del rol (por ejemplo, **`EC2InstanceRole`**).

> **Note**
>
> Anote el nombre del rol para utilizarlo en el siguiente paso. Tendrá que elegir este rol cuando cree la instancia.

Elija Crear rol.

Lanzamiento de una instancia de

1. Abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En la barra de navegación lateral, elija Instancias y seleccione Lanzar instancias en la parte superior de la página.
3. En Name (Nombre), escriba **`MyCodePipelineDemo`**. Esto asigna a la instancia una clave de etiqueta **`Name`** y un valor de etiqueta **`MyCodePipelineDemo`**. Más adelante, creas una CodeDeploy aplicación que implementa la aplicación de ejemplo en esta instancia. CodeDeployselecciona las instancias que se van a implementar en función de las etiquetas.
4. En Imágenes de aplicaciones y sistemas operativos (Amazon Machine Image), busque la opción de AMI de Amazon Linux con el AWS logotipo y asegúrese de que está seleccionada. (Esta AMI se describe como AMI de Amazon Linux 2 (HVM) y se denomina "Free tier eligible" (Apta para capa gratuita).)
5. En Tipo de instancia, elija el tipo `t2.micro` apto para la capa gratuita como configuración de hardware de la instancia.
6. En Par de claves (inicio de sesión), seleccione un par de claves o cree uno.

   Puede elegir Continuar sin un par de claves.

   > **Note**
   >
   > A efectos de este tutorial, puede continuar sin utilizar un par de claves. Si desea usar SSH para conectarse a sus instancias, cree o use un par de claves.

7. En Configuración de red, haga lo siguiente:

   En IP Auto-assign pública, asegúrese de que el estado sea Habilitado.

Para el grupo de seguridad creado, elija HTTP y, a continuación, en Tipo de origen, elija Mi IP.

8. Amplíe Advanced details (Detalles avanzados). En Perfil de instancia de IAM, elija el rol de IAM que creó en el procedimiento anterior (por ejemplo, **`EC2InstanceRole`**).
9. En Resumen, en Número de instancias, especifique 1.
10. Seleccione Iniciar instancia.
11. Puede ver el estado del lanzamiento en la página Instances. Al lanzar una instancia, su estado inicial es `pending`. Una vez iniciada la instancia, el estado cambia a `running` y recibe un nombre de DNS público. (Si no aparece la columna DNS público, elija el Show/Hide icono y, a continuación, seleccione DNS público).

## Paso 4: Crea una aplicación en CodeDeploy

En CodeDeploy, una aplicación es un recurso que contiene la aplicación de software que desea implementar. Más adelante, utilizará esta aplicación CodePipeline para automatizar las implementaciones de la aplicación de muestra en su instancia de Amazon EC2.

En primer lugar, debe crear un rol que permita CodeDeploy realizar despliegues. A continuación, debe crear una aplicación de CodeDeploy .

Para crear un rol de CodeDeploy servicio

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de la consola, elija Roles.
3. Elija Crear rol.
4. En Seleccionar entidad de confianza, elija Servicio de AWS. En Use case (Caso de uso), elija CodeDeploy. Elija una CodeDeploy de las opciones de la lista. Elija Siguiente. La política administrada `AWSCodeDeployRole` ya está asociada al rol.
5. Elija Siguiente.
6. Especifique un nombre para el rol (por ejemplo, **`CodeDeployRole`**) y elija Create role (Crear rol).

Para crear una aplicación en CodeDeploy

1. Abra la CodeDeploy consola en https://console.aws.amazon.com/codedeploy.
2. Si la página Aplicaciones no aparece, en el menú, elija Aplicaciones.

3. Elija Creación de aplicación.
4. En Application name (Nombre de aplicación), escriba **MyDemoApplication**.
5. En  Compute Platform, elige  EC2/On-premises.
6. Elija Creación de aplicación.

Para crear un grupo de implementación en CodeDeploy

Un grupo de implementación es un recurso que define la configuración relacionada con la implementación, como las instancias que implementar y con qué rapidez implementarlas.

1. En la página que muestra su aplicación, elija Create deployment group (Crear grupo de implementaciones).
2. En Nombre de grupo de implementación, escriba **MyDemoDeploymentGroup**.
3. En Rol de servicio, elija el ARN del rol de servicio que creó anteriormente (por ejemplo, **arn:aws:iam::*account_ID*:role/CodeDeployRole**).
4. En Tipo de  implementación, elija  In-place.
5. En Environment configuration (Configuración de entorno), elija Amazon EC2 Instances (Instancias de Amazon EC2). En el campo Clave, introduzca **Name**. En el campo Valor, escriba el nombre que utilizó para etiquetar la instancia (por ejemplo, **MyCodePipelineDemo**).
6. En Configuración del  agente con AWS Systems Manager, elija  Ahora y programe las actualizaciones. Primero, instale el agente en la instancia. La instancia de Linux ya está configurada con el agente SSM y ahora se actualizará con el CodeDeploy agente.
7. En Deployment configuration (Configuración de implementación), elija `CodeDeployDefault.OneAtaTime`.
8. En Equilibrador de carga, asegúrese de que no esté seleccionada la opción Habilitar equilibrio de carga. No es necesario configurar un balanceador de carga ni elegir un grupo de destino para este ejemplo.
9. Elija Crear grupo de implementación.

## Paso 5: Crea tu primera canalización en CodePipeline

Ya está preparado para crear y ejecutar su primera canalización. En este paso, creas una canalización que se ejecuta automáticamente cuando el código se envía a tu CodeCommit repositorio.

Para crear una CodePipeline canalización

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyFirstPipeline`**.
5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.
6. En Función de servicio, seleccione Nueva función de servicio CodePipeline para poder crear una función de servicio en IAM.
7. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
8. En Paso 3: agregar la etapa de origen, en Proveedor de origen, elija CodeCommit. En Nombre del repositorio, elige el nombre del CodeCommit repositorio en Paso 1: Crea un CodeCommit repositorio el que creaste. En Branch name (Nombre de la ramificación), elija `main` y, a continuación, elija Next step (Paso siguiente).

   Tras seleccionar el nombre del repositorio y la sucursal, aparecerá un mensaje en el que se mostrará la regla de Amazon CloudWatch Events que se va a crear para esta canalización.

   En Change detection options, deje los valores predeterminados. Esto permite CodePipeline usar Amazon CloudWatch Events para detectar cambios en tu repositorio de origen.

   Elija Siguiente.
9. En Paso 4: agregar la etapa de compilación, elija Omitir la etapa de compilación y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más. Elija Siguiente.

> **Note**
>
> En este tutorial, va a implementar código que no requiere ningún servicio de compilación, por lo que puede omitir este paso. Sin embargo, si es necesario compilar el código fuente antes de implementarlo en instancias, puede configurar CodeBuild en este paso.

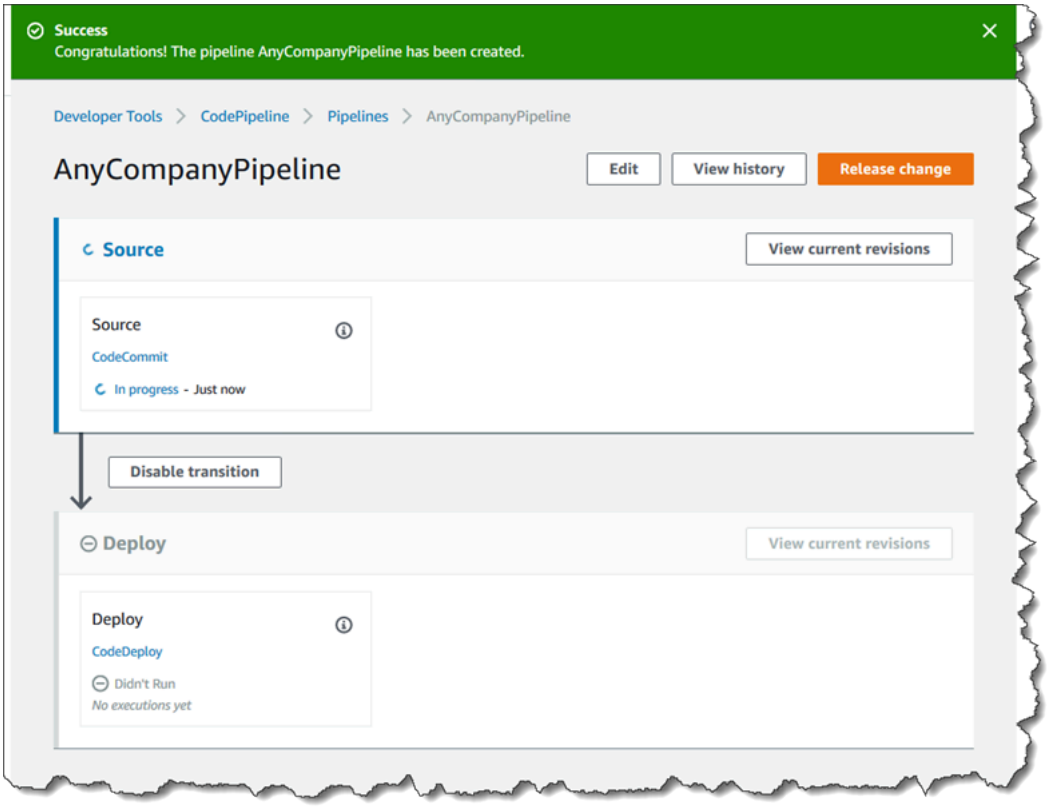
10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.

11. En el paso 6: Agregue la etapa de implementación y, en el proveedor de implementación, elija CodeDeploy. En Application name (Nombre de aplicación), escriba **`MyDemoApplication`**. En Deployment group (Grupo de implementación), elija **`MyDemoDeploymentGroup`** y, a continuación, elija Next step (Paso siguiente).

12. En el paso 7: Revisa, revisa la información y, a continuación, selecciona Crear canalización.

13. La canalización comienza a ejecutarse después de crearla. Descarga el código de su CodeCommit repositorio y crea una CodeDeploy implementación en su instancia EC2. Puede ver los mensajes de progreso y éxito y fracaso a medida que el CodePipeline ejemplo implementa la página web en la instancia Amazon EC2 de la implementación. CodeDeploy




¡Enhorabuena! Acabas de crear una canalización sencilla en CodePipeline.

A continuación, podrá verificar los resultados.

Para comprobar que la canalización se ha ejecutado correctamente

1. Vea el progreso inicial de la canalización. El estado de cada etapa cambia de No executions yet (Sin ejecuciones) a In Progress (En curso) y después a Succeeded (Realizado correctamente) o Failed (Error). La canalización debería completar la primera ejecución en unos minutos.
2. Cuando aparezca Successed para ver el estado de la canalización, en el área de estado de la etapa de implementación, seleccione CodeDeploy. Esto abre la CodeDeploy consola. Si no aparece Succeded, consulteSolución de problemas CodePipeline.
3. En la pestaña Implementaciones, elija el ID de la implementación. En la página de la implementación, en la sección Eventos del ciclo de vida de la implementación, elija el ID de la instancia. Se abrirá la consola de EC2.
4. En la pestaña Description (Descripción), en Public DNS (DNS público), copie la dirección (por ejemplo, `ec2-192-0-2-1.us-west-2.compute.amazonaws.com`) y después péguela en la barra de direcciones de su explorador web.

Aparece la página web de la aplicación de muestra que descargaste y subiste a tu CodeCommit repositorio.

Para obtener más información sobre las etapas, las acciones y cómo funcionan las canalizaciones, consulte CodePipeline conceptos.

## Paso 6: Modifica el código de tu CodeCommit repositorio

Se ha configurado la canalización de modo que se ejecute siempre que se realicen cambios en el código de su repositorio de CodeCommit. En este paso, realizarás cambios en el archivo HTML que forma parte de la CodeDeploy aplicación de ejemplo del CodeCommit repositorio. Cuando envíe estos cambios, la canalización se ejecuta de nuevo y los cambios que realice estarán visibles en la dirección web a la que accedió anteriormente.

1. Cambiar los directorios a su repositorio local:

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

2. Use un editor de texto para modificar el archivo `index.html`:

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

3. Revise el contenido del archivo `index.html` para cambiar el color de fondo y parte del texto de la página web y, a continuación, guarde el archivo.

```
<!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. Para confirmar los cambios e insertarlos en el CodeCommit repositorio, ejecute los siguientes comandos, uno por uno:

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

```
git push
```

Para comprobar que la canalización se ha ejecutado correctamente

1. Vea el progreso inicial de la canalización. El estado de cada etapa cambia de No executions yet (Sin ejecuciones) a In Progress (En curso) y después a Succeeded (Realizado correctamente) o Failed (Error). La ejecución de la canalización debería completarse en unos minutos.

2. Cuando aparezca Succeeded (Correcto) en el estado de la acción, actualice la página de demostración a la que ha accedido anteriormente en el navegador.

   En la página web actualizada se mostrará:

## Paso 7: Limpiar recursos

Puede utilizar algunos de los recursos que ha creado en este tutorial para otros tutoriales de esta guía. Por ejemplo, puedes reutilizar la CodeDeploy aplicación y la implementación. Sin embargo, una vez completado este o cualquier otro tutorial, debe eliminar la canalización y los recursos que utiliza para que no se le cobre por el uso continuado de esos recursos. En primer lugar, elimine la canalización, luego la CodeDeploy aplicación y su instancia Amazon EC2 asociada y, por último, el CodeCommit repositorio.

Para limpiar los recursos usados en este tutorial

1. Para limpiar sus CodePipeline recursos, siga las instrucciones de Eliminar una canalización en AWS CodePipeline.
2. Para limpiar tus CodeDeploy recursos, sigue las instrucciones del tutorial sobre Clean Up Deployment.
3. Para eliminar el CodeCommit repositorio, sigue las instrucciones que se indican en Eliminar un CodeCommit repositorio.

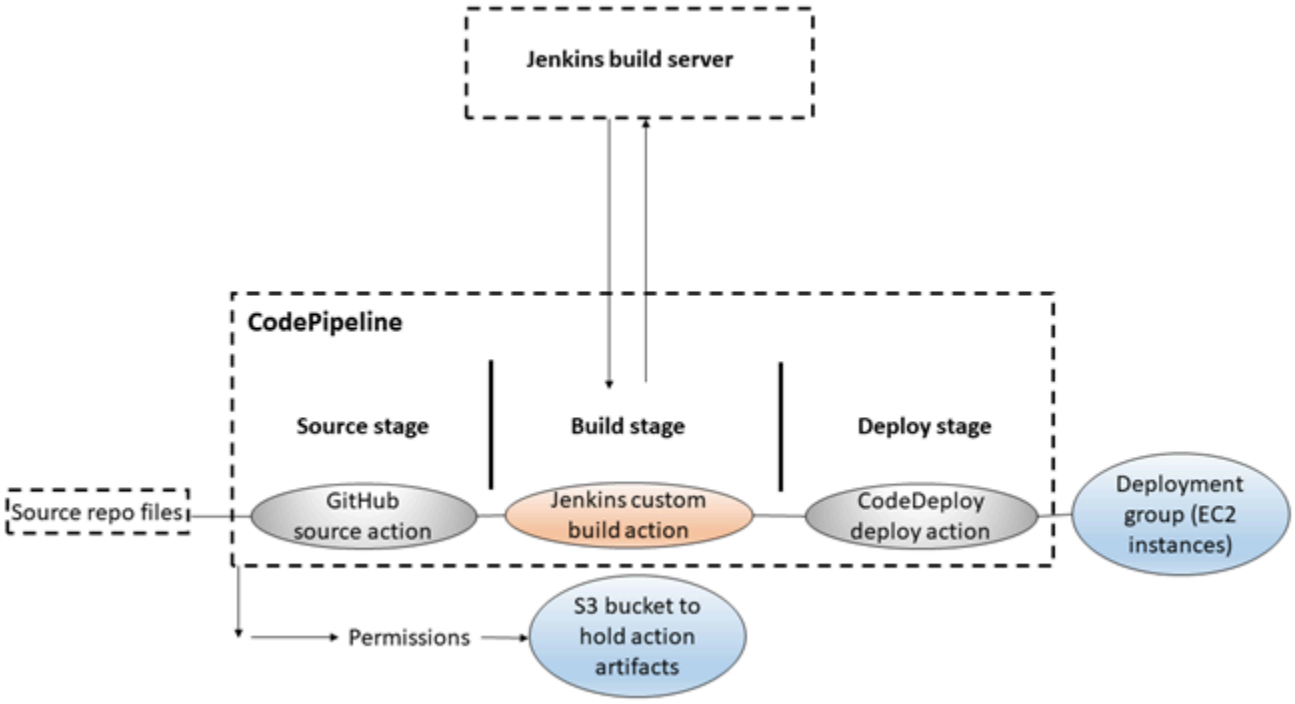
## Paso 8: Documentación adicional

Más información sobre cómo CodePipeline funciona:

- Para obtener más información sobre las etapas, las acciones y cómo funcionan las canalizaciones, consulte CodePipeline conceptos.
- Para obtener información sobre las acciones que puede realizar mediante CodePipeline, consulteIntegraciones con tipos de CodePipeline acciones.
- Pruebe este tutorial más avanzado, Tutorial: Crear una canalización de cuatro etapas. Crea una canalización de varias etapas que incluye un paso que compila código antes de implementarlo.

# Tutorial: Crear una canalización de cuatro etapas

Ahora que ha creado su primera canalización en Tutorial: Crear una canalización simple (bucket de S3) o Tutorial: Crear una canalización simple (CodeCommit repositorio), puede comenzar a crear canalizaciones más complejas. En este tutorial se explica cómo crear una canalización en cuatro etapas que utiliza un GitHub repositorio para el código fuente, un servidor de compilación de Jenkins para crear el proyecto y una CodeDeploy aplicación para implementar el código creado en un servidor provisional. En el siguiente diagrama se muestra la canalización inicial de tres etapas.




Una vez creada la canalización, la editará para añadir una etapa con una acción de prueba para probar el código, utilizando Jenkins también.

Para crear esta canalización, primero debe configurar los recursos necesarios. Por ejemplo, si quieres usar un GitHub repositorio para tu código fuente, debes crear el repositorio antes de poder agregarlo a una canalización. Este tutorial le ayudará a configurar Jenkins en una instancia EC2 con fines demostrativos.

> **Important**
>
> Muchas de las acciones que añades a tu canalización en este procedimiento implican AWS recursos que debes crear antes de crear la canalización. AWS Los recursos para tus acciones de origen siempre deben crearse en la misma AWS región en la que creas tu

canalización. Por ejemplo, si creas tu canalización en la región Este de EE. UU. (Ohio), tu CodeCommit repositorio debe estar en la región Este de EE. UU. (Ohio).
Puedes añadir acciones interregionales al crear tu canalización. AWS Los recursos para las acciones interregionales deben estar en la misma AWS región en la que planeas ejecutar la acción. Para obtener más información, consulte Agregue una acción interregional en CodePipeline.

**Important**

Como parte de la creación de una canalización, se utilizará un depósito de artefactos de S3 proporcionado por el cliente CodePipeline para los artefactos. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

Antes de empezar este tutorial, debe haber completado los requisitos previos generales indicados en Empezar con CodePipeline.

Temas


- Paso 1: completar los requisitos previos de
- Paso 2: Crea una canalización en CodePipeline
- Paso 3: Agregar otra etapa a la canalización
- Paso 4: limpie los recursos


## Paso 1: completar los requisitos previos de

Para integrarse con Jenkins, es AWS CodePipeline necesario que instales el CodePipeline complemento para Jenkins en cualquier instancia de Jenkins con la que quieras usarlo. CodePipeline También debes configurar un usuario o rol de IAM dedicado para usarlos como permisos entre tu proyecto de Jenkins y. CodePipeline La forma más sencilla de integrar Jenkins CodePipeline es instalar Jenkins en una instancia de EC2 que utilice un rol de instancia de IAM que hayas creado para la integración de Jenkins. Para que los enlaces de la canalización de acciones de Jenkins se conecten correctamente, debe configurar las opciones del proxy y del firewall del servidor o de la instancia EC2 de forma que permitan las conexiones entrantes en el puerto que el proyecto de

Jenkins utiliza. Asegúrese de que ha configurado Jenkins para autenticar a los usuarios y aplicar el control de acceso antes de permitir conexiones en esos puertos (por ejemplo, 443 y 8443 si ha configurado Jenkins para que solo use conexiones HTTPS, o 80 y 8080 si permite conexiones HTTP). Para obtener más información, consulte Securing Jenkins.

> **Note**
>
> En este tutorial se usa un ejemplo de código y se configuran los pasos de compilación necesarios para convertir el ejemplo de Haml a HTML. Puedes descargar el código de muestra de código abierto del GitHub repositorio siguiendo los pasos que se indican en. Copia o clona la muestra en un repositorio GitHub Necesitará tener toda la muestra en su GitHub repositorio, no solo el archivo.zip.
> En este tutorial también se supone lo siguiente:
>
> - Está familiarizado con la instalación y administración de Jenkins y con la creación de proyectos de Jenkins.
> - Ha instalado Rake y la gema Haml para Ruby en el mismo equipo o la misma instancia que aloja el proyecto de Jenkins.
> - Ha establecido las variables de entorno del sistema necesarias para poder ejecutar comandos de Rake desde la línea de comandos o el terminal (por ejemplo, en sistemas Windows, se modifica la variable PATH para incluir el directorio donde se instaló Rake).

Temas


- Copia o clona la muestra en un repositorio GitHub
- Creación de un rol de IAM para usar en la integración de Jenkins
- Instale y configure Jenkins y el CodePipeline complemento para Jenkins


## Copia o clona la muestra en un repositorio GitHub

Para clonar la muestra y enviarla a un GitHub repositorio

1. Descarga el código de muestra del GitHub repositorio o clona los repositorios en tu computadora local. Existen dos muestras de paquetes:

   - Si va a implementar la muestra en instancias de Amazon Linux, RHEL o Ubuntu Server, elija codepipeline-jenkins-aws-codedeploy_linux.zip.

- Si va a implementar la muestra en instancias de Windows Server, elija CodePipeline-Jenkins-AWSCodeDeploy _ Windows.zip.

2. En el repositorio, elija Fork para clonar el repositorio de muestra en un repositorio de su GitHub cuenta. Para obtener más información, consulte la Documentación de GitHub.

## Creación de un rol de IAM para usar en la integración de Jenkins

Como práctica recomendada, considera lanzar una instancia EC2 para alojar tu servidor Jenkins y utilizar una función de IAM para conceder a la instancia los permisos necesarios para interactuar con ella. CodePipeline

1. Inicia sesión en Consola de administración de AWS y abre la consola de IAM en. https://console.aws.amazon.com/iam/
2. En la consola de IAM, en el panel de navegación, seleccione Roles y Crear rol.
3. En Select type of trusted entity (Seleccionar el tipo de entidad de confianza), elija Servicio de AWS. En Choose the service that will use this role (Elegir el servicio que utilizará este rol), elija EC2. En Select your use case (Seleccionar su caso de uso), elija EC2.
4. Elija Siguiente: permisos. En la página Attach permissions policies (Asociar políticas de permisos), seleccione la política administrada `AWSCodePipelineCustomActionAccess` y, a continuación, elija Next: Tags (Siguiente: etiquetas). Elija Siguiente: Revisar.
5. En la página Review (Revisar), en Role name (Nombre del rol), especifique el nombre del rol que va a crear específicamente para la integración de Jenkins (por ejemplo, *`JenkinsAccess`*) y elija Create role (Crear rol).

Cuando cree la instancia EC2 en la que va a instalar Jenkins, en Step 3: Configure Instance Details (Paso 3: Configurar detalles de la instancia), asegúrese de que elige el rol de instancia (por ejemplo, *`JenkinsAccess`*).

Para obtener más información sobre los roles de instancia y Amazon EC2, consulte Roles de IAM para Amazon EC2, Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias de Amazon EC2 y Creación de un rol para delegar permisos en un Servicio de AWS.

## Instale y configure Jenkins y el CodePipeline complemento para Jenkins

Para instalar Jenkins y el CodePipeline complemento para Jenkins

1. Cree la instancia EC2 en la que va a instalar Jenkins y en Step 3: Configure Instance Details (Paso 3: Configurar detalles de instancia), asegúrese de elegir el rol de instancia que ha creado (por ejemplo, *`JenkinsAccess`*). Para obtener más información sobre la creación de instancias EC2, consulte Lanzamiento de una instancia de Amazon EC2 en la Guía del usuario de Amazon EC2.

   > **Note**
   >
   > Si ya posee los recursos de Jenkins que desea usar, puede hacerlo, pero debe crear un usuario de IAM especial, aplicar la política administrada `AWSCodePipelineCustomActionAccess` a ese usuario y configurar y usar las credenciales de acceso de dicho usuario en su recurso de Jenkins. Si desea usar la interfaz de usuario de Jenkins para proporcionar las credenciales, configure Jenkins para que solo permita HTTPS. Para obtener más información, consulte Solución de problemas CodePipeline.

2. Instale Jenkins en la instancia EC2. Para obtener más información, consulte la documentación de Jenkins para instalar Jenkins y comenzar y obtener acceso a Jenkins, además de details of integration with Jenkins en Integraciones de productos y servicios con CodePipeline.
3. Lance Jenkins y en la página de inicio, elija Manage Jenkins.
4. En la página Manage Jenkins, elija Manage Plugins.
5. Elija la pestaña Available (Disponible) y, en el cuadro de búsqueda Filter (Filtrar), escriba **`AWS CodePipeline`**. Selecciona CodePipeline Plugin for Jenkins de la lista y elige Descargar ahora e instalarlo después de reiniciar.
6. En la Plugins/Upgrades página de instalación, selecciona Reiniciar Jenkins cuando la instalación haya finalizado y no haya ningún trabajo en ejecución.
7. Elija Back to Dashboard.
8. En la página de inicio, elija New Item.
9. En Nombre del elemento, introduzca un nombre para el proyecto de Jenkins (por ejemplo,*`MyDemoProject`*). Elija Freestyle project y después haga clic en OK.

> **Note**
>
> Asegúrese de que el nombre del proyecto cumple los requisitos de CodePipeline. Para obtener más información, consulte Cuotas en AWS CodePipeline.

10. En la página de configuración del proyecto, seleccione la casilla Execute concurrent builds if necessary. En Source Code Management (Administración de código fuente), elija AWS CodePipeline. Si ha instalado Jenkins en una instancia EC2 y la ha configurado AWS CLI con el perfil del usuario de IAM que creó para la integración entre Jenkins CodePipeline y Jenkins, deje todos los demás campos vacíos.
11. Seleccione  Avanzado y, en  Proveedor, introduzca un nombre para el proveedor de la acción tal y como aparecerá en CodePipeline (por ejemplo,). *MyJenkinsProviderName* Asegúrese de que el nombre es único y fácil de recordar. Lo usará cuando añada una acción de compilación a la canalización más adelante en este tutorial y nuevamente cuando añada una acción de prueba.

    > **Note**
    >
    > Este nombre de acción debe cumplir los requisitos de denominación de acciones de CodePipeline. Para obtener más información, consulte Cuotas en AWS CodePipeline.

12. En Build Triggers, desmarque las casillas marcadas y seleccione Poll SCM. En Schedule (Programación), escriba cinco asteriscos separados por espacios, de la siguiente manera:

    ```
    * * * * *
    ```

    Esto sondea CodePipeline cada minuto.
13. En Build, elija Add build step. Elija Ejecutar intérprete de comandos (Amazon Linux, RHEL o Ubuntu Server) Ejecutar el comando por lotes (Windows Server) y, a continuación, introduzca lo siguiente:

    ```
    rake
    ```

> **Note**
>
> Asegúrese de que el entorno está configurado con las variables y los valores obligatorios para ejecutar rake; de lo contrario, la compilación presentará errores.

14. Selecciona  Añadir acción posterior a la compilación y, a continuación, elige AWS CodePipeline Publisher. Elija Añadir y, en Ubicaciones de salida de compilación, deje la ubicación en blanco. Esta configuración es la predeterminada. Creará un archivo comprimido al final del proceso de compilación.
15. Elija Save para guardar su proyecto de Jenkins.

## Paso 2: Crea una canalización en CodePipeline

En esta parte del tutorial, va a crear una canalización utilizando el asistente Create Pipeline (Crear canalización).

Para crear un proceso de publicación CodePipeline automatizado

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Si es necesario, use el selector de regiones para cambiar a la región en la que se encuentran los recursos de canalización. Por ejemplo, si ha creado recursos para el tutorial anterior en `us-east-2`, asegúrese de que el selector de regiones esté establecido en Este de EE. UU. (Ohio).

   Para obtener más información sobre las regiones y los puntos finales disponibles CodePipeline, consulte los AWS CodePipeline puntos finales y las cuotas.
3. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
4. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
5. En la página Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba el nombre de la canalización.
6. CodePipeline ofrece canalizaciones de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.

7. En Función de servicio, seleccione Nueva función de servicio CodePipeline para poder crear una función de servicio en IAM.
8. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
9. En la página Paso 3: Agregar etapa de origen, en Proveedor de origen, elija GitHub.
10. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para la acción GitHub de origen, consulteGitHub conexiones.
11. En Paso 4: agregar la etapa de compilación, elija Agregar Jenkins. En Nombre del proveedor, introduce el nombre de la acción que proporcionaste en el CodePipeline complemento para Jenkins (por ejemplo*MyJenkinsProviderName*). Este nombre debe coincidir exactamente con el nombre del CodePipeline complemento para Jenkins. En Server URL (URL del servidor), escriba la URL de la instancia EC2 en la que se ha instalado Jenkins. En Nombre del proyecto, ingresa el nombre del proyecto que creaste en Jenkins, por ejemplo*MyDemoProject*, y luego elige Siguiente.
12. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.
13. En el paso 6: Agrega la etapa de implementación, reutiliza la CodeDeploy aplicación y el grupo de implementación en Tutorial: Crear una canalización simple (bucket de S3) los que creaste. En Deploy provider (Proveedor de implementación), elija CodeDeploy. En Nombre de la aplicación, escriba **`CodePipelineDemoApplication`** o haga clic en el botón de actualización y elija el nombre de la aplicación en la lista. En Deployment group (Grupo de implementación), escriba **`CodePipelineDemoFleet`** o elija un grupo de la lista. A continuación, elija Next (Siguiente).

    > **Note**
    >
    > Puede usar sus propios CodeDeploy recursos o crear otros nuevos, pero puede incurrir en costos adicionales.
14. En el paso 7: Revisa la información y, a continuación, selecciona Crear canalización.
15. La canalización se inicia automáticamente y ejecuta la muestra en la canalización. Puede ver los mensajes de progreso y éxito y fracaso a medida que la canalización compila el ejemplo de Haml en HTML y lo implementa como una página web en cada una de las instancias de Amazon EC2 de la implementación. CodeDeploy

# Paso 3: Agregar otra etapa a la canalización

Ahora, agregará una etapa de prueba y, después, agregará a esa etapa una acción de prueba que use la prueba de Jenkins del ejemplo para determinar si la página web tiene contenido. Esta prueba solo tiene fines ilustrativos.

> **Note**
>
> Si no quisiese añadir otra etapa a la canalización, podría añadir una acción de prueba a la etapa de ensayo de la canalización, antes o después de la acción de implementación.

## Agregar una etapa de prueba a la canalización

Temas

- Buscar la dirección IP de una instancia
- Crear un proyecto de Jenkins para probar la implementación
- Crear una cuarta etapa

### Buscar la dirección IP de una instancia

Para comprobar la dirección IP de una instancia en la que ha implementado el código

1. Cuando el estado de canalización aparezca como Succeeded en el área de estado de la etapa Staging (Ensayo), elija Details.
2. En la sección Deployment Details, en Instance ID, elija el ID de instancia de una de las instancias implementadas correctamente.
3. Copie la dirección IP de la instancia (por ejemplo,). *192.168.0.4* Usará esta dirección IP en la prueba de Jenkins.

### Crear un proyecto de Jenkins para probar la implementación

Para crear el proyecto de Jenkins

1. En la instancia en la que ha instalado Jenkins, abra Jenkins y en la página principal, elija New Item.

2. En Nombre del elemento, introduzca un nombre para el proyecto de Jenkins (por ejemplo,*MyTestProject*). Elija Freestyle project y después haga clic en OK.

   > **Note**
   >
   > Asegúrese de que el nombre de su proyecto cumpla con los CodePipeline requisitos. Para obtener más información, consulte Cuotas en AWS CodePipeline.

3. En la página de configuración del proyecto, seleccione la casilla Execute concurrent builds if necessary. En Source Code Management (Administración de código fuente), elija AWS CodePipeline. Si ha instalado Jenkins en una instancia EC2 y la ha configurado AWS CLI con el perfil del usuario de IAM que creó para la integración entre Jenkins CodePipeline y Jenkins, deje todos los demás campos vacíos.

   > **Important**
   >
   > Si está configurando un proyecto de Jenkins y no está instalado en una instancia de Amazon EC2, o está instalado en una instancia EC2 que ejecuta un sistema operativo Windows, complete los campos según lo exija la configuración del puerto y el host de su proxy, y proporcione las credenciales del usuario o rol de IAM que configuró para la integración entre Jenkins y CodePipeline.

4. Elija Avanzado y en Categoría, elija Prueba.

5. En Provider, introduzca el mismo nombre que utilizó para el proyecto de compilación (por ejemplo,). *MyJenkinsProviderName* Usará este nombre cuando añada la acción de prueba a la canalización más adelante en este tutorial.

   > **Note**
   >
   > Este nombre debe cumplir los requisitos de CodePipeline nomenclatura para las acciones. Para obtener más información, consulte Cuotas en AWS CodePipeline.

6. En Build Triggers, desmarque las casillas marcadas y seleccione Poll SCM. En Schedule (Programación), escriba cinco asteriscos separados por espacios, de la siguiente manera:

   ```
   * * * * *
   ```

   Esto sondea CodePipeline cada minuto.

7. En Build, elija Add build step. Si va a realizar la implementación en instancias de Amazon Linux, RHEL, o Ubuntu Server, elija Ejecutar intérprete de comandos. A continuación, escriba lo siguiente, donde la dirección IP será la dirección de la instancia EC2 que copió anteriormente:

   ```
   TEST_IP_ADDRESS=192.168.0.4 rake test
   ```

   Si va a realizar la implementación en instancias de Windows Server, elija Ejecutar comando por lotes y escriba la siguiente información, donde la dirección IP es la dirección de la instancia EC2 que copió anteriormente:

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

   > **Note**
   >
   > La prueba adopta el puerto 80 como puerto predeterminado. Si desea especificar un puerto diferente, añada una indicación de puerto de prueba, tal y como se indica a continuación:
   >
   > ```
   > TEST_IP_ADDRESS=192.168.0.4 TEST_PORT=8000 rake test
   > ```

8. Selecciona  Añadir acción posterior a la compilación y, a continuación, elige AWS CodePipeline Publisher. No elija Add.
9. Elija Save para guardar su proyecto de Jenkins.

### Crear una cuarta etapa

Para añadir una etapa a la canalización que incluya la acción de prueba de Jenkins

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En Name (Nombre), elija el nombre de la canalización que ha creado, MySecondPipeline.
3. En la página de detalles de la canalización, elija Edit.
4. En la página Edit (Editar), elija + Stage (+ Etapa) para añadir una etapa inmediatamente después de la etapa de compilación.
5. En el campo de nombre de la etapa nueva, escriba un nombre (por ejemplo, **`Testing`**) y elija + Add action group (Agregar grupo de acciones).

6. En Nombre de la acción, introduzca*MyJenkinsTest-Action*. En Proveedor de pruebas, elija el nombre del proveedor que especificó en Jenkins (por ejemplo,*MyJenkinsProviderName*). En Nombre del proyecto, ingresa el nombre del proyecto que creaste en Jenkins (por ejemplo,*MyTestProject*). En Artefacto de entrada, elige el artefacto de la compilación de Jenkins cuyo nombre predeterminado sea y*BuildArtifact*, a continuación, elige Listo.

   > **Note**
   >
   > Dado que la acción de prueba de Jenkins opera en la aplicación integrada en el paso de compilación de Jenkins, utilice el artefacto de compilación para el artefacto de entrada de la acción de prueba.

   Para obtener más información acerca de los artefactos de entrada y salida y de la estructura de las canalizaciones, consulte CodePipeline referencia de estructura de tuberías.

7. En la página Edit, elija Save pipeline changes. En el cuadro de diálogo Save pipeline changes, elija Save and continue.

8. Aunque se ha añadido la nueva etapa a la canalización, el estado No executions yet aparece en esa etapa porque no se han producido cambios que activen otra ejecución de la canalización. Para ejecutar la muestra en la canalización revisada, elija Liberar cambio en la página de detalles de la canalización.

   La vista de canalización muestra las etapas y acciones de su canalización y el estado de la revisión que se ejecuta en esas cuatro etapas. El tiempo que tarde en ejecutarse la canalización en todas las etapas dependerá del tamaño de los artefactos, la complejidad de la compilación y las acciones de prueba y otros factores.

## Paso 4: limpie los recursos

Una vez completado este tutorial, debe eliminar la canalización y los recursos que utiliza para que no se le cobre por el uso continuado de esos recursos. Si no va a seguir utilizándolo CodePipeline, elimine la canalización, luego la CodeDeploy aplicación y sus instancias Amazon EC2 asociadas y, por último, el bucket de Amazon S3 utilizado para almacenar los artefactos. También debe considerar la posibilidad de eliminar otros recursos, como el GitHub repositorio, si no tiene intención de seguir utilizándolos.

Para limpiar los recursos usados en este tutorial

1. Abra una sesión de terminal en su máquina Linux, macOS o Unix local o en un símbolo del sistema en su máquina Windows local y ejecute el comando delete-pipeline para eliminar la canalización creada. En **`MySecondPipeline`**, debería escribir el siguiente comando:

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

   Este comando no devuelve nada.
2. Para limpiar tus CodeDeploy recursos, sigue las instrucciones de Limpiar.
3. Para limpiar los recursos de la instancia, elimine la instancia EC2 en la que instaló Jenkins. Para obtener más información, consulte Eliminación de la instancia.
4. Si no tiene intención de crear más canalizaciones ni de CodePipeline volver a utilizarlas, elimine el bucket de Amazon S3 utilizado para almacenar artefactos para su canalización. Para eliminar el bucket, siga las instrucciones que se indican en Eliminar un bucket.
5. Si no tiene intención de volver a usar los demás recursos de esta canalización, plantéese eliminarlos siguiendo las instrucciones de la guía de ese recurso específico. Por ejemplo, si quieres eliminar el GitHub repositorio, sigue las instrucciones que se indican en Eliminar un repositorio del sitio GitHub web.

# Tutorial: configurar una regla de CloudWatch eventos para recibir notificaciones por correo electrónico sobre los cambios en el estado de los oleoductos

Tras configurar una canalización AWS CodePipeline, puede configurar una regla de CloudWatch eventos para enviar notificaciones siempre que se produzcan cambios en el estado de ejecución de las canalizaciones o en las etapas o acciones de las canalizaciones. Para obtener más información sobre el uso de CloudWatch los eventos para configurar las notificaciones de los cambios en el estado de las canalizaciones, consulte. Supervisión de CodePipeline eventos

En este tutorial, va a configurar una notificación para enviar un correo electrónico cuando el estado de una canalización cambie a FAILED. En este tutorial se utiliza un método de transformación de entrada al crear la regla de CloudWatch eventos. Transforma los detalles del esquema del mensaje para entregarlo en un formato legible.

> **Note**
>
> Al crear los recursos para este tutorial, como la notificación de Amazon SNS y la regla de CloudWatch eventos, asegúrese de que los recursos se crean en la misma AWS región que su canalización.

Temas

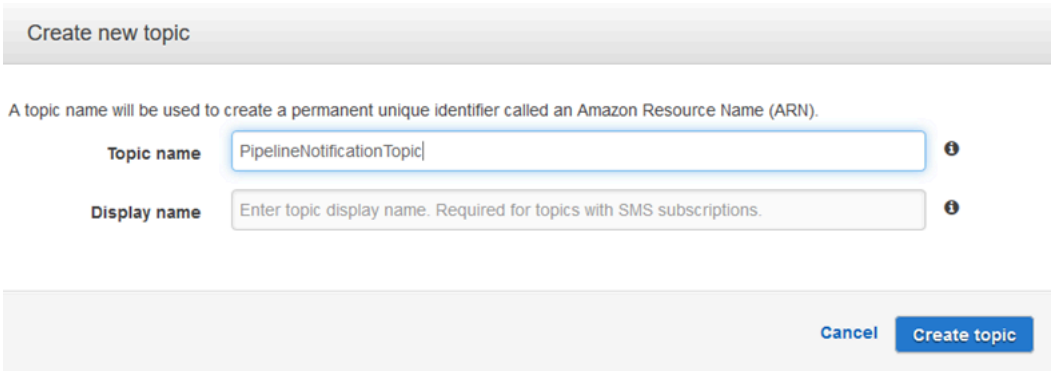

- Paso 1: Configurar una notificación de correo electrónico mediante Amazon SNS
- Paso 2: Crear una regla y agregar el tema de SNS como destino
- Paso 3: Limpiar recursos


## Paso 1: Configurar una notificación de correo electrónico mediante Amazon SNS

Amazon SNS coordina el uso de temas para entregar mensajes a clientes o puntos de conexión de suscripción. Utilice Amazon SNS para crear un tema de notificación y, a continuación, suscríbase al tema con su dirección de correo electrónico. El tema de Amazon SNS se añadirá como objetivo a su regla de CloudWatch eventos. Para obtener más información, consulte la Guía para desarrolladores de Amazon Simple Notification Service.

Cree o identifique un tema en Amazon SNS. CodePipeline utilizará CloudWatch Events para enviar notificaciones sobre este tema a través de Amazon SNS. Para crear un tema:

1. Abra la consola de Amazon SNS en https://console.aws.amazon.com/sns.
2. Seleccione Crear tema.
3. En el cuadro de diálogo Create new topic (Crear un nuevo tema), en Topic name (Nombre del tema), escriba un nombre para el tema (por ejemplo, **`PipelineNotificationTopic`**).



4. Seleccione Crear tema.

   Para obtener más información, consulte Crear un tema en la Guía para desarrolladores de Amazon SNS.

Suscriba a uno o varios destinatarios al tema para que reciban notificaciones por correo electrónico. Para suscribir a un destinatario a un tema:

1. En la consola de Amazon SNS, en la lista Temas, seleccione la casilla situada junto al tema nuevo. Elija Acciones, Suscribirse a tema.
2. En el cuadro de diálogo Create subscription, compruebe que aparece un ARN en Topic ARN.
3. En Protocolo, elige Correo electrónico.
4. En Endpoint, escriba la dirección de correo electrónico completa del destinatario.
5. Elija Create Subscription.
6. Amazon SNS envía un correo electrónico de confirmación de suscripción al destinatario. Para recibir notificaciones por correo electrónico, el destinatario debe utilizar el enlace Confirm subscription de este correo electrónico. Cuando el destinatario hace clic en el enlace, si se ha suscrito correctamente, Amazon SNS muestra un mensaje de confirmación en el navegador web.

   Para obtener más información, consulte Suscribirse a un tema en la Guía del desarrollador de Amazon SNS.

## Paso 2: Crear una regla y agregar el tema de SNS como destino

Cree una regla de notificación de CloudWatch eventos con el origen del evento CodePipeline como fuente de eventos.

1. Abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. En el panel de navegación, seleccione Events (Eventos).
3. Seleccione Creación de regla. En Event source (Origen de eventos), elija AWS CodePipeline. Como Event Type, elija Pipeline Execution State Change.
4. Seleccione Specific state(s) (Estado[s] específico[s]) y, a continuación, elija **`FAILED`**.

5. Elija Edit para abrir el editor de JSON para el panel Event Pattern Preview. Añada el parámetro **`pipeline`** con el nombre de la canalización, tal y como se muestra en el siguiente ejemplo para una canalización denominada “myPipeline”.

   Puede copiar el patrón de eventos aquí y pegarlo en la consola:

   ```
   {
     "source": [
       "aws.codepipeline"
     ],
     "detail-type": [
       "CodePipeline Pipeline Execution State Change"
     ],
     "detail": {
       "state": [
         "FAILED"
       ],
       "pipeline": [
         "myPipeline"
       ]
     }
   }
   ```

6. En Targets, seleccione Add target.
7. En la lista de destinos, elija SNS topic. En Topic, introduzca el tema que ha creado.
8. Expanda Configure input, a continuación, elija Input Transformer.
9. En el cuadro Input Path, escriba los siguientes pares clave-valor.

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

   En el cuadro Input Template, escriba lo siguiente:

   ```
   "The Pipeline <pipeline> has failed."
   ```

10. Seleccione Configurar los detalles.
11. En la página Configure rule details, escriba un nombre y una descripción opcional. Para State, deje seleccionado el cuadro Enabled.
12. Seleccione Creación de regla.

13. Confirma que ahora CodePipeline está enviando notificaciones de compilación. Por ejemplo, compruebe si hay correos electrónicos de notificación de compilación en su bandeja de entrada.
14. Para cambiar el comportamiento de una regla, en la CloudWatch consola, elige la regla y, a continuación, elige Acciones y Editar. Edite la regla, elija Configurar detalles y, a continuación, elija Actualizar regla.

    Para dejar de usar una regla para enviar notificaciones de compilación, en la CloudWatch consola, elige la regla y, a continuación, selecciona Acciones y Desactivar.

    Para eliminar una regla, en la CloudWatch consola, selecciónela y, a continuación, seleccione Acciones y Eliminar.

### Paso 3: Limpiar recursos

Una vez completado este tutorial, debe eliminar la canalización y los recursos que utiliza para que no se le cobre por el uso continuado de esos recursos.

Para obtener información sobre cómo limpiar la notificación de SNS y eliminar la regla de Amazon CloudWatch Events, consulte Clean Up (darse de baja de un tema de Amazon SNS) y consulte la referencia `DeleteRule` en la referencia de la API de Amazon CloudWatch Events.

## Tutorial: Crea una canalización que compile y pruebe tu aplicación para Android con AWS Device Farm

Puedes usarlo AWS CodePipeline para configurar un flujo de integración continuo en el que tu aplicación se compila y se prueba cada vez que se envía una confirmación. En este tutorial, se muestra cómo crear y configurar una canalización para compilar y probar tu aplicación de Android con el código fuente en un GitHub repositorio. La canalización detecta la llegada de una nueva GitHub confirmación y, a continuación, se utiliza CodeBuildpara compilar la aplicación y Device Farm para probarla.

> **Important**
>
> Como parte de la creación de una canalización en la consola, para los artefactos se utilizará un depósito de artefactos CodePipeline de S3. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a

la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y fiable.

**Important**

Muchas de las acciones que añades a la canalización en este procedimiento implican AWS recursos que debes crear antes de crear la canalización. AWS Los recursos para las acciones de origen siempre deben crearse en la misma AWS región en la que se creó la canalización. Por ejemplo, si creas tu canalización en la región EE.UU. Este (Ohio), tu CodeCommit repositorio debe estar en la región EE.UU. Este (Ohio).
Puedes añadir acciones entre regiones al crear tu canalización. AWS los recursos para las acciones entre regiones deben estar en la misma AWS región en la que planeas ejecutar la acción. Para obtener más información, consulte Agregue una acción interregional en CodePipeline.

Puede probarlo con su aplicación Android actual y las definiciones de prueba, o usar la aplicación de muestra y las definiciones de prueba que proporciona Device Farm.

Antes de empezar

1. Inicia sesión en la AWS Device Farm consola y selecciona Crear un proyecto nuevo.
2. Elija el proyecto. En el navegador, copie la URL de su nuevo proyecto. La dirección URL contiene el ID del proyecto.
3. Copie y conserve este ID de proyecto. Lo usará al crear la canalización en CodePipeline.

   Aquí mostramos una URL de ejemplo para un proyecto. Para extraer el ID del proyecto, copie el valor detrás de `projects/`. En este ejemplo, el ID del proyecto es `eec4905f-98f8-40aa-9afc-4c1cfexample`.

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

## Configure CodePipeline para usar sus pruebas de Device Farm

1.

Agrega y confirma un archivo llamado `buildspec.yml`raíz del código de tu aplicación y envíalo a tu repositorio. CodeBuild usa este archivo para ejecutar los comandos y acceder a los artefactos necesarios para compilar tu aplicación.

```
version: 0.2

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

2. (Opcional) Si usa Calabash o Appium para probar su aplicación, añada el archivo de definición de prueba al repositorio. Después podrá configurar CodeBuild para usar las definiciones con el fin de llevar a cabo el conjunto de pruebas.

   Si utiliza las pruebas de Device Farm incorporadas, puede omitir este paso.

3. Para crear la canalización y añadir una etapa de código fuente, haga lo siguiente:

   a. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

   b. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.

   c. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.

   d. En la página Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba el nombre de la canalización.

   e. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios de CodePipeline, consulte Precios.

   f. En Service role (Rol de servicio), deje la opción New service role (Nuevo rol de servicio) seleccionada y no haga ningún cambio en Role name (Nombre de rol). También puede usar un rol de servicio que haya creado anteriormente.

> **Note**
>
> Si utilizas un rol de CodePipeline servicio que se creó antes de julio de 2018, tendrás que añadir permisos para Device Farm. Para ello, abra la consola de IAM, busque el rol y, a continuación, añada los siguientes permisos a la política del rol. Para obtener más información, consulte Agrega permisos al rol de CodePipeline servicio.
>
> ```
> {
>      "Effect": "Allow",
>      "Action": [
>         "devicefarm:ListProjects",
>         "devicefarm:ListDevicePools",
>         "devicefarm:GetRun",
>         "devicefarm:GetUpload",
>         "devicefarm:CreateUpload",
>         "devicefarm:ScheduleRun"
>      ],
>      "Resource": "*"
> }
> ```

g. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).

h. En la página Paso 3: Agregar la etapa de origen, en Proveedor de fuentes, elija GitHub (mediante GitHub la aplicación).

i. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para la acción GitHub de origen, consulteGitHub conexiones.

j. En Repository (Repositorio), elija el repositorio de código fuente.

k. En Branch (Ramificación), elija la ramificación que desea utilizar.

l. Deje los valores predeterminados restantes para la acción de origen. Elija Siguiente.

4. En Paso 4: agregación de la etapa de compilación, agregue una etapa de compilación:

   a. En Proveedor de compilación, elija Otros proveedores de compilación y, a continuación, elija AWS CodeBuild. En el campo Region (Región) conserve el valor predeterminado de la región de la canalización.

   b. Elija Crear proyecto.

c. En Project name (Nombre de proyecto), escriba un nombre para este proyecto de compilación.

d. En Environment image (Imagen de entorno), elija Managed image (Imagen administrada). En Operating system (Sistema operativo), elija Ubuntu.

e. En Runtime, elija Standard (Estándar). En Imagen, escoja: 5.0. aws/codebuild/standard

CodeBuild usa esta imagen del sistema operativo, que tiene instalado Android Studio, para compilar tu aplicación.

f. En Rol de servicio, elige tu rol de CodeBuild servicio actual o crea uno nuevo.

g. En Build specifications (Especificaciones de compilación), elija Use a buildspec file (Usar un archivo buildspec).

h. Selecciona Continuar a CodePipeline. Esto vuelve a la CodePipeline consola y crea un CodeBuild proyecto que utiliza el contenido del `buildspec.yml` repositorio para la configuración. El proyecto de compilación utiliza un rol de servicio para administrar los permisos del Servicio de AWS . Es posible que este paso tarde un par de minutos.

i. Elija Siguiente.

5. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

Elija Siguiente.

6. En la página Paso 6: Añadir fase de despliegue, seleccione Omitir fase de despliegue y, a continuación, acepte el mensaje de advertencia seleccionando Omitir de nuevo. Elija Siguiente.

7. En el paso 7: Revisar, selecciona Crear canalización. Debe ver un diagrama que muestra las etapas de código fuente y de compilación.

8. Añada una acción de prueba de Device Farm a su canalización:

a. En la parte superior derecha, elija Edit (Editar).

b. En la parte inferior del diagrama, seleccione + Add stage (Añadir etapa). En Nombre de la etapa, escriba un nombre; por ejemplo, **`Test`**.

c. Elija + Add action group (Añadir grupo de acciones).

d. En Nombre de la acción, escriba un nombre.

e. En Proveedor de la acción, elija AWS Device Farm. En el campo Region (Región) conserve el valor predeterminado de la región de la canalización.

f. En Input artifacts (Artefactos de entrada), elija el artefacto de entrada que coincida con el artefacto de salida de la etapa anterior a la de prueba, como `BuildArtifact`.

   En la AWS CodePipeline consola, puedes encontrar el nombre del artefacto de salida de cada etapa pasando el ratón sobre el icono de información del diagrama de canalización. Si tu proceso de procesamiento prueba tu aplicación directamente desde la etapa de origen, elige. SourceArtifact Si la canalización incluye una etapa de compilación, elige BuildArtifact.

g. En ProjectId, introduce tu ID de proyecto de Device Farm. Siga los pasos que se indican al principio de este tutorial para recuperar el ID del proyecto.

h. En DevicePoolArn, introduzca el ARN del grupo de dispositivos. Para obtener el conjunto de dispositivos disponible ARNs para el proyecto, incluido el ARN de los principales dispositivos, utilice la AWS CLI para introducir el siguiente comando:

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

i. En AppType, ingresa Android.

   A continuación, se muestra una lista de valores válidos para AppType:

   - iOS
   - Android
   - Web

j. En App (Aplicación), escriba la ruta del paquete de la aplicación compilado. La ruta es relativa a la raíz del artefacto de entrada de la etapa de prueba. Típicamente, esta ruta es similar a `app-release.apk`.

k. En TestType, introduce tu tipo de prueba y, a continuación, en Prueba, introduce la ruta del archivo de definición de la prueba. La ruta es relativa a la raíz del artefacto de entrada de la prueba.

   A continuación, se muestra una lista de valores válidos para TestType:

   - APPIUM_JAVA_JUNIT
   - APPIUM_JAVA_TESTNG
   - APPIUM_NODE
   - APPIUM_RUBY
   - APPIUM_PYTHON

- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- BUILTIN_FUZZ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

> **Note**
>
> No se admiten los nodos de entorno personalizados.

l. En los campos restantes, proporcione la configuración que sea adecuada para su prueba y tipo de aplicación.

m. (Opcional) En Advanced (Avanzado), proporcione información acerca de la configuración en la ejecución de prueba.

n. Seleccione Save.

o. En la etapa que está editando, elija Done (Listo). En el panel de AWS CodePipeline , elija Save (Guardar) y, a continuación, elija Save (Guardar) cuando aparezca el mensaje de advertencia.

p. Para enviar los cambios y comenzar una compilación de canalización, seleccione Publicar modificación y, a continuación, Publicar.

# Tutorial: Crea una canalización que pruebe tu aplicación para iOS con AWS Device Farm

Puede usarlo AWS CodePipeline para configurar fácilmente un flujo de integración continuo en el que su aplicación se pruebe cada vez que cambie el bucket de origen. En este tutorial se muestra cómo crear y configurar una canalización para probar su aplicación iOS compilada desde un bucket de S3. La canalización detecta la llegada de un cambio guardado a través de Amazon CloudWatch Events y, a continuación, utiliza Device Farm para probar la aplicación creada.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará un depósito de artefactos de S3 proporcionado por el cliente CodePipeline para fabricar artefactos. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y fiable.

> **Important**
>
> Muchas de las acciones que añades a la canalización en este procedimiento implican AWS recursos que debes crear antes de crear la canalización. AWS Los recursos para las acciones de origen siempre deben crearse en la misma AWS región en la que se creó la canalización. Por ejemplo, si creas tu canalización en la región EE.UU. Este (Ohio), tu CodeCommit repositorio debe estar en la región EE.UU. Este (Ohio).
> Puedes añadir acciones entre regiones al crear tu canalización. AWS los recursos para acciones entre regiones deben estar en la misma AWS región en la que planeas ejecutar la acción. Para obtener más información, consulte Agregue una acción interregional en CodePipeline.

Puede probarlo con la aplicación iOS de muestra o usar la suya.

Antes de empezar

1. Inicia sesión en la AWS Device Farm consola y selecciona Crear un proyecto nuevo.
2. Elija el proyecto. En el navegador, copie la URL de su nuevo proyecto. La dirección URL contiene el ID del proyecto.
3. Copie y conserve este ID de proyecto. Lo usará al crear la canalización en CodePipeline.

   Aquí mostramos una URL de ejemplo para un proyecto. Para extraer el ID del proyecto, copie el valor detrás de `projects/`. En este ejemplo, el ID del proyecto es `eec4905f-98f8-40aa-9afc-4c1cfexample`.

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

# Configure CodePipeline para usar sus pruebas de Device Farm (ejemplo de Amazon S3)

1. Cree o utilice un bucket de S3 con el control de versiones habilitado. Puede seguir las instrucciones de Paso 1: creación de un bucket de origen de S3 para la aplicación para crear un bucket de S3.
2. En la consola de Amazon S3 para su bucket, elija Cargar y siga las instrucciones para cargar el archivo .zip.

   La aplicación compilada de muestra debe estar empaquetada en un archivo .zip.
3. Para crear la canalización y añadir una etapa de código fuente, haga lo siguiente:

   a. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

   b. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.

   c. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.

   d. En la página Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba el nombre de la canalización.

   e. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios de CodePipeline, consulte Precios.

   f. En Service role (Rol de servicio), deje la opción New service role (Nuevo rol de servicio) seleccionada y no haga ningún cambio en Role name (Nombre de rol). También puede usar un rol de servicio que haya creado anteriormente.

   > **Note**
   >
   > Si usa un rol de CodePipeline servicio que se creó antes de julio de 2018, debe añadir permisos para Device Farm. Para ello, abra la consola de IAM, busque el rol y, a continuación, añada los siguientes permisos a la política del rol. Para obtener más información, consulte Agrega permisos al rol de CodePipeline servicio.
   >
   > ```
   > {
   >     "Effect": "Allow",
   > ```

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

g. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).

h. En la página Paso 3: agregar la etapa de origen, en Proveedor de origen, elija Amazon S3.

i. En Ubicación de Amazon S3, escriba el bucket, como `my-storage-bucket` y la clave de objeto, como `s3-ios-test-1.zip` para su archivo .zip.

j. Elija Siguiente.

4. En Paso 4: agregación de la etapa de compilación, cree una etapa de compilación de marcador de posición para su canalización. De este modo puede crear la canalización en el asistente. Después de usar el asistente para crear su canalización de dos etapas, no se necesita más esta etapa de compilación de marcador de posición. Una vez completada la canalización, se elimina esta segunda etapa y se añade la nueva etapa de prueba en el paso 5.

   a. En Build provider (Proveedor de compilación), elija Add Jenkins (Añadir Jenkins). Esta selección de compilación es un marcador de posición. No se utiliza.

   b. En Provider name (Nombre del proveedor), escriba un nombre. El nombre es un marcador de posición. No se utiliza.

   c. En Server URL (URL del servidor), escriba el texto. El texto es un marcador de posición. No se utiliza.

   d. En Project name (Nombre del proyecto), escriba un nombre. El nombre es un marcador de posición. No se utiliza.

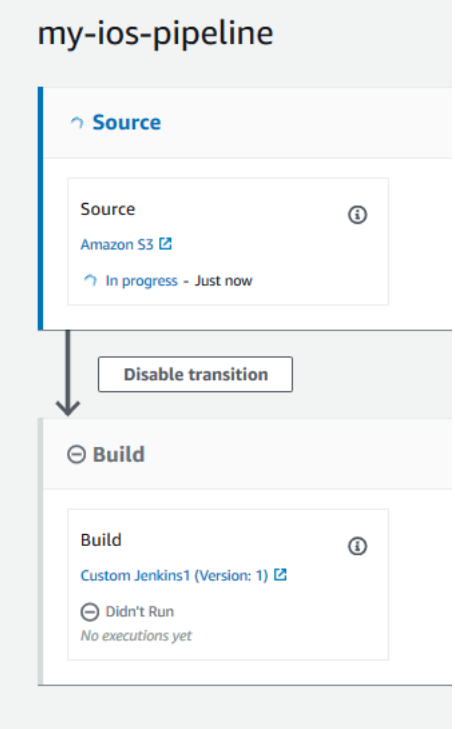
   e. Elija Siguiente.

   f. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

      Elija Siguiente.

g. En la página Paso 6: Añadir fase de despliegue, seleccione Omitir fase de despliegue y, a continuación, acepte el mensaje de advertencia seleccionando Omitir de nuevo.

h. En el paso 7: Revisar, selecciona Crear canalización. Debe ver un diagrama que muestra las etapas de código fuente y de compilación.




5. Añada una acción de prueba de Device Farm a su canalización del siguiente modo:

a. En la parte superior derecha, elija Edit (Editar).

b. Elija Edit stage (Editar etapa). Elija Eliminar. Esto elimina la etapa de marcador de posición ahora que ya no la necesita para la creación de canalizaciones.

c. En la parte inferior del diagrama, seleccione + Add stage (Añadir etapa).

d. En Stage name (Nombre de la etapa), escriba un nombre para esta, por ejemplo, Test, y, a continuación, elija Add stage (Añadir etapa).

e. Elija + Add action group (Añadir grupo de acciones).

f. En Nombre de acción, introduce un nombre, como DeviceFarmTest.

g. En Proveedor de la acción, elija AWS Device Farm. En el campo Region (Región) conserve el valor predeterminado de la región de la canalización.

h. En Input artifacts (Artefactos de entrada), elija el artefacto de entrada que coincida con el artefacto de salida de la etapa anterior a la de prueba, como `SourceArtifact`.

En la AWS CodePipeline consola, para encontrar el nombre del artefacto de salida de cada etapa, coloca el cursor sobre el icono de información del diagrama de canalización. Si tu proceso de procesamiento prueba tu aplicación directamente desde la etapa de origen, elige. SourceArtifact Si la canalización incluye una etapa de compilación, elige BuildArtifact.

i. En ProjectId, elige tu ID de proyecto de Device Farm. Siga los pasos que se indican al principio de este tutorial para recuperar el ID del proyecto.

j. En DevicePoolArn, introduzca el ARN del grupo de dispositivos. Para obtener el conjunto de dispositivos disponible ARNs para el proyecto, incluido el ARN de los principales dispositivos, utilice la AWS CLI para introducir el siguiente comando:

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

k. En AppType, ingresa iOS.

A continuación, se muestra una lista de valores válidos para AppType:

- iOS
- Android
- Web

l. En App (Aplicación), escriba la ruta del paquete de la aplicación compilado. La ruta es relativa a la raíz del artefacto de entrada de la etapa de prueba. Típicamente, esta ruta es similar a `ios-test.ipa`.

m. En TestType, introduzca el tipo de prueba y, a continuación, en Prueba, introduzca la ruta del archivo de definición de la prueba. La ruta es relativa a la raíz del artefacto de entrada de la prueba.

Si utiliza una de las pruebas de Device Farm integradas, escriba el tipo de prueba que ha configurado en el proyecto de Device Farm, por ejemplo BUILTIN_FUZZ. En FuzzEventCount, introduzca un tiempo en milisegundos, como 6000. En FuzzEventThrottle, introduzca un tiempo en milisegundos, como 50.

Si no utiliza una de las pruebas de Device Farm integradas, escriba el tipo de prueba y, en Prueba, escriba la ruta del archivo de definición de prueba. La ruta es relativa a la raíz del artefacto de entrada de la prueba.

A continuación, se muestra una lista de valores válidos para TestType:

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG
- APPIUM_NODE
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- BUILTIN_FUZZ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

> **Note**
>
> No se admiten los nodos de entorno personalizados.

n. En los campos restantes, proporcione la configuración que sea adecuada para su prueba y tipo de aplicación.

o. (Opcional) En Advanced (Avanzado), proporcione información acerca de la configuración en la ejecución de prueba.

p. Seleccione Save.

q. En la etapa que está editando, elija Done (Listo). En el panel de AWS CodePipeline , elija Save (Guardar) y, a continuación, elija Save (Guardar) cuando aparezca el mensaje de advertencia.

r. Para enviar los cambios y comenzar una ejecución de la canalización, elija Release change (Publicar modificación) y, a continuación, Release (Publicar).

# Tutorial: Crear una canalización que se implemente en Service Catalog

Service Catalog le permite crear y aprovisionar productos basados en AWS CloudFormation plantillas.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará un depósito de artefactos de S3 proporcionado por el cliente CodePipeline para los artefactos. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

En este tutorial se muestra cómo crear y configurar una canalización para implementar la plantilla de producto en Service Catalog y entregar los cambios que haya realizado en su repositorio de origen (ya creado en Amazon GitHub S3 CodeCommit o Amazon).

> **Note**
>
> Cuando Amazon S3 es el proveedor de origen de la canalización, debe cargar en el bucket todos los archivos de origen empaquetados como un solo archivo .zip. De lo contrario, la acción de origen dará error.

En primer lugar, debe crear un producto en Service Catalog y, a continuación, crear una canalización en AWS CodePipeline. Este tutorial proporciona dos opciones para la configuración de la implementación:

- Crear un producto en Service Catalog y cargar un archivo de plantilla en el repositorio de origen. Proporciona la versión del producto y la configuración de implementación en la CodePipeline consola (sin un archivo de configuración independiente). Consulte Opción 1: Realizar la implementación en Service Catalog sin un archivo de configuración.

> **Note**
>
> El archivo de plantilla se puede crear en formato YAML o JSON.

- Crear un producto en Service Catalog y cargar un archivo de plantilla en el repositorio de origen. Proporcione la versión del producto y la configuración de implementación en un archivo de configuración distinto. Consulte Opción 2: Realizar la implementación en Service Catalog con un archivo de configuración.

## Opción 1: Realizar la implementación en Service Catalog sin un archivo de configuración

En este ejemplo, subes el archivo de AWS CloudFormation plantilla de muestra para un bucket de S3 y, a continuación, creas tu producto en Service Catalog. A continuación, crea la canalización y especifica la configuración de implementación en la CodePipeline consola.

### Paso 1: Cargar un archivo de plantilla de ejemplo en un repositorio de código fuente

1. Abra un editor de texto. Cree una plantilla de ejemplo pegando lo siguiente en el archivo. Guarde el archivo como `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"
```

```
    }
  }
}
```

Esta plantilla permite AWS CloudFormation crear un bucket de S3 que puede usar Service Catalog.

2. Cargue el archivo `S3_template.json` en el repositorio de AWS CodeCommit .

## Paso 2: Crear un producto en Service Catalog

1. Como administrador de TI, inicie sesión en la consola de Service Catalog., visite la página Productos y, a continuación, elija Cargar nuevo producto.
2. En la página Upload new product (Cargar nuevo producto), complete lo siguiente:
   a. En Product name (Nombre del producto), introduzca el nombre que desea usar para el nuevo producto.
   b. En Description (Descripción), escriba la descripción del catálogo de productos. Esta descripción se muestra en el listado de productos para ayudar al usuario a elegir el producto correcto.
   c. En Provided by (Proporcionado por), escriba el nombre del departamento de TI o del administrador.
   d. Elija Siguiente.
3. (Opcional) En Enter support details (Introducir detalles de soporte), escriba la información de contacto de soporte del producto y elija Next (Siguiente).
4. En Version details (Detalles de la versión), realice lo siguiente:
   a. Elija Upload a template file (Cargar un archivo de plantilla). Busque su archivo `S3_template.json` y cárguelo.
   b. En Version title (Título de versión), escriba el nombre de la versión del producto (por ejemplo, **`devops S3 v2`**).
   c. En Description (Descripción), escriba detalles que distingan esta versión de otras versiones.
   d. Elija Siguiente.
5. En la página Review (Revisar), compruebe que la información es correcta y, a continuación, elija Create (Crear).

6. En la página Products (Productos), en el navegador, copie la URL de su nuevo producto. Contiene el ID del producto. Copie y conserve este ID de producto. Lo usará al crear la canalización en CodePipeline.

   A continuación se muestra la dirección URL de un producto llamado `my-product`. Para extraer el ID del producto, copie el valor entre el signo igual (=) y el signo ampersand (&). En este ejemplo, el ID del producto es `prod-example123456`.

   ```
   https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
   productCreated=prod-example123456&createdProductTitle=my-product
   ```

   > **Note**
   >
   > Copie la dirección URL de su producto antes de salir de la página. Una vez que salga de esta página, debe utilizar la CLI para obtener el ID del producto.

   Transcurridos unos segundos, el producto aparecerá en la página Products (Productos). Puede que necesite actualizar el navegador para ver el producto en la lista.

## Paso 3: Crear la canalización

1. Para asignar un nombre a la canalización y seleccionar los parámetros para la canalización, haga lo siguiente:

   a. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

   b. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.

   c. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.

   d. En el Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba un nombre para su canalización.

   e. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.

f. En Función de servicio, seleccione Nueva función de servicio CodePipeline para poder crear una función de servicio en IAM.

g. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).

2. Para agregar una etapa de origen en la página Paso 3: agregación de la etapa de origen, realice lo siguiente:

   a. En Source provider (Proveedor de código fuente), elija AWS CodeCommit.

   b. En Repository name (Nombre de repositorio) y Branch name (Nombre de ramificación), escriba el repositorio y la ramificación que desea utilizar para su acción de código fuente.

   c. Elija Siguiente.

3. En Paso 4: agregar la etapa de compilación, elija Omitir la etapa de compilación y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

4. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

   Elija Siguiente.

5. En Paso 6: agregación de la etapa de implementación, lleve a cabo lo siguiente:

   a. En Deploy provider (Proveedor de implementación), elija AWS Service Catalog.

   b. Para la configuración de implementación, elija Enter deployment configuration (Especificar configuración de implementación).

   c. En ID de producto, pegue el ID de producto copiado de la consola de Service Catalog.

   d. En Template file, path (Ruta de archivo de plantilla), escriba la ruta relativa donde se almacena el archivo de plantilla.

   e. En Tipo de producto, elija la plantilla de CloudFormation .

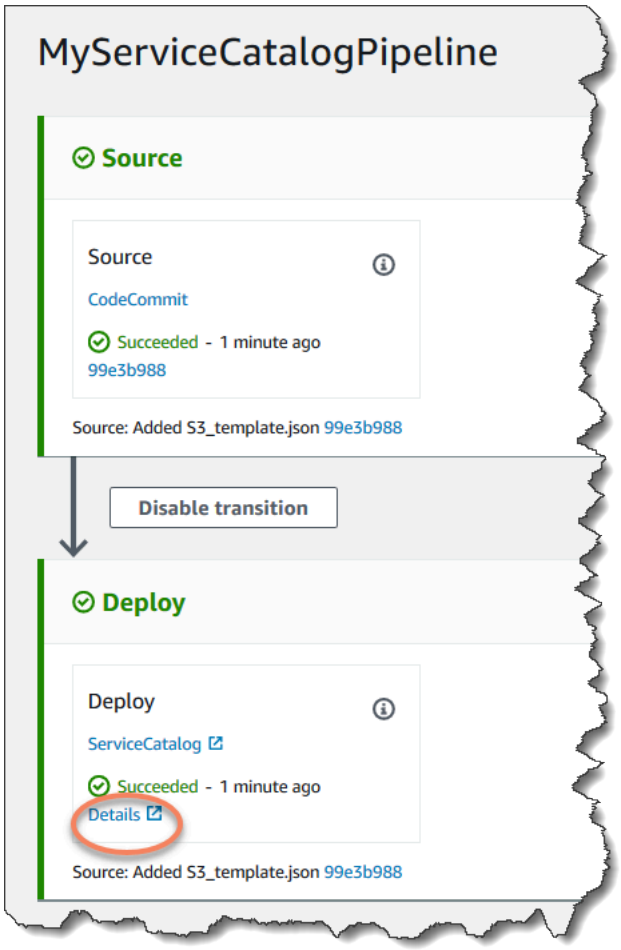
   f. En Nombre de versión de producto, escriba el nombre de la versión del producto que ha especificado en Service Catalog. Si desea implementar el cambio de plantilla en una nueva versión de producto, escriba un nombre de versión del producto que no se haya utilizado en cualquier versión de producto anterior en el mismo producto.

   g. Para Input artifact (Artefacto de entrada), elija el artefacto de entrada de código fuente.

   h. Elija Siguiente.

6. En Paso 7: revisión, revise la configuración de la canalización y después elija Crear.

7. Después de que la canalización se ejecute correctamente, en la etapa de implementación, elija Details (Detalles). Esto abre el producto en Service Catalog.




8. En su información del producto, seleccione el nombre de la versión para abrir la plantilla del producto. Vea la implementación de la plantilla.

## Paso 4: Enviar un cambio y verificar el producto en Service Catalog

1. Consulta tu canalización en la CodePipeline consola y, en la fase de origen, selecciona Detalles. El AWS CodeCommit repositorio de código fuente se abre en la consola. Elija Edit (Editar) y haga un cambio en el archivo (por ejemplo, en la descripción).

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

2. Confirme y envíe el cambio. La canalización comienza después de enviar el cambio. Cuando la ejecución de la canalización se haya completado, en la etapa de implementación, elija Detalles para abrir su producto en Service Catalog.

3. En su información del producto, seleccione el nuevo nombre de la versión para abrir la plantilla del producto. Vea el cambio de plantilla implementado.

## Opción 2: Realizar la implementación en Service Catalog con un archivo de configuración

En este ejemplo, subes el archivo de AWS CloudFormation plantilla de muestra para un bucket de S3 y, a continuación, creas tu producto en Service Catalog. También puede cargar un archivo de configuración distinto que especifique la configuración de implementación. A continuación, cree la canalización y especifique la ubicación del archivo de configuración.

### Paso 1: Cargar un archivo de plantilla de ejemplo en un repositorio de código fuente

1. Abra un editor de texto. Cree una plantilla de ejemplo pegando lo siguiente en el archivo. Guarde el archivo como `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"
    }
  }
}
```

Esta plantilla permite AWS CloudFormation crear un bucket de S3 que puede usar Service Catalog.

2. Cargue el archivo `S3_template.json` en el repositorio de AWS CodeCommit .

## Paso 2: Crear el archivo de configuración de implementación del producto

1. Abra un editor de texto. Cree el archivo de configuración para su producto. El archivo de configuración se usa para definir la implementación de Service Catalog parameters/preferences. Utilice este archivo al crear la canalización.

   Este ejemplo proporciona un `ProductVersionName` de "devops S3 v2" y un `ProductVersionDescription` de `MyProductVersionDescription`. Si desea implementar el cambio de plantilla en una nueva versión de producto, escriba un nombre de versión del producto que no se haya utilizado en ninguna versión de producto anterior en el mismo producto.

   Guarde el archivo como `sample_config.json`.

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

   Este archivo crea la información de versión del producto cada vez que se ejecute la canalización.

2. Cargue el archivo `sample_config.json` en el repositorio de AWS CodeCommit . Asegúrese de cargar este archivo en el repositorio de código fuente.

## Paso 3: Crear un producto en Service Catalog

1. Como administrador de TI, inicie sesión en la consola de Service Catalog., visite la página Productos y, a continuación, elija Cargar nuevo producto.
2. En la página Upload new product (Cargar nuevo producto), complete lo siguiente:

a. En Product name (Nombre del producto), introduzca el nombre que desea usar para el nuevo producto.

b. En Description (Descripción), escriba la descripción del catálogo de productos. Esta descripción aparece en el listado de productos para ayudar al usuario a elegir el producto correcto.

c. En Provided by (Proporcionado por), escriba el nombre del departamento de TI o del administrador.

d. Elija Siguiente.

3. (Opcional) En Enter support details (Introducir detalles de soporte), escriba la información de contacto de soporte del producto y elija Next (Siguiente).

4. En Version details (Detalles de la versión), realice lo siguiente:

a. Elija Upload a template file (Cargar un archivo de plantilla). Busque su archivo `S3_template.json` y cárguelo.

b. En Version title (Título de versión), escriba el nombre de la versión del producto (por ejemplo, "devops S3 v2").

c. En Description (Descripción), escriba detalles que distingan esta versión de otras versiones.

d. Elija Siguiente.

5. En la página Review (Revisar), compruebe que la información es correcta y, a continuación, elija Confirm and upload (Confirmar y cargar).

6. En la página Products (Productos), en el navegador, copie la URL de su nuevo producto. Contiene el ID del producto. Copie y conserve este ID de producto. Lo usará al crear la canalización en CodePipeline.

   A continuación se muestra la dirección URL de un producto llamado `my-product`. Para extraer el ID del producto, copie el valor entre el signo igual (=) y el signo ampersand (&). En este ejemplo, el ID del producto es `prod-example123456`.

```
https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
productCreated=prod-example123456&createdProductTitle=my-product
```

> **Note**
>
> Copie la dirección URL de su producto antes de salir de la página. Una vez que salga de esta página, debe utilizar la CLI para obtener el ID del producto.

Transcurridos unos segundos, el producto aparecerá en la página Products (Productos). Puede que necesite actualizar el navegador para ver el producto en la lista.

## Paso 4: Crear la canalización

1. Para asignar un nombre a la canalización y seleccionar los parámetros para la canalización, haga lo siguiente:
   a. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
   b. Elija Empezar. Elija Crear canalización y, a continuación, escriba un nombre para su canalización.
   c. En Rol de servicio, elija Nuevo rol de servicio CodePipeline para poder crear un rol de servicio en IAM.
   d. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
2. Para añadir una etapa de código fuente, haga lo siguiente:
   a. En Source provider (Proveedor de código fuente), elija AWS CodeCommit.
   b. En Repository name (Nombre de repositorio) y Branch name (Nombre de ramificación), escriba el repositorio y la ramificación que desea utilizar para su acción de código fuente.
   c. Elija Siguiente.
3. En Add build stage (Añadir etapa de compilación), elija Skip build stage (Omitir etapa de compilación) y, a continuación, acepte el mensaje de advertencia eligiendo Skip (Omitir) una vez más.
4. En Add deploy stage (Añadir etapa de implementación), lleve a cabo lo siguiente:
   a. En Deploy provider (Proveedor de implementación), elija AWS Service Catalog.
   b. Elija Use configuration file (Usar un archivo de configuración).

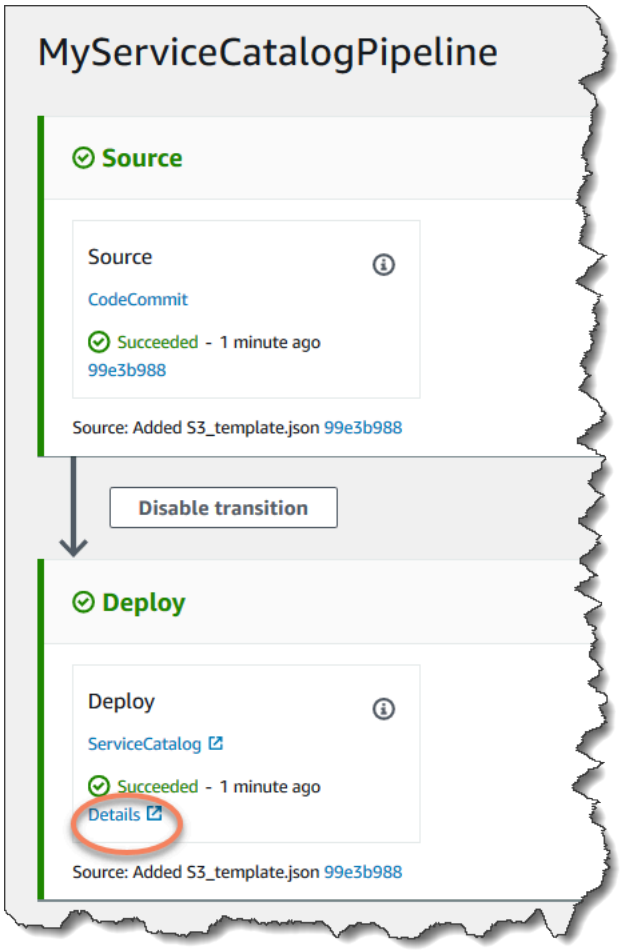
c. En ID de producto, pegue el ID de producto copiado de la consola de Service Catalog.

d. En Configuration file path (Ruta de archivo de configuración), escriba la ruta del archivo de configuración en el repositorio.

e. Elija Siguiente.

5. En Review (Revisar), revise la configuración de la canalización y después elija Create (Crear).

6. Después de que la canalización se ejecute correctamente, en su etapa de implementación, elija Detalles para abrir el producto en Service Catalog.




7. En su información del producto, seleccione el nombre de la versión para abrir la plantilla del producto. Vea la implementación de la plantilla.

## Paso 5: Enviar un cambio y verificar el producto en Service Catalog

1. Consulta tu canalización en la CodePipeline consola y, en la fase de origen, selecciona Detalles. El AWS CodeCommit repositorio de código fuente se abre en la consola. Elija Edit (Editar) y, a continuación, haga un cambio en el archivo (por ejemplo, en la descripción).

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

2. Confirme y envíe el cambio. La canalización comienza después de enviar el cambio. Cuando la ejecución de la canalización se haya completado, en la etapa de implementación, elija Detalles para abrir su producto en Service Catalog.
3. En su información del producto, seleccione el nuevo nombre de la versión para abrir la plantilla del producto. Vea el cambio de plantilla implementado.

# Tutorial: Crea una canalización con AWS CloudFormation

Los ejemplos proporcionan plantillas de ejemplo que te permiten AWS CloudFormation crear una canalización que implemente tu aplicación en tus instancias cada vez que cambie el código fuente. La plantilla de ejemplo crea una canalización que se puede visualizar en AWS CodePipeline. La canalización detecta la llegada de un cambio guardado a través de Amazon CloudWatch Events.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará un depósito de artefactos de S3 proporcionado por el cliente CodePipeline para los artefactos. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

Temas

- Ejemplo 1: Crear un AWS CodeCommit canalización con AWS CloudFormation
- Ejemplo 2: Cree una canalización de Amazon S3 con AWS CloudFormation

## Ejemplo 1: Crear un AWS CodeCommit canalización con AWS CloudFormation

En este tutorial se muestra cómo usar la AWS CloudFormation consola para crear una infraestructura que incluya una canalización conectada a un repositorio de CodeCommit origen. En este tutorial, utilizarás el archivo de plantilla de ejemplo que se proporciona para crear tu pila de recursos, que incluye tu almacén de artefactos, canalizaciones y recursos de detección de cambios, como la regla

de Amazon Events. CloudWatch Después de crear tu pila de recursos AWS CloudFormation, puedes ver tu canalización en la consola. AWS CodePipeline La canalización es una canalización de dos etapas con una etapa CodeCommit de origen y una etapa de CodeDeploy implementación.

Requisitos previos:

Debe haber creado los siguientes recursos para usarlos con la plantilla de AWS CloudFormation ejemplo:

- Debe haber creado un repositorio de origen. Puedes usar el AWS CodeCommit repositorio en el que creasteTutorial: Crear una canalización simple (CodeCommit repositorio).
- Debe haber creado una CodeDeploy aplicación y un grupo de implementación. Puede utilizar los recursos de CodeDeploy creados en Tutorial: Crear una canalización simple (CodeCommit repositorio).
- Elige uno de estos enlaces para descargar el archivo de AWS CloudFormation plantilla de ejemplo para crear una canalización: YAML | JSON

  Descomprima el archivo y colóquelo en su equipo local.
- Descarga el archivo de aplicación Linux.zip de muestra SampleApp _.

Crea tu canalización en AWS CloudFormation

1. Descomprime los archivos de SampleApp _ Linux.zip y súbelos a tu AWS CodeCommit repositorio. Debe cargar los archivos descomprimidos en el directorio raíz del repositorio. Puede seguir las instrucciones en Paso 2: Agrega código de muestra a tu CodeCommit repositorio para enviar los archivos a su repositorio.
2. Abre la AWS CloudFormation consola y elige Create Stack. Elija Con nuevos recursos (estándar).
3. En Especificar plantilla, elija Cargar una plantilla. Seleccione Elegir un archivo y luego seleccione el archivo de plantilla desde el equipo local. Elija Siguiente.
4. En Stack Name (Nombre de pila), escriba el nombre de la canalización. Se muestran los parámetros especificados en la plantilla de muestra. Introduzca los siguientes parámetros:

   a. En ApplicationName, introduce el nombre de tu CodeDeploy aplicación.

   b. En BetaFleet, introduzca el nombre de su grupo de CodeDeploy implementación.

   c. En BranchName, introduce la rama del repositorio que deseas usar.

d. En RepositoryName, introduce el nombre de tu repositorio de CodeCommit origen.

5. Elija Siguiente. Acepte los valores predeterminados en la siguiente página y, a continuación, elija Next (Siguiente).
6. En Capacidades, seleccione Reconozco que AWS CloudFormation podría crear recursos de IAM y, a continuación, elija Create stack.
7. Una vez creada la pila, consulte la lista de eventos para comprobar si hay errores.

   Solución de problemas

   El usuario de IAM que está creando la canalización AWS CloudFormation puede necesitar permisos adicionales para crear recursos para la canalización. La política exige los siguientes permisos para poder AWS CloudFormation crear los recursos de Amazon CloudWatch Events necesarios para la CodeCommit canalización:

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

8. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

   En Canalizaciones, elija la canalización y luego Ver. El diagrama muestra las etapas de código fuente e implementación de la canalización.

> **Note**
>
> Para ver la canalización que se creó, busque la columna ID lógico en la pestaña Recursos para su pila en CloudFormation. Anote el nombre de la canalización en la columna ID físico. En CodePipeline ella, puedes ver la canalización con el mismo identificador físico (nombre de la canalización) en la región en la que creaste la pila.

9. En el repositorio de origen, confirme y envíe un cambio. Sus recursos de detección de cambios recogen el cambio y se inicia la canalización.

## Ejemplo 2: Cree una canalización de Amazon S3 con AWS CloudFormation

En este tutorial se muestra cómo usar la AWS CloudFormation consola para crear una infraestructura que incluya una canalización conectada a un bucket de origen de Amazon S3. En este tutorial, utilizará el archivo de plantilla de ejemplo que se proporciona para crear su pila de recursos, que incluye el bucket de origen, el almacén de artefactos, la canalización y los recursos de detección de cambios, como las reglas y el registro de Amazon CloudWatch Events. CloudTrail Después de crear tu pila de recursos AWS CloudFormation, puedes ver tu canalización en la consola. AWS CodePipeline La canalización consta de dos etapas: una etapa de origen de Amazon S3 y una etapa de CodeDeploy implementación.

Requisitos previos:

Debe disponer de los siguientes recursos para utilizarlos con la plantilla de AWS CloudFormation muestra:

- Debe haber creado las instancias de Amazon EC2, donde instaló el CodeDeploy agente en las instancias. Debe haber creado una CodeDeploy aplicación y un grupo de implementación. Utilice Amazon EC2 y CodeDeploy los recursos en Tutorial: Crear una canalización simple (CodeCommit repositorio) los que creó.
- Elija los siguientes enlaces para descargar los archivos de AWS CloudFormation plantilla de muestra para crear una canalización con una fuente de Amazon S3:
  - Descargue la plantilla de ejemplo para su canalización: YAML | JSON
  - Descargue la plantilla de muestra para su CloudTrail bucket y su ruta: YAML | JSON samples/codepipeline-s3-cloudtrail-json.zip
  - Descomprima los archivos y colóquelos en su equipo local.
- Descarga la aplicación de ejemplo desde SampleApp _ Linux.zip.

  Guarde el archivo .zip en su equipo local. Debe cargar el archivo .zip una vez que se haya creado la pila.

Crea tu canalización en AWS CloudFormation

1. Abre la AWS CloudFormation consola y elige Create Stack. Elija Con nuevos recursos (estándar).
2. En Elegir una plantilla, elija Cargar una plantilla. Seleccione Elegir un archivo y luego seleccione el archivo de plantilla desde el equipo local. Elija Siguiente.
3. En Stack Name (Nombre de pila), escriba el nombre de la canalización. Se muestran los parámetros especificados en la plantilla de muestra. Introduzca los siguientes parámetros:

   a. En ApplicationName, introduce el nombre de tu CodeDeploy aplicación. Puede reemplazar el nombre predeterminado `DemoApplication`.

   b. En BetaFleet, introduzca el nombre de su grupo de CodeDeploy implementación. Puede reemplazar el nombre predeterminado `DemoFleet`.

   c. En SourceObjectKey, introduzca`SampleApp_Linux.zip`. Puede cargar este archivo en el bucket después de que la plantilla cree el bucket y la canalización.
4. Elija Siguiente. Acepte los valores predeterminados en la siguiente página y, a continuación, elija Next (Siguiente).
5. En Capacidades, seleccione Reconozco que eso AWS CloudFormation podría crear recursos de IAM y, a continuación, elija Create stack.
6. Una vez creada la pila, consulte la lista de eventos para comprobar si hay errores.

   Solución de problemas

   El usuario de IAM que está creando la canalización AWS CloudFormation puede necesitar permisos adicionales para crear recursos para la canalización. La política exige los siguientes permisos para poder crear los recursos de Amazon CloudWatch Events necesarios para la canalización de Amazon S3: AWS CloudFormation

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

```
    "Resource": "resource_ARN"
}
```

7. En CloudFormation la pestaña Recursos de su pila, consulte los recursos que se crearon para su pila.

> **Note**
>
> Para ver la canalización que se creó, busque la columna ID lógico en la pestaña Recursos para su pila en CloudFormation. Anote el nombre de la canalización en la columna ID físico. En CodePipeline, puedes ver la canalización con el mismo identificador físico (nombre de la canalización) en la región en la que creaste tu pila.

Elija el bucket de S3 con una etiqueta `sourcebucket` en el nombre, como `s3-cfn-codepipeline-sourcebucket-y04EXAMPLE.`. No elija el bucket del artefacto de la canalización.

El bucket de origen está vacío porque CloudFormation ha creado recientemente el recurso. Abra la consola de Amazon S3 y localice su bucket de `sourcebucket`. Elija Upload (Cargar) y siga las instrucciones para cargar el archivo .zip `SampleApp_Linux.zip`.

> **Note**
>
> Cuando Amazon S3 es el proveedor de origen de la canalización, debe cargar en el bucket todos los archivos de origen empaquetados como un solo archivo .zip. De lo contrario, la acción de origen dará error.

8. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

   En Canalizaciones, elija la canalización y luego Ver. El diagrama muestra las etapas de código fuente e implementación de la canalización.

9. Complete los pasos del siguiente procedimiento para crear sus recursos de AWS CloudTrail .

Crea tu AWS CloudTrail recursos en AWS CloudFormation

1. Abre la AWS CloudFormation consola y elige Create Stack.

2. En Elegir una plantilla, elija Subir una plantilla en Amazon S3. Elija Examinar y, a continuación, seleccione el archivo de plantilla para los AWS CloudTrail recursos de su equipo local. Elija Siguiente.
3. En Stack name (Nombre de pila), escriba un nombre para la pila de recursos. Se muestran los parámetros especificados en la plantilla de muestra. Introduzca los siguientes parámetros:
   - En SourceObjectKey, acepte el valor predeterminado para el archivo zip de la aplicación de ejemplo.
4. Elija Siguiente. Acepte los valores predeterminados en la siguiente página y, a continuación, elija Next (Siguiente).
5. En Capacidades, seleccione Reconozco que AWS CloudFormation podría crear recursos de IAM y, a continuación, elija Crear.
6. Una vez creada la pila, consulte la lista de eventos para comprobar si hay errores.

   La política exige los siguientes permisos para poder crear los AWS CloudFormation CloudTrail recursos necesarios para la canalización de Amazon S3:

```
{
     "Effect": "Allow",
     "Action": [
        "cloudtrail:CreateTrail",
        "cloudtrail:DeleteTrail",
        "cloudtrail:StartLogging",
        "cloudtrail:StopLogging",
        "cloudtrail:PutEventSelectors"
     ],
     "Resource": "resource_ARN"
}
```

7. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

   En Canalizaciones, elija la canalización y luego Ver. El diagrama muestra las etapas de código fuente e implementación de la canalización.
8. En el bucket de origen, confirme y envíe un cambio. Sus recursos de detección de cambios recogen el cambio y se inicia la canalización.

# Tutorial: Cree una canalización que utilice variables de AWS CloudFormation acciones de despliegue

En este tutorial, utilizarás la AWS CodePipeline consola para crear una canalización con una acción de despliegue. Cuando se ejecuta la canalización, la plantilla crea una pila y también crea un archivo `outputs`. Las salidas generadas por la plantilla de pila son las variables generadas por la AWS CloudFormation acción en CodePipeline.

En la acción en la que se crea la pila a partir de la plantilla, se designa un espacio de nombres variable. Acciones posteriores pueden consumir las variables producidas por el archivo `outputs`. En este ejemplo, se crea un conjunto de cambios en función de la `StackName` variable producida por la AWS CloudFormation acción. Después de una aprobación manual, ejecute el conjunto de cambios y, a continuación, cree una acción de eliminación de pila que elimine la pila en función de la variable `StackName`.

> **Important**
>
> Como parte de la creación de una canalización, para los artefactos se utilizará un depósito de artefactos de S3 proporcionado CodePipeline por el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

Temas


- Requisitos previos: crear un AWS CloudFormation un rol de servicio y un repositorio CodeCommit
- Paso 1: Descargue, edite y cargue la muestra AWS CloudFormation plantilla
- Paso 2: Crear la canalización
- Paso 3: Añade un CloudFormation acción de implementación para crear el conjunto de cambios
- Paso 4: Agregar una acción de aprobación manual
- Paso 5: Agregue una acción de CloudFormation implementación para ejecutar el conjunto de cambios
- Paso 6: Añade una acción CloudFormation de despliegue para eliminar la pila

# Requisitos previos: crear un AWS CloudFormation un rol de servicio y un repositorio CodeCommit

Debe disponer de lo siguiente:

- Un CodeCommit repositorio. Puedes usar el AWS CodeCommit repositorio en el que creasteTutorial: Crear una canalización simple (CodeCommit repositorio).
- En este ejemplo se crea una pila de Amazon DocumentDB a partir de una plantilla. Debe usar AWS Identity and Access Management (IAM) para crear un rol de AWS CloudFormation servicio con los siguientes permisos para Amazon DocumentDB.

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

# Paso 1: Descargue, edite y cargue la muestra AWS CloudFormation plantilla

Descarga el archivo de AWS CloudFormation plantilla de muestra y súbelo a tu CodeCommit repositorio.

1. Desplácese hasta la plantilla de ejemplo de su región. Por ejemplo, utilice la tabla de https://docs.aws.amazon.com/documentdb/latest/developerguide/quick_start_cfn.html#quick_start_cfn-launch_stack para elegir la región y descargar la plantilla. Descargue la plantilla para un clúster de Amazon DocumentDB. El nombre de archivo es `documentdb_full_stack.yaml`.
2. Descomprima el archivo `documentdb_full_stack.yaml` y ábralo en un editor de texto. Realice los siguientes cambios.

   a. Para este ejemplo, agregue el siguiente parámetro `Purpose:` a la sección `Parameters` de la plantilla.

```
Purpose:
  Type: String
  Default: testing
  AllowedValues:
    - testing
```

```
      - production
    Description: The purpose of this instance.
```

b. Para este ejemplo, agregue el siguiente resultado `StackName` a la sección `Outputs:` de la plantilla.

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

3. Sube el archivo de plantilla a tu AWS CodeCommit repositorio. Debe cargar el archivo de plantilla descomprimido y editado en el directorio raíz de su repositorio.

   Para usar la CodeCommit consola para subir tus archivos:

   a. Abre la CodeCommit consola y elige tu repositorio en la lista de repositorios.

   b. Elija Add file (Añadir archivo) y, a continuación, Upload file (Cargar archivo).

   c. Seleccione Choose file (Elegir archivo) y, a continuación, busque el archivo. Para confirmar el cambio, introduzca su nombre de usuario y la dirección de correo electrónico. Seleccione Confirmar cambios.

   Su archivo debe tener un aspecto similar a este en el nivel raíz de su repositorio:

```
documentdb_full_stack.yaml
```

## Paso 2: Crear la canalización

En esta sección, debe crear una canalización con las siguientes acciones:

- Una fase de origen con una CodeCommit acción en la que el artefacto de origen es el archivo de plantilla.
- Una etapa de despliegue con una acción CloudFormation de despliegue.

A cada acción de las fases de origen e implementación creadas por el asistente se le asigna un espacio de nombres variable, `SourceVariables` y `DeployVariables`, respectivamente. Debido a que las acciones tienen asignado un espacio de nombres, las variables configuradas en este ejemplo están disponibles para las acciones posteriores. Para obtener más información, consulte Referencia de variables.

Para crear una canalización con el asistente

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyCFNDeployPipeline`**.
5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.
6. En Service role (Rol de servicio), realice una de las operaciones siguientes:
   - Seleccione Nuevo rol de servicio para CodePipeline poder crear un rol de servicio en IAM.
   - Elija Existing service role (Rol de servicio existente) En Role name (Nombre del rol), elija el nombre del rol de servicio en la lista.
7. En Artifact store (Almacén de artefactos):
   a. Seleccione Ubicación predeterminada para utilizar con la canalización el almacén de artefactos predeterminado (por ejemplo, el bucket de Amazon S3 que se estableció como predeterminado) que esté en la región que seleccionó para la canalización.
   b. Elija Ubicación personalizada si ya dispone de un almacén de artefactos (por ejemplo, un bucket de artefactos de Amazon S3) en la misma región que la canalización.

> **Note**
>
> Este no es el bucket de origen para su código fuente. Este es el almacén de artefactos de la canalización. Cada canalización debe tener su propio almacén de artefactos independiente, como un bucket de S3. Al crear o editar una canalización, debes tener un depósito de artefactos en la región de canalización y un depósito de artefactos por AWS región en la que ejecutes una acción.
> Para obtener más información, consulte Artefactos de entrada y salida y CodePipeline referencia de estructura de tuberías.

Elija Siguiente.

8. En Paso 3: agregar la etapa de origen:

   a. En Source provider (Proveedor de código fuente), elija AWS CodeCommit.

   b. En Nombre del repositorio, elige el nombre del CodeCommit repositorio en el que creaste. Paso 1: Crea un CodeCommit repositorio

   c. En Nombre de ramificación, elija el nombre de la ramificación que incluye la última actualización del código.

   Tras seleccionar el nombre del repositorio y la sucursal, aparecerá la regla de Amazon CloudWatch Events que se va a crear para esta canalización.

   Elija Siguiente.

9. En Paso 4: agregar la etapa de compilación, elija Omitir la etapa de compilación y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

   Elija Siguiente.

10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.

11. En el paso 6: añadir la etapa de implementación:

    a. En Nombre de acción, elija Implementar. En Deploy provider (Proveedor de implementación), elija CloudFormation.

    b. En Modo acción, elija Crear o actualizar una pila.

    c. En Nombre de la pila, escriba un nombre para la pila. Este es el nombre de la pila que creará la plantilla.

    d. En Nombre del archivo de salida, escriba un nombre para el archivo de salida, como **`outputs`**. Este es el nombre del archivo que la acción creará después de que se cree la pila.

    e. Expanda Advanced (Avanzadas). En Sobrescritura de parámetros, especifique las invalidaciones de la plantilla como pares de clave-valor. Por ejemplo, esta plantilla requiere las siguientes invalidaciones.

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

Si no especifica las invalidaciones, la plantilla crea una pila con valores predeterminados.

f. Elija Siguiente.

g. En el paso 7: revisar, selecciona Crear canalización. Debería ver un diagrama con las etapas de la canalización. Permita que su canalización se ejecute. Su canalización de dos etapas está completa y lista para agregar las etapas adicionales.

## Paso 3: Añade un CloudFormation acción de implementación para crear el conjunto de cambios

Crea una próxima acción en proceso que te permita CloudFormation crear el conjunto de cambios antes de la acción de aprobación manual.

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

   En Canalizaciones, elija la canalización y luego Ver. El diagrama muestra las etapas de código fuente e implementación de la canalización.

2. Elija editar la canalización o continuar mostrando la canalización en el modo Editar.

3. Elija editar la etapa Implementación.

4. Agregue una acción de implementación que creará un conjunto de cambios para la pila que se creó en la acción anterior. Esta acción se añade después de la acción existente en la etapa.

   a. En Nombre de la acción, introduzca Change_Set. En Proveedor de acción, seleccione AWS CloudFormation .

   b. En Artefacto de entrada, elija SourceArtifact.

   c. En Action mode (Modo acción), elija Create or replace a change set (Crear o reemplazar un conjunto de cambios).

d. En Nombre de la pila, escriba la sintaxis de la variable como se muestra. Este es el nombre de la pila para la que se crea el conjunto de cambios, donde se asigna el espacio de nombres predeterminado `DeployVariables` a la acción.

```
#{DeployVariables.StackName}
```

e. En Nombre del conjunto de cambios, escriba el nombre del conjunto de cambios.

```
my-changeset
```

f. En Sobrescritura de parámetros, cambie el parámetro `Purpose` de `testing` a `production`.

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

g. Elija Listo para guardar la acción.

## Paso 4: Agregar una acción de aprobación manual

Cree una acción de aprobación manual en la canalización.

1. Elija editar la canalización o continuar mostrando la canalización en el modo Editar.
2. Elija editar la etapa Implementación.
3. Agregue una acción de aprobación manual después de la acción de implementación que crea el conjunto de cambios. Esta acción le permite verificar el conjunto de cambios del recurso creado CloudFormation antes de que la canalización ejecute el conjunto de cambios.

# Paso 5: Agregue una acción de CloudFormation implementación para ejecutar el conjunto de cambios

Cree una siguiente acción en su proceso que CloudFormation permita ejecutar el conjunto de cambios después de la acción de aprobación manual.

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

   En Canalizaciones, elija la canalización y luego Ver. El diagrama muestra las etapas de código fuente e implementación de la canalización.
2. Elija editar la canalización o continuar mostrando la canalización en el modo Editar.
3. Elija editar la etapa Implementación.
4. Agregue una acción de implementación que ejecutará el conjunto de cambios aprobado en la acción manual anterior:

   a. En Nombre de la acción, introduzca Execute_Change_Set. En Proveedor de acción, seleccione AWS CloudFormation.

   b. En Artefacto de entrada, elija SourceArtifact.

   c. En Action mode (Modo de acción), elija Execute a change set (Ejecutar un conjunto de cambios).

   d. En Nombre de la pila, escriba la sintaxis de la variable como se muestra. Este es el nombre de la pila para la que se crea el conjunto de cambios.

   ```
   #{DeployVariables.StackName}
   ```

   e. En Nombre del conjunto de cambios, escriba el nombre del conjunto de cambios que creó en la acción anterior.

   ```
   my-changeset
   ```

   f. Elija Listo para guardar la acción.

   g. Continúe la ejecución de la canalización.

## Paso 6: Añade una acción CloudFormation de despliegue para eliminar la pila

Crea una acción final en tu proceso que permita obtener el nombre de la pila CloudFormation a partir de la variable del archivo de resultados y eliminar la pila.

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

   En Canalizaciones, elija la canalización y luego Ver. El diagrama muestra las etapas de código fuente e implementación de la canalización.
2. Elija esta opción para editar la canalización.
3. Elija editar la etapa Implementación.
4. Agregue una acción de implementación que eliminará la pila:

   a. En Nombre de la acción, elige DeleteStack. En Deploy provider (Proveedor de implementación), elija CloudFormation.

   b. En Modo acción, elija Eliminar una pila.

   c. En Nombre de la pila, escriba la sintaxis de la variable como se muestra. Este es el nombre de la pila que la acción eliminará.

   d. Elija Listo para guardar la acción.

   e. Elija Guardar para guardar la canalización.

   La canalización se ejecuta cuando se guarda.

# Tutorial: implementación estándar de Amazon ECS con CodePipeline

Este tutorial le ayuda a crear un proceso de implementación continua (CD) completo e integral con Amazon ECS con CodePipeline.

> **Important**
>
> Como parte de la creación de una canalización en la consola, se utilizará un depósito de artefactos de S3 CodePipeline para los artefactos. (Es diferente del bucket que se usa para

una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

> **Note**
>
> Este tutorial es para la acción de implementación estándar de Amazon ECS para. CodePipeline Para ver un tutorial en el que se usa Amazon ECS para CodeDeploy blue/green implementar acciones CodePipeline, consulte Tutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR.

> **Note**
>
> Este tutorial trata sobre la acción de implementación estándar de Amazon ECS CodePipeline con una acción de código fuente. Para ver un tutorial en el que se utiliza la acción de implementación estándar de Amazon ECS junto con la acción de ECRBuildAndPublish compilación CodePipeline para insertar la imagen, consulte Tutorial: Cree y envíe una imagen de Docker a Amazon ECR con CodePipeline (tipo V2).

## Requisitos previos

Para poder usar este tutorial para crear su propia canalización de implementación continua debe tener instalados algunos recursos. Esto es lo que necesita para empezar:

> **Note**
>
> Todos estos recursos deben crearse en la misma AWS región.

- Un repositorio de control de código fuente (se usa en este tutorial CodeCommit) con el código fuente de tu aplicación y tu archivo Dockerfile. Para obtener más información, consulta Cómo crear un CodeCommit repositorio en la Guía del AWS CodeCommit usuario.
- Un repositorio de imágenes de Docker (este tutorial utiliza Amazon ECR) que contenga una imagen que haya creado desde el origen de Dockerfile y de la aplicación. Para obtener más

información, consulte Creación de un repositorio e Inserción de una imagen en la Guía del usuario de Amazon Elastic Container Registry.

- Una definición de tarea de Amazon ECS que haga referencia a la imagen de Docker alojada en su repositorio de imágenes. Para obtener más información, consulte Creación de una definición de tarea en la Guía para desarrolladores de Amazon Elastic Container Service.

> **Important**
>
> La acción de implementación estándar de Amazon ECS para CodePipeline crea su propia revisión de la definición de la tarea en función de la revisión utilizada por el servicio Amazon ECS. Si crea nuevas revisiones para la definición de la tarea sin actualizar el servicio Amazon ECS, la acción de implementación ignorará esas revisiones.

A continuación, se muestra un ejemplo de definición de tarea utilizada en este tutorial. El valor que utiliza para `name` y `family` se utilizará en el siguiente paso del archivo de especificaciones de compilación.

```
{
  "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": []
}
```

- Un clúster de Amazon ECS que ejecute un servicio que utilice la definición de tarea mencionada anteriormente. Para obtener más información, consulte Creación de un clúster y Creación de un servicio en la Guía para desarrolladores de Amazon Elastic Container Service.

Una vez satisfechos estos requisitos previos, puede continuar con el tutorial y crear su canalización de implementación continua.

## Paso 1: Añadir un archivo de especificación de compilación a su repositorio de código fuente

Este tutorial sirve CodeBuild para crear su imagen de Docker y enviar la imagen a Amazon ECR. Añada un archivo `buildspec.yml` al repositorio del código fuente que indique a CodeBuild cómo hacerlo. La siguiente especificación de compilación de ejemplo hace lo siguiente:

- Pre-build etapa:
  - Inicie sesión en Amazon ECR.
  - Establece el URI del repositorio en la imagen de ECR y añade una etiqueta de imagen con los siete primeros caracteres del ID de confirmación de Git del código fuente.
- Etapa de compilación:
  - Crea la imagen de Docker y etiqueta la imagen como `latest` y con el ID de confirmación de Git.
- Post-build escenario:
  - Inserta la imagen en el repositorio de ECR con ambas etiquetas.

- Escribe un archivo denominado `imagedefinitions.json` en la raíz de la compilación con el nombre del contenedor del servicio de Amazon ECS y la imagen y la etiqueta. La etapa de implementación de la canalización de implementación continua utiliza esta información para crear una nueva revisión de la definición de tarea del servicio y, a continuación, actualiza el servicio para usar la nueva definición de tarea. El archivo `imagedefinitions.json` es necesario para el proceso de trabajo de ECS.

Pegue este texto de ejemplo para crear el archivo `buildspec.yml` y sustituya los valores de la imagen y la definición de la tarea. En este texto, se utiliza el ID de cuenta de ejemplo 111122223333.

```
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
```

La especificación de compilación se ha creado para el siguiente ejemplo de definición de tarea proporcionada en Requisitos previos, usada por el servicio de Amazon ECS para este tutorial. El valor de `REPOSITORY_URI` se corresponde con el repositorio `image` (sin etiquetas de imagen) y el valor *`hello-world`* situado cerca del final del archivo se corresponde con el nombre de contenedor de la definición de tarea.

Para añadir un **`buildspec.yml`** archivo a su repositorio de origen

1. Abra un editor de texto y copie y pegue la especificación de compilación anterior en un nuevo archivo.
2. Sustituya el valor de `REPOSITORY_URI` (*`012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world`*) por el URI de su repositorio de Amazon ECR (sin etiquetas de imagen) para la imagen de Docker. Sustituya *`hello-world`* por el nombre de contenedor de la definición de tarea del servicio que hace referencia a la imagen de Docker.
3. Confirme la operación e inserte el archivo `buildspec.yml` en el repositorio de código fuente.
   a. Añada el archivo.

   ```
   git add .
   ```

   b. Valide el cambio con.

   ```
   git commit -m "Adding build specification."
   ```

   c. Envíe la confirmación.

   ```
   git push
   ```

## Paso 2: Crear la canalización de implementación continua

Utilice el CodePipeline asistente para crear las etapas de la canalización y conectar el repositorio de origen al servicio de ECS.

Para crear la canalización

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página Welcome, elija Create pipeline.

Si es la primera vez que la usas CodePipeline, aparecerá una página introductoria en lugar de la página de bienvenida. Seleccione Get Started Now.

3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En el paso 2: elección de la configuración de la canalización, en Nombre de la canalización, escriba un nombre para su canalización. En este tutorial, el nombre de la canalización es hello-world.
5. En Tipo de canalización, mantenga la selección predeterminada en V2. Los tipos de canalización difieren en características y precio. Para obtener más información, consulte Tipos de canalización. Elija Siguiente.
6. En la página Paso 3: Añadir fase de código fuente, en Proveedor de código fuente, seleccione AWS CodeCommit.
   a. En Repository name (Nombre del repositorio), elija el nombre del repositorio de CodeCommit que desea utilizar como ubicación de origen para la canalización.
   b. En Branch name (Nombre de ramificación), elija la ramificación que desea usar y seleccione Next (Siguiente).
7. En la página Paso 4: Agregar etapa de compilación, en Proveedor de compilación AWS CodeBuild, elija y, a continuación, elija Crear proyecto.
   a. En Project name, elija un nombre exclusivo para su proyecto de compilación. En este tutorial, el nombre del proyecto es hello-world.
   b. En Environment image (Imagen de entorno), elija Managed image (Imagen administrada).
   c. En Operating system (Sistema operativo), elija Amazon Linux 2.
   d. En Runtime(s) (Tiempo de ejecución), elija Standard (Estándar).
   e. Para Imagen, elija **`aws/codebuild/amazonlinux2-x86_64-standard:3.0`**.
   f. En Image version (Versión de imagen) y Environment type (Tipo de entorno), utilice los valores predeterminados.
   g. Seleccione Enable this flag if you want to build Docker images or want your builds to get elevated privileges (Habilite este indicador si desea compilar imágenes de Docker o que sus compilaciones tengan privilegios elevados).
   h. Deselecciona los CloudWatch registros. Puede que tengas que expandir Avanzado.
   i. Elige Continuar hasta. CodePipeline

j. Elija Siguiente.

> **Note**
>
> El asistente crea un rol de CodeBuild servicio para tu proyecto de compilación, denominado codebuild- ***build-project-name*** -service-role. Anote este nombre de rol, ya que le añadirá los permisos de Amazon ECR más adelante.

8. En la página Paso 5: Agregar la fase de implementación, para el proveedor de implementación, elija Amazon ECS.

   a. En Nombre del clúster, elija el clúster de en el que se ejecuta el servicio. En este tutorial, el clúster es default.

   b. En Service name (Nombre de servicio), elija el servicio que desea actualizar y seleccione Next (Siguiente). En este tutorial, el nombre del servicio es hello-world.

9. En la página Step 6: Review, revise la configuración de la canalización y elija Create pipeline para crear la canalización.

> **Note**
>
> Ahora que se ha creado la canalización, intenta ejecutarse a través de las diferentes etapas de canalización. Sin embargo, el CodeBuild rol predeterminado creado por el asistente no tiene permisos para ejecutar todos los comandos contenidos en el `buildspec.yml` archivo, por lo que se produce un error en la fase de creación. En la siguiente sección se añaden los permisos para la etapa de compilación.

## Paso 3: Añada los permisos de Amazon ECR al rol CodeBuild

El CodePipeline asistente creó un rol de IAM para el proyecto de compilación, denominado CodeBuild codebuild- -service-role. ***build-project-name*** En este tutorial, el nombre es codebuild-hello-world-service-role. Dado que el archivo `buildspec.yml` realiza llamadas a operaciones de la API de Amazon ECR, el rol debe tener una política que conceda permisos para realizar estas llamadas a Amazon ECR. El siguiente procedimiento le ayuda a asociar los permisos adecuados al rol.

Para añadir los permisos de Amazon ECR al rol CodeBuild

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación izquierdo, elija Roles.
3. En el cuadro de búsqueda, escriba codebuild- y elija el rol que creó el CodePipeline asistente. En este tutorial, el nombre del rol es codebuild-hello-world-service-role.
4. En la página Summary (Resumen), elija Attach policies (Asociar políticas).
5. Seleccione la casilla situada a la izquierda de la AmazonEC2ContainerRegistryPowerUser política y elija Adjuntar política.

## Paso 4: Probar la canalización

Tu canalización debe tener todo lo necesario para ejecutar una implementación AWS continua nativa de extremo a extremo. Ahora, pruebe su funcionalidad enviando un cambio de código al repositorio de código fuente.

Para probar la canalización

1. Realice una modificación del código en el repositorio de código fuente configurado, valide y envíe el cambio.
2. Abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
3. Seleccione su canalización de la lista.
4. Vea el progreso en la canalización a través de sus etapas. Su canalización debería completarse y el servicio de Amazon ECS debería ejecutar la imagen de Docker que se creó a partir de la modificación del código.

# Tutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR

En este tutorial, configurará una canalización para implementar aplicaciones de contenedor mediante una blue/green implementación que admita imágenes de Docker. AWS CodePipeline En una blue/green implementación, puedes lanzar la nueva versión de tu aplicación junto con la versión anterior y probar la nueva versión antes de redirigir el tráfico. También puede monitorizar el proceso de implementación y revertirlo rápidamente si hay algún problema.

> **Important**
>
> Como parte de la creación de una canalización, para los artefactos se utilizará un depósito de artefactos de S3 proporcionado por CodePipeline el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

> **Note**
>
> Este tutorial está dirigido a la acción de CodeDeploy blue/green implementación de Amazon ECS. CodePipeline Para ver un tutorial que utiliza la acción de implementación estándar de Amazon ECS en CodePipeline, consulteTutorial: implementación estándar de Amazon ECS con CodePipeline.

El proceso completado detecta los cambios en la imagen, que se almacena en un repositorio de imágenes como Amazon ECR y se utiliza CodeDeploy para enrutar e implementar el tráfico a un balanceador de cargas y un clúster de Amazon ECS. CodeDeploy utiliza un detector para redirigir el tráfico al puerto del contenedor actualizado especificado en el archivo. AppSpec Para obtener información sobre cómo se utilizan el balanceador de cargas, el detector de producción, los grupos objetivo y la aplicación de Amazon ECS en una blue/green implementación, consulte el tutorial: Implementación de un servicio de Amazon ECS.

La canalización también está configurada para usar una ubicación de origen, por ejemplo CodeCommit, donde se almacena la definición de la tarea de Amazon ECS. En este tutorial, configurará cada uno de estos AWS recursos y, a continuación, creará su canalización con etapas que contengan acciones para cada recurso.

La canalización de entrega continua compilará e implementará automáticamente las imágenes de contenedor cada vez que cambie el código fuente o se cargue una nueva imagen base en Amazon ECR.

Este flujo utiliza los siguientes artefactos:

- Un archivo de imagen de Docker que especifica el nombre del contenedor y el URI de su repositorio de imágenes de Amazon ECR.

- Una definición de tareas de Amazon ECS que muestra el nombre de la imagen de Docker, el nombre del contenedor, el nombre del servicio de Amazon ECS y la configuración del equilibrador de carga.
- Un CodeDeploy AppSpec archivo que especifica el nombre del archivo de definición de tareas de Amazon ECS, el nombre del contenedor de la aplicación actualizada y el puerto del contenedor al que se CodeDeploy redirige el tráfico de producción. También puede especificar la configuración de red opcional y funciones de Lambda que puede ejecutar durante enlaces de eventos del ciclo de vida de implementación.

> **Note**
>
> Cuando se confirma un cambio en el repositorio de imágenes de Amazon ECR, la acción de origen de la canalización crea un archivo `imageDetail.json` para dicha confirmación. Para obtener más información sobre el archivo `imageDetail.json`, consulte Detail.json archivo de imagen para las acciones de blue/green implementación de Amazon ECS.

Al crear o editar la canalización y actualizar o especificar artefactos de código fuente para la etapa de implementación, asegúrese de que apunte a los artefactos de código fuente con el nombre y la versión más recientes que desee utilizar. Después de configurar la canalización, a medida que realiza cambios en su imagen o definición de tareas, es posible que tenga que actualizar los archivos de artefactos de código fuente en sus repositorios y, a continuación, editar la etapa de implementación en la canalización.

Temas


- Requisitos previos
- Paso 1: Crear una imagen y enviarla al repositorio de Amazon ECR
- Paso 2: Cree los archivos AppSpec fuente y de definición de tareas y envíelos a un repositorio CodeCommit
- Paso 3: Crear el equilibrador de carga de aplicaciones y los grupos de destino
- Paso 4: Crear un clúster y un servicio de Amazon ECS
- Paso 5: Cree su grupo de CodeDeploy aplicaciones e implementaciones (plataforma informática ECS)
- Paso 6: Crear la canalización
- Paso 7: Realizar un cambio en la canalización y verificar la implementación

## Requisitos previos

Debe de haber creado los siguientes recursos:

- UN CodeCommit REPOSITORIO. Puedes usar el AWS CodeCommit repositorio en el que creasteTutorial: Crear una canalización simple (CodeCommit repositorio).
- Lance una instancia de Linux de Amazon EC2 e instale Docker para crear una imagen, tal y como se muestra en este tutorial. Puede omitir este requisito previo si ya dispone de una imagen que desea utilizar.

## Paso 1: Crear una imagen y enviarla al repositorio de Amazon ECR

En esta sección, utiliza Docker para crear una imagen y, a continuación, utiliza el AWS CLI para crear un repositorio de Amazon ECR y enviar la imagen al repositorio.

> **Note**
>
> Puede omitir este paso si ya dispone de una imagen que desea utilizar.

Para crear una imagen

1. Inicie sesión en la instancia de Linux en la que se ha instalado Docker.

   Despliegue una imagen de `nginx`. Este comando proporciona la imagen de `nginx:latest`:

   ```
   docker pull nginx
   ```

2. Ejecute docker images. Debería ver la imagen en la lista.

   ```
   docker images
   ```

Para crear un repositorio de Amazon ECR y enviar la imagen

1. Cree un repositorio de Amazon ECR para almacenar la imagen . Anote el `repositoryUri` en la salida.

   ```
   aws ecr create-repository --repository-name nginx
   ```

Salida:

```
{
    "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. Etiquete la imagen con el valor de `repositoryUri` del paso anterior.

```
docker tag nginx:latest aws_account_id.dkr.ecr.us-east-1.amazonaws.com/nginx:latest
```

3. Ejecute el comando aws ecr get-login-password, tal como se muestra en este ejemplo de la región de `us-west-2` y el ID de cuenta 111122223333.

```
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. Envíe la imagen a Amazon ECR utilizando el `repositoryUri` del paso anterior.

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

## Paso 2: Cree los archivos AppSpec fuente y de definición de tareas y envíelos a un repositorio CodeCommit

En esta sección, debe crear un archivo JSON de definición de tareas y registrarlo en Amazon ECS. A continuación, creas un AppSpec archivo para tu cliente de Git CodeDeploy y lo utilizas para subir los archivos a tu CodeCommit repositorio.

Para crear una definición de tareas para la imagen

1. Cree un archivo denominado `taskdef.json` con el siguiente contenido. En `image`, introduzca el nombre de su imagen, por ejemplo nginx. Este valor se actualiza cuando se ejecuta la canalización.

> **Note**
>
> Asegúrese de que el rol de ejecución especificado en la definición de la tarea contiene `AmazonECSTaskExecutionRolePolicy`. Para obtener más información, consulte Rol de IAM de ejecución de tareas de Amazon ECS en la Guía para desarrolladores de Amazon ECS.

```
{
    "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. Registre la definición de tareas con el archivo `taskdef.json`.

```
aws ecs register-task-definition --cli-input-json file://taskdef.json
```

3. Después de registrar la definición de tareas, edite el archivo para eliminar el nombre de la imagen e incluya el texto del marcador de posición <IMAGE1_NAME> en el campo de la imagen.

```
{
    "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"
}
```

Para crear un AppSpec archivo

- El AppSpec archivo se usa para CodeDeploy las implementaciones. El archivo, que incluye campos opcionales, utiliza este formato:

```
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"
```

Para obtener más información sobre el AppSpec archivo, incluidos ejemplos, consulte CodeDeploy AppSpec File Reference.

Cree un archivo denominado `appspec.yaml` con el siguiente contenido. Para `TaskDefinition`, no cambie el texto del marcador de posición `<TASK_DEFINITION>`. Este valor se actualiza cuando se ejecuta la canalización.

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: <TASK_DEFINITION>
        LoadBalancerInfo:
          ContainerName: "sample-website"
          ContainerPort: 80
```

Para subir archivos a tu CodeCommit repositorio

1. Sube o sube los archivos a tu CodeCommit repositorio. Estos archivos son el artefacto de código fuente creado por el asistente Create Pipeline (Crear canalización) para la acción de implementación en CodePipeline. Sus archivos deberían ser parecidos a estos en su directorio local:

```
/tmp
  |my-demo-repo
    |-- appspec.yaml
    |-- taskdef.json
```

2. Elija el método que desea usar para cargar los archivos:

   a. Para utilizar la línea de comandos de Git desde un repositorio clonado en el equipo local:

      i. Cambie los directorios a su repositorio local:

      ```
      (For Linux, macOS, or Unix) cd /tmp/my-demo-repo
      (For Windows) cd c:\temp\my-demo-repo
      ```

      ii. Ejecute el siguiente comando para preparar todos los archivos de una vez:

      ```
      git add -A
      ```

      iii. Ejecute el siguiente comando para confirmar los archivos con un mensaje de confirmación:

      ```
      git commit -m "Added task definition files"
      ```

      iv. Ejecuta el siguiente comando para subir los archivos de tu repositorio local a tu CodeCommit repositorio:

      ```
      git push
      ```

   b. Para usar la CodeCommit consola para subir tus archivos:

      i. Abre la CodeCommit consola y elige tu repositorio en la lista de repositorios.

      ii. Elija Add file (Añadir archivo) y, a continuación, Upload file (Cargar archivo).

      iii. Elija Choose file (Elegir archivo) y luego busque el archivo. Para confirmar el cambio, introduzca su nombre de usuario y la dirección de correo electrónico. Seleccione Confirmar cambios.

      iv. Repita este paso para cada archivo que desee cargar.

## Paso 3: Crear el equilibrador de carga de aplicaciones y los grupos de destino

En esta sección, va a crear un Equilibrador de carga de aplicación de Amazon EC2. Más tarde, utilizará los nombres de subred y los valores del grupo de destino que se crean con el equilibrador de carga al crear el servicio de Amazon ECS. Puede crear un equilibrador de carga de aplicación o

un equilibrador de carga de red. El equilibrador de carga debe utilizar una VPC con dos subredes públicas en diferentes zonas de disponibilidad. En estos pasos, confirmará la VPC predeterminada, creará un equilibrador de carga y, a continuación, creará dos grupos de destino para el equilibrador de carga. Para obtener más información, consulte Grupos de destino de los Network Load Balancers.

Para verificar la VPC predeterminada y las subredes públicas

1. Inicie sesión en Consola de administración de AWS y abra la consola de Amazon VPC en. https://console.aws.amazon.com/vpc/
2. Compruebe la VPC predeterminada que se va a utilizar. En el panel de navegación, elija Your VPCs (Sus VPC). Fíjese en qué VPC muestra Yes (Sí) en la columna Default VPC (VPC predeterminada). Esta es la VPC predeterminada. Contiene subredes predeterminadas para que pueda seleccionarlas.
3. Elija Subnets (Subredes). Elija dos subredes que muestren Yes (Sí) en la columna Default subnet (Subred predeterminada).

   > **Note**
   >
   > Anote los ID de subred. Los necesitará más adelante en este tutorial.

4. Elija las subredes y, a continuación, elija la pestaña Description (Descripción). Compruebe que las subredes que desea utilizar se encuentran en diferentes zonas de disponibilidad.
5. Elija las subredes y, a continuación, elija la pestaña Route Table (Tabla de ruteo). Para verificar que cada subred que desea utilizar es una subred pública, confirme que se haya incluido una fila con el gateway en la tabla de ruteo.

Para crear un Equilibrador de carga de aplicación de Amazon EC2

1. Inicie sesión en la consola Amazon EC2 Consola de administración de AWS y ábrala en. https://console.aws.amazon.com/ec2/
2. En el panel de navegación, seleccione Equilibradores de carga.
3. Elija Crear equilibrador de carga.
4. Elija Equilibrador de carga de aplicación y a continuación, Crear.
5. En Nombre, especifique el nombre del equilibrador de carga.
6. En Scheme (Esquema), elija Internet-facing (Expuesto a Internet).
7. En IP address type (Tipo de dirección IP), elija ipv4.

8. Configure dos puertos de agentes de escucha para su equilibrador de carga:

    a. En Protocolo de equilibrador de carga, elija HTTP. En Puerto del equilibrador de carga, escriba **80**.

    b. Elija Agregar oyente.

    c. En Protocolo de equilibrador de carga del segundo agente de escucha, elija HTTP. En Puerto del equilibrador de carga, escriba **8080**.

9. En Availability Zones (Zonas de disponibilidad), en VPC, elija la VPC predeterminada. A continuación, seleccione las dos subredes predeterminadas que desea utilizar.
10. Elija Next: Configure Security Settings (Siguiente: Establecer la configuración de seguridad).
11. Elija Next: Configure Security Groups (Siguiente: configurar grupos de seguridad).
12. Elija Select an existing security group (Seleccionar un grupo de seguridad existente) y tome nota del ID del grupo de seguridad.
13. Elija Next: Configure Routing (Siguiente: Configuración del enrutamiento).
14. En Target group (Grupo de destino), elija New target group (Nuevo grupo de destino) y configure su primer grupo de destino:

    a. En Name (Nombre), introduzca un nombre de grupo de destino (por ejemplo, **target-group-1**).

    b. En Target type (Tipo de destino), elija IP.

    c. En Protocol (Protocolo), elija HTTP. En Port (Puerto), introduzca **80**.

    d. Elija Next: Register Targets (Siguiente: Registrar destinos).

15. Elija Next: Review (Siguiente: análisis) y, a continuación, seleccione Create (Crear).

Para crear un segundo grupo de destino para el equilibrador de carga

1. Después de aprovisionar el equilibrador de carga, abra la consola de Amazon EC2. En el panel de navegación, elija Target Groups.
2. Elija Crear grupo de destino.
3. En Name (Nombre), introduzca un nombre de grupo de destino (por ejemplo, **target-group-2**).
4. En Target type (Tipo de destino), elija IP.
5. En Protocol (Protocolo), elija HTTP. En Port (Puerto), introduzca **8080**.
6. En VPC, elija la VPC predeterminada.

7. Seleccione Crear.

> **Note**
>
> Para que la implementación se ejecute, debe tener dos grupos de destino creados para el equilibrador de carga. Solo tiene que tomar nota del ARN de su primer grupo de destino. Este ARN se utiliza en el archivo JSON de `create-service` en el siguiente paso.

Para actualizar el equilibrador de carga para incluir el segundo grupo de destino

1. Abra la consola de Amazon EC2. En el panel de navegación, seleccione Equilibradores de carga.
2. Elija el equilibrador de carga y, a continuación, elija la pestaña Agentes de escucha. Elija el agente de escucha con el puerto 8080 y, a continuación, elija Edit (Editar).
3. Elija el icono del lápiz que aparece junto a Forward to (Reenviar a). Elija su segundo grupo de destino y, a continuación, elija la marca de verificación. Elija Update (Actualizar) para guardar las actualizaciones.

## Paso 4: Crear un clúster y un servicio de Amazon ECS

En esta sección, creará un clúster y un servicio de Amazon ECS que dirijan el tráfico CodeDeploy durante la implementación (a un clúster de Amazon ECS en lugar de a instancias de EC2). Para crear el servicio de Amazon ECS, debe utilizar los nombres de las subredes, el grupo de seguridad y el valor del grupo de destino que creó con el equilibrador de carga para crear el servicio.

> **Note**
>
> Al seguir estos pasos para crear un clúster de Amazon ECS, utiliza la plantilla de  clúster solo para  redes, que aprovisiona los contenedores de AWS Fargate. AWS Fargate es una tecnología que administra la infraestructura de instancias de contenedores por ti. No necesita seleccionar o crear de forma manual instancias de Amazon EC2 para su clúster de Amazon ECS.

Creación de un clúster de Amazon ECS

1. Abra la consola clásica de Amazon ECS en https://console.aws.amazon.com/ecs/.
2. En el panel de navegación, seleccione Clusters (Clústeres).
3. Elija Create cluster.
4. Elija la plantilla de clúster Solo redes que utiliza AWS Fargate y, a continuación, elija Paso siguiente.
5. Escriba un nombre de clúster en la página Configure cluster (Configurar el clúster), Puede añadir una etiqueta opcional al recurso. Seleccione Crear.

Para crear un servicio de Amazon ECS

Utilice el AWS CLI para crear su servicio en Amazon ECS.

1. Cree un archivo JSON y asígnele el nombre `create-service.json`. Pegue lo siguiente en el archivo JSON.

   En el campo `taskDefinition`, cuando se registra una definición de tarea en Amazon ECS, se le asigna una familia. Esto es similar a un nombre para las distintas versiones de la definición de tarea, que se especifican con un número de revisión. En este ejemplo, utilice `ecs-demo:1` con la familia y el número de revisión del archivo. Utilice los nombres de subred, el grupo de seguridad y el valor del grupo de destino que creó con el equilibrador de carga en el Paso 3: Crear el equilibrador de carga de aplicaciones y los grupos de destino.

   > **Note**
   >
   > Debe incluir el ARN del grupo de destino en este archivo. Abra la consola de Amazon EC2 y, desde el panel de navegación, en LOAD BALANCING (equilibrador de carga), elija Grupos de destino. Elija su primer grupo de destino. Copie el ARN desde la pestaña Description (Descripción).

```
{
    "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. Ejecute el comando create-service y especifique el archivo JSON:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

En este ejemplo, se crea un servicio denominado `my-service`.

> **Note**
>
> Este comando de ejemplo crea un servicio denominado my-service. Si ya tiene un servicio con este nombre, el comando devuelve un error.

```
aws ecs create-service --service-name my-service --cli-input-json file://create-
service.json
```

La salida devuelve los campos de descripción de su servicio.

3. Ejecute el comando describe-services para comprobar que se ha creado el servicio.

```
aws ecs describe-services --cluster cluster-name --services service-name
```

## Paso 5: Cree su grupo de CodeDeploy aplicaciones e implementaciones (plataforma informática ECS)

Al crear una CodeDeploy aplicación y un grupo de implementación para la plataforma informática de Amazon ECS, la aplicación se usa durante la implementación para hacer referencia al grupo de implementación, los grupos objetivo, los oyentes y el comportamiento de redireccionamiento del tráfico correctos.

Para crear una aplicación CodeDeploy

1. Abra la CodeDeploy consola y seleccione  Crear aplicación.
2. En Application name (Nombre de aplicación), escriba el nombre que desea utilizar.
3. En Compute platform (Plataforma de computación), elija Amazon ECS.
4. Elija Creación de aplicación.

Para crear un grupo CodeDeploy de implementación

1. En la página de la aplicación, en la pestaña Deployment groups (Grupos de implementaciones), elija Create deployment group (Crear grupo de implementaciones).
2. En Nombre de grupo de implementación, introduzca un nombre que describa el grupo de implementación.
3. En Rol de  servicio, elija un rol de servicio que otorgue CodeDeploy acceso a Amazon ECS. Para crear un nuevo rol de servicio, siga estos pasos:

   a. Abra la consola de IAM en https://console.aws.amazon.com/iam/.

   b. En el panel de la consola, elija Roles.

c. Elija Crear rol.

d. En Seleccionar el tipo de entidad de confianza, seleccione Servicio de AWS. En Elija un caso de uso, seleccione CodeDeploy. En Selecciona tu caso de uso, selecciona CodeDeploy - ECS. Elija Siguiente: permisos. La política administrada `AWSCodeDeployRoleForECS` ya está asociada al rol.

e. Elija Next: Tags (Siguiente: Etiquetas) y Next: Review (Siguiente: Revisar).

f. Especifique un nombre para el rol (por ejemplo, **`CodeDeployECSRole`**) y elija Create role (Crear rol).

4. En Configuración de entorno, elija el nombre del clúster y el nombre del servicio de Amazon ECS.
5. En Equilibradores de carga), seleccione el nombre del equilibrador de carga que sirve el tráfico a su servicio de Amazon ECS.
6. En Puerto del oyente de producción, elija el puerto y el protocolo del oyente que envía el tráfico de producción a su servicio de ECS de Amazon. En Test listener port (Puerto de agente de escucha de prueba), elija y el puerto y el protocolo del agente de escucha de prueba.
7. En Target group 1 name (Nombre de grupo de destino 1) y Target group 2 name (Nombre de grupo de destino 2), elija los grupos de destino utilizados para dirigir el tráfico durante su implementación. Asegúrese de que se trata de los grupos de destino que ha creado para el equilibrador de carga.
8. Elija Redirigir el tráfico inmediatamente para determinar cuánto tiempo debe transcurrir después de una implementación correcta antes de redirigir el tráfico hacia la tarea de Amazon ECS actualizada.
9. Elija Crear grupo de implementación.

## Paso 6: Crear la canalización

En esta sección, debe crear una canalización con las siguientes acciones:

- CodeCommit Acción en la que los artefactos de origen son la definición de la tarea y el AppSpec archivo.
- Una etapa de origen con una acción de origen de Amazon ECR donde el artefacto de código de fuente es el archivo de imagen.
- Una etapa de implementación con una acción de implementación de Amazon ECS en la que la implementación se ejecuta con una CodeDeploy aplicación y un grupo de implementación.

Para crear una canalización de dos etapas con el asistente

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyImagePipeline`**.
5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.
6. En Función de servicio, seleccione Nueva función de servicio CodePipeline para poder crear una función de servicio en IAM.
7. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
8. En Paso 3: agregar la etapa de origen, en Proveedor de origen, elija AWS CodeCommit. En Repository name (Nombre de repositorio), elija el nombre del repositorio de CodeCommit que ha creado en Paso 1: Crea un CodeCommit repositorio. En Nombre de ramificación, elija el nombre de la ramificación que incluye la última actualización del código.

   Elija Siguiente.
9. En Paso 4: agregar la etapa de compilación, elija Omitir la etapa de compilación y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más. Elija Siguiente.
10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.
11. En el paso 6: agregar la etapa de implementación:

    a. En Deploy provider, elija Amazon ECS (Blue/Green). En Application name (Nombre de aplicación), elija el nombre de la aplicación en la lista como, por ejemplo, `codedeployapp`. En Deployment group (Grupo de implementación), escriba o elija el nombre de un grupo de implementaciones en la lista como, por ejemplo, `codedeploydeplgroup`.

> **Note**
>
> El nombre "Deploy" es el nombre predeterminado de la etapa creada en el paso Step 4: Deploy (Paso 4: Implementar) y "Source" es el nombre que recibe la primera etapa de la canalización.

b. En la definición de tareas de Amazon ECS, elija SourceArtifact. En el campo, introduzca **`taskdef.json`**.

c. En AWS CodeDeploy AppSpec Archivo, elija SourceArtifact. En el campo, introduzca **`appspec.yaml`**.

> **Note**
>
> En este momento, no rellene ninguna información en Dynamically update task definition image (Actualizar dinámicamente imagen de definición de tareas).

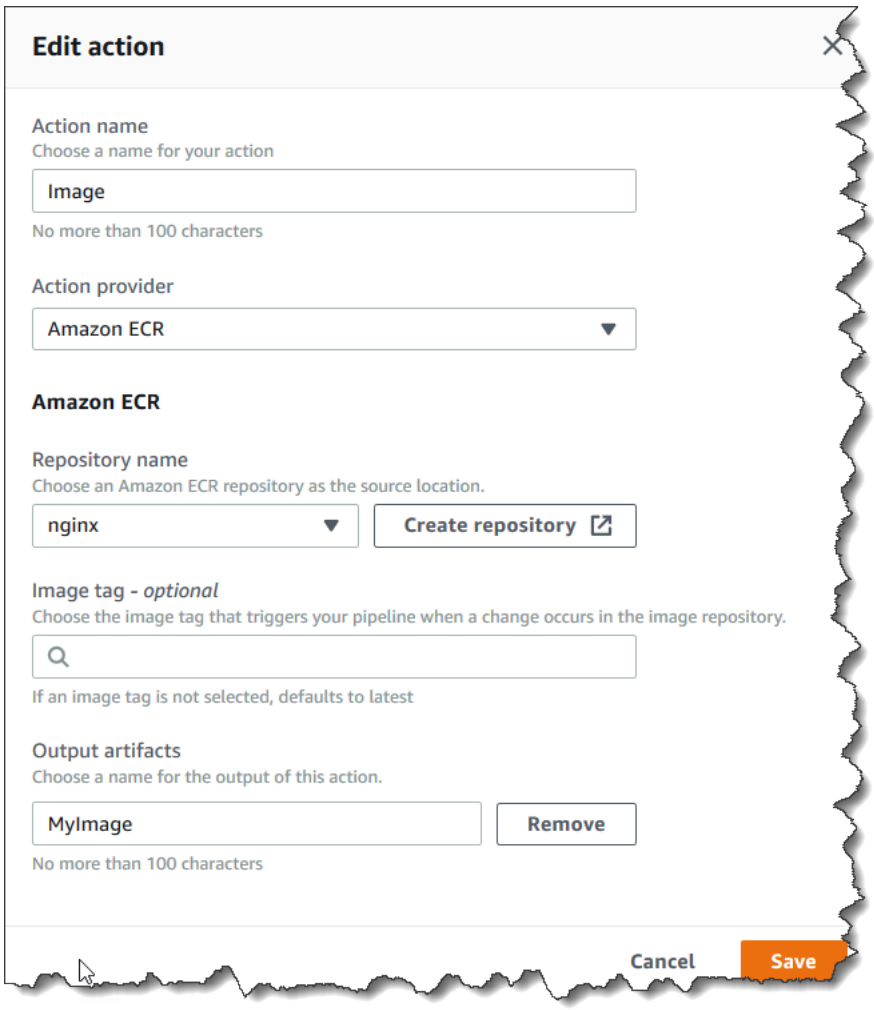
d. Elija Siguiente.

12. En el paso 7: Revisa, revisa la información y, a continuación, selecciona Crear canalización.

Para añadir una acción de origen de Amazon ECR a la canalización

Vea la canalización y añada una acción de origen de Amazon ECR a la canalización.

1. Elija la canalización. En la parte superior izquierda, elija Editar.
2. En la etapa de código fuente, elija Edit stage (Editar etapa).
3. Para añadir una acción paralela, selecciona + Añadir acción junto a la acción CodeCommit de origen.
4. En Nombre de la acción, escriba un nombre (por ejemplo, **`Image`**).
5. En Action provider (Proveedor de acción), elija Amazon ECR.




6. En Nombre de repositorio, elija el nombre de su repositorio de Amazon ECR.
7. En Image tag (Etiqueta de imagen), especifique el nombre de la imagen y la versión, si es diferente de la última.
8. En Output artifacts (Artefactos de salida), elija el valor predeterminado del artefacto de salida (por ejemplo, `MyImage`), que contiene el nombre de la imagen y la información del URI del repositorio que desea que utilice la siguiente etapa.
9. Elija Save (Guardar) en la pantalla de acciones. Elija Done (Listo) en la pantalla de etapas. Elija Save (Guardar) en la canalización. Un mensaje muestra la regla de Amazon CloudWatch Events que se va a crear para la acción fuente de Amazon ECR.

Para conectar los artefactos de código fuente con la acción de implementación

1. Seleccione  Editar  en la fase de implementación y elija el icono para editar la  acción de Amazon ECS (Blue/Green).

2. Desplácese hasta el final del panel. En Input artifacts (Artefactos de entrada), elija Add (Añadir). Añada el artefacto de origen desde su nuevo repositorio de Amazon ECR (por ejemplo, `MyImage`).
3. En Definición de tarea, elija y SourceArtifact, a continuación, verifique **`taskdef.json`** que se haya introducido.
4. En AWS CodeDeploy AppSpec Archivo, elija y SourceArtifact, a continuación, compruebe que **`appspec.yaml`** se ha introducido.
5. En Actualizar dinámicamente la imagen de definición de la tarea, en Introducir artefacto con URI de imagen, elija y MyImage, a continuación, introduzca el texto del marcador de posición que se utiliza en el `taskdef.json` archivo:. **`IMAGE1_NAME`** Seleccione Save.
6. En el AWS CodePipeline panel, elija Guardar cambio de canalización y, a continuación, seleccione Guardar cambio. Consulte la canalización actualizada.

   Después de crear esta canalización de ejemplo, la configuración de la acción de las entradas de la consola aparece en la estructura de la canalización tal y como se indica a continuación:

```
"configuration": {
  "AppSpecTemplateArtifact": "SourceArtifact",
  "AppSpecTemplatePath": "appspec.yaml",
  "TaskDefinitionTemplateArtifact": "SourceArtifact",
  "TaskDefinitionTemplatePath": "taskdef.json",
  "ApplicationName": "codedeployapp",
  "DeploymentGroupName": "codedeploydeplgroup",
  "Image1ArtifactName": "MyImage",
  "Image1ContainerName": "IMAGE1_NAME"
},
```

7. Para enviar los cambios y comenzar una compilación de canalización, seleccione Publicar modificación y, a continuación, Publicar.
8. Elige la acción de despliegue para verla CodeDeploy y ver el progreso del cambio de tráfico.

> **Note**
>
> Es posible que vea un paso de implementación que muestre un tiempo de espera opcional. De forma predeterminada, CodeDeploy espera una hora después de una implementación exitosa antes de terminar el conjunto de tareas original. Puede utilizar

este tiempo para revertir o terminar la tarea, pero la implementación se completa cuando finaliza el conjunto de tareas.

### Paso 7: Realizar un cambio en la canalización y verificar la implementación

Realice un cambio en la imagen y luego envíe ese cambio al repositorio de Amazon ECR. Esto desencadena la ejecución de la canalización. Compruebe que el cambio del código fuente de la imagen se ha implementado.

## Tutorial: Crear una canalización que implemente una habilidad de Amazon Alexa

En este tutorial, puede configurar una canalización que enviará constantemente la habilidad de Alexa mediante Alexa Skills Kit como proveedor de implementación en la etapa de implementación. La canalización completa detecta cambios en la habilidad cuando se realiza un cambio en los archivos de código fuente en el repositorio de código fuente. Esto hace que la canalización utilice Alexa Skills Kit para realizar la implementación en la etapa de desarrollo de la habilidad de Alexa.

**Important**

Como parte de la creación de una canalización, se utilizará un depósito de artefactos de S3 proporcionado por el cliente CodePipeline para los artefactos. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y fiable.

**Note**

Esta característica no está disponible en las regiones de Asia Pacífico (Hong Kong) ni Europa (Milán). Para utilizar otras acciones de implementación disponibles en esa región, consulte Integraciones de acciones de implementación.

Para crear una habilidad personalizada como una función Lambda, consulte Hospedar una habilidad personalizada como una función Lambda AWS. También puede crear una canalización que utilice

archivos fuente de Lambda y un CodeBuild proyecto para implementar cambios en Lambda que se ajusten a sus habilidades.

## Requisitos previos

Debe disponer de lo siguiente:

- Un CodeCommit repositorio. Puedes usar el AWS CodeCommit repositorio en el que lo creasteTutorial: Crear una canalización simple (CodeCommit repositorio).
- Una cuenta de desarrollador de Amazon. Esta cuenta es la propietaria de las habilidades de Alexa. Puede crear una cuenta de forma gratuita en Alexa Skills Kit.
- Una habilidad de Alexa. Puede crear una habilidad de ejemplo mediante el tutorial que describe cómo obtener código de muestra de habilidades personalizadas.
- Instale la CLI de ASK y configúrela utilizando `ask init` con sus credenciales de AWS . Consulte Install and initialize ASK CLI.

## Paso 1: Crear un perfil de seguridad de LWA para los servicios de desarrolladores de Alexa

En esta sección, debe crear un perfil de seguridad para utilizarlo con Login with Amazon (LWA). Si ya tiene un perfil, puede omitir este paso.

- Utilice los pasos que se indican en generate-lwa-tokens para crear un perfil de seguridad.
- Después de crear el perfil, anote el Client ID (ID de cliente) y el Client Secret (Secreto de cliente).
- Asegúrese de introducir la devolución permitida tal y URLs como se indica en las instrucciones. El comando URLs allow the ASK CLI redirija las solicitudes de token de actualización.

## Paso 2: Crea los archivos fuente de habilidades de Alexa y envíalos a tu CodeCommit repositorio

En esta sección, debe crear y enviar los archivos de código fuente de la habilidad de Alexa al repositorio que utiliza la canalización para la etapa de código fuente. Para la habilidad que ha creado en la consola para desarrolladores de Amazon, debe crear y enviar lo siguiente:

- Un archivo `skill.json`.

- Una carpeta `interactionModel/custom`

> **Note**
>
> Esta estructura de directorios cumple con los requisitos de formato del paquete de habilidades de Alexa Skills Kit, tal y como se describe en Skill package format. Si la estructura de directorios no utiliza el formato de paquete de habilidades correcto, los cambios no se implementan correctamente en la consola de Alexa Skills Kit.

Para crear los archivos de código fuente de la habilidad

1. Recupere el ID de habilidad de la consola para desarrolladores de Alexa Skills Kit. Utilice este comando:

   ```
   ask api list-skills
   ```

   Localice la habilidad por su nombre y a continuación, copie el ID asociado en el campo `skillId`.

2. Genere un archivo `skill.json` que contenga los detalles de la habilidad. Utilice este comando:

   ```
   ask api get-skill -s skill-ID > skill.json
   ```

3. (Opcional) Cree una carpeta `interactionModel/custom`.

   Utilice este comando para generar el archivo del modelo de interacción en la carpeta. Para la configuración regional, este tutorial utiliza en-US como configuración regional del nombre de archivo.

   ```
   ask api get-model --skill-id skill-ID --locale locale >
       ./interactionModel/custom/locale.json
   ```

Para enviar archivos a tu CodeCommit repositorio

1. Envía o sube los archivos a tu CodeCommit repositorio. Estos archivos son el artefacto de código fuente creado por el asistente Create Pipeline (Crear canalización) para la acción de implementación en AWS CodePipeline. Sus archivos deberían ser parecidos a estos en su directorio local:

```
skill.json
/interactionModel
  /custom
    |en-US.json
```

2. Elija el método que desea usar para cargar los archivos:

   a. Para utilizar la línea de comandos de Git desde un repositorio clonado en el equipo local:

      i. Ejecute el siguiente comando para preparar todos los archivos de una vez:

      ```
      git add -A
      ```

      ii. Ejecute el siguiente comando para confirmar los archivos con un mensaje de confirmación:

      ```
      git commit -m "Added Alexa skill files"
      ```

      iii. Ejecuta el siguiente comando para enviar los archivos de tu repositorio local a tu CodeCommit repositorio:

      ```
      git push
      ```

   b. Para usar la CodeCommit consola para cargar tus archivos:

      i. Abre la CodeCommit consola y elige tu repositorio en la lista de repositorios.

      ii. Elija Add file (Añadir archivo) y, a continuación, Upload file (Cargar archivo).

      iii. Elija Choose file (Elegir archivo) y luego busque el archivo. Para confirmar el cambio, introduzca su nombre de usuario y la dirección de correo electrónico. Seleccione Confirmar cambios.

      iv. Repita este paso para cada archivo que desee cargar.

## Paso 3: Utilizar los comandos de la CLI de ASK para crear un token de actualización

CodePipeline utiliza un token de actualización basado en el ID de cliente y el secreto de tu cuenta de desarrollador de Amazon para autorizar las acciones que realiza en tu nombre. En esta sección,

se utiliza la CLI de ASK para crear el token. Estas credenciales se utilizan al usar el asistente Create Pipeline (Crear canalización).

Para crear un token de actualización con las credenciales de una cuenta de desarrollador de Amazon

1. Utilice el siguiente comando :

```
ask util generate-lwa-tokens
```

2. Cuando se le solicite, introduzca el ID y el secreto del cliente, tal y como se muestra en este ejemplo:

```
? Please type in the client ID:
amzn1.application-client.example112233445566
? Please type in the client secret:
example112233445566
```

3. El navegador muestra la página de inicio de sesión. Inicie sesión con las credenciales de su cuenta de desarrollador de Amazon.
4. Vuelva a la pantalla de la línea de comandos. El token de acceso y el token de actualización se generan en la salida. Copie el token de actualización que se devuelven en la salida.

## Paso 4: Crear la canalización

En esta sección, debe crear una canalización con las siguientes acciones:

- Una fase de origen con una CodeCommit acción en la que los artefactos de origen son los archivos de habilidades de Alexa que respaldan tu habilidad.
- Una etapa de implementación con una acción de implementación de Alexa Skills Kit.

Para crear una canalización con el asistente

1. Inicia sesión en la CodePipeline consola Consola de administración de AWS y ábrela en http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Elija la AWS región en la que desea crear el proyecto y sus recursos. El tiempo de ejecución de la habilidad de Alexa está disponible solo en las siguientes regiones:

- Asia-Pacífico (Tokio)
- Europa (Irlanda)
- Este de EE. UU. (Norte de Virginia)
- Oeste de EE. UU. (Oregón)

3. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
4. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
5. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyAlexaPipeline`**.
6. CodePipeline proporciona canalizaciones de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios de CodePipeline, consulte Precios.
7. En Función de servicio, elija Nueva función de servicio CodePipeline para poder crear una función de servicio en IAM.
8. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
9. En Paso 3: agregar la etapa de origen, en Proveedor de origen, elija AWS CodeCommit. En Nombre del repositorio, elija el nombre del CodeCommit repositorio en Paso 1: Crea un CodeCommit repositorio el que lo creó. En Nombre de ramificación, elija el nombre de la ramificación que incluye la última actualización del código.

   Tras seleccionar el nombre y la sucursal del repositorio, aparecerá un mensaje con la regla de Amazon CloudWatch Events que se va a crear para esta canalización.

   Elija Siguiente.
10. En Paso 4: agregar la etapa de compilación, elija Omitir la etapa de compilación y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.
11. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.
12. En el paso 6: Añadir la etapa de despliegue:

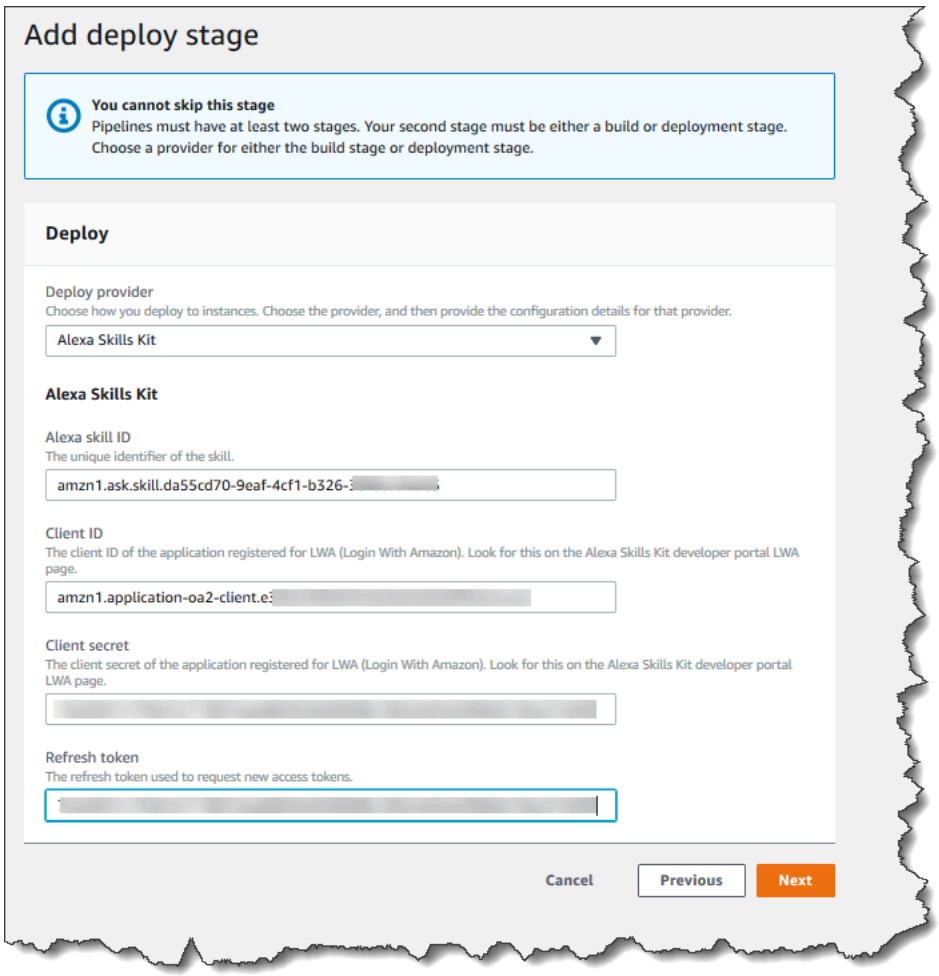
a. En Deploy provider (Proveedor de implementación), elija Alexa Skills Kit.

b. En Alexa skill ID (ID de habilidad de Alexa), especifique el ID de habilidad asignado a la habilidad en la consola para desarrolladores de Alexa Skills Kit.

c. En Client ID (ID de cliente), especifique el ID de la aplicación que ha registrado.

d. En Client secret (Secreto de cliente), especifique el secreto de cliente que eligió cuando se registró.

e. En Refresh token (Token de actualización), especifique el token que ha generado en el paso 3.




f. Elija Siguiente.

13. En el paso 7: Revisa, revisa la información y, a continuación, selecciona Crear canalización.

## Paso 5: Realizar un cambio en cualquier archivo de origen y verificar la implementación

Realice un cambio en la habilidad y, a continuación, envíe ese cambio al repositorio. Esto desencadena la ejecución de la canalización. Compruebe que la habilidad se actualiza en la consola para desarrolladores de Alexa Skills Kit.

# Tutorial: Crear una canalización que utilice Amazon S3 como proveedor de implementación

En este tutorial, va a configurar una canalización que enviará constantemente archivos utilizando Amazon S3 como proveedor de acciones de implementación en la etapa de implementación. La canalización completa detecta cambios cuando se realiza un cambio en los archivos de código fuente en el repositorio de código fuente. Esto hace que la canalización utilice Amazon S3 para implementar los archivos en el bucket. Cada vez que modifique o añada los archivos del sitio web en su ubicación de origen, la implementación creará el sitio web con los archivos más recientes.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará un depósito de artefactos de S3 proporcionado por el cliente CodePipeline para los artefactos. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y fiable.

> **Note**
>
> Incluso si elimina archivos del repositorio de origen, la acción de implementación de S3 no elimina los objetos de S3 correspondientes a los archivos eliminados.

En este tutorial, encontrará dos opciones:

- Puede crear una canalización que implemente un sitio web estático en un bucket público de S3. En este ejemplo, se crea una canalización con una acción de AWS CodeCommit origen y una

acción de despliegue de Amazon S3. Consulte Opción 1: Implementar los archivos de un sitio web estático en Amazon S3.

- Cree una canalización que compile el TypeScript código de muestra JavaScript y despliegue el artefacto de CodeBuild salida en su bucket de S3 para archivarlo. En este ejemplo, se crea una canalización con una acción de origen de Amazon S3, una acción de CodeBuild compilación y una acción de despliegue de Amazon S3. Consulte Opción 2: Implementar archivos compilados archivados en Amazon S3 desde un bucket de origen de S3.

> **Important**
>
> Muchas de las acciones que añada a la canalización en este procedimiento implican AWS recursos que debe crear antes de crear la canalización. AWS Los recursos para las acciones de origen siempre deben crearse en la misma AWS región en la que se creó la canalización. Por ejemplo, si creas tu canalización en la región EE.UU. Este (Ohio), tu CodeCommit repositorio debe estar en la región EE.UU. Este (Ohio).
>
> Puedes añadir acciones entre regiones al crear tu canalización. AWS los recursos para las acciones entre regiones deben estar en la misma AWS región en la que planeas ejecutar la acción. Para obtener más información, consulte Agregue una acción interregional en CodePipeline.

## Opción 1: Implementar los archivos de un sitio web estático en Amazon S3

En este ejemplo, descarga el archivo de plantilla de sitio web estático de ejemplo, carga los archivos en su AWS CodeCommit repositorio, crea su depósito y lo configura para el alojamiento. A continuación, utilice la AWS CodePipeline consola para crear su canalización y especificar una configuración de despliegue de Amazon S3.

### Requisitos previos

Debe disponer de lo siguiente:

- Un CodeCommit repositorio. Puedes usar el AWS CodeCommit repositorio en el que lo creasteTutorial: Crear una canalización simple (CodeCommit repositorio).
- Archivos de código fuente del sitio web estático. Utilice este enlace para descargar un ejemplo de sitio web estático. La descarga de sample-website.zip produce los siguientes archivos:
  - Un archivo `index.html`

- Un archivo `main.css`
- Un archivo `graphic.jpg`

- Un bucket de S3 configurado para alojar sitios web. Consulte Alojamiento de un sitio web estático en Amazon S3. No olvide que el bucket debe crearse en la misma región que la canalización.

> **Note**
>
> Para alojar un sitio web, el bucket debe tener acceso de lectura público, lo que concede acceso a todo el mundo. Salvo en el caso del alojamiento de sitios web, debe dejar la configuración de acceso predeterminada que bloquea el acceso público a los buckets de S3.

## Paso 1: Envía los archivos fuente a tu CodeCommit repositorio

En esta sección, debe enviar los archivos de código fuente al repositorio que utiliza la canalización para la etapa de código fuente.

Para enviar archivos a tu CodeCommit repositorio

1. Extraiga los archivos de ejemplo que ha descargado. No cargue el archivo ZIP en el repositorio.
2. Envía o sube los archivos a tu CodeCommit repositorio. Estos archivos son el artefacto de código fuente creado por el asistente Create Pipeline (Crear canalización) para la acción de implementación en CodePipeline. Sus archivos deberían ser parecidos a estos en su directorio local:

   ```
   index.html
   main.css
   graphic.jpg
   ```

3. Puedes usar Git o la CodeCommit consola para cargar tus archivos:

   a. Para utilizar la línea de comandos de Git desde un repositorio clonado en el equipo local:

      i. Ejecute el siguiente comando para preparar todos los archivos de una vez:

      ```
      git add -A
      ```

ii. Ejecute el siguiente comando para confirmar los archivos con un mensaje de confirmación:

```
git commit -m "Added static website files"
```

iii. Ejecuta el siguiente comando para enviar los archivos de tu repositorio local a tu CodeCommit repositorio:

```
git push
```

b. Para usar la CodeCommit consola para cargar tus archivos:

i. Abre la CodeCommit consola y elige tu repositorio en la lista de repositorios.

ii. Elija Add file (Añadir archivo) y, a continuación, Upload file (Cargar archivo).

iii. Seleccione Choose file (Elegir archivo) y, a continuación, busque el archivo. Para confirmar el cambio, introduzca su nombre de usuario y la dirección de correo electrónico. Seleccione Confirmar cambios.

iv. Repita este paso para cada archivo que desee cargar.

## Paso 2: Crear la canalización

En esta sección, debe crear una canalización con las siguientes acciones:

- Una etapa de origen con una CodeCommit acción en la que los artefactos de origen son los archivos de tu sitio web.
- Una etapa de implementación con una acción de implementación de Amazon S3.

Para crear una canalización con el asistente

1. Inicie sesión Consola de administración de AWS y abra la CodePipeline consola en http:// console.aws.amazon. com/codesuite/codepipeline/home.
2. En la página Bienvenido, Introducción o en la página Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyS3DeployPipeline`**.

5. En Tipo de canalización, seleccione V2. Para obtener más información, consulte Tipos de canalización. Elija Siguiente.
6. En Función de servicio, elija Nueva función de servicio CodePipeline para poder crear una función de servicio en IAM.
7. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
8. En Paso 3: agregar la etapa de origen, en Proveedor de origen, elija AWS CodeCommit. En Repository name (Nombre de repositorio), elija el nombre del repositorio de CodeCommit que ha creado en Paso 1: Crea un CodeCommit repositorio. En Nombre de ramificación, elija el nombre de la ramificación que incluye la última actualización del código. A menos que haya creado otra ramificación, solo estará disponible `main`.

   Tras seleccionar el nombre y la sucursal del repositorio, se muestra la regla de Amazon CloudWatch Events que se va a crear para esta canalización.

   Elija Siguiente.
9. En Paso 4: agregar la etapa de compilación, elija Omitir la etapa de compilación y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

   Elija Siguiente.
10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.
11. En el paso 6: Añadir la etapa de despliegue:

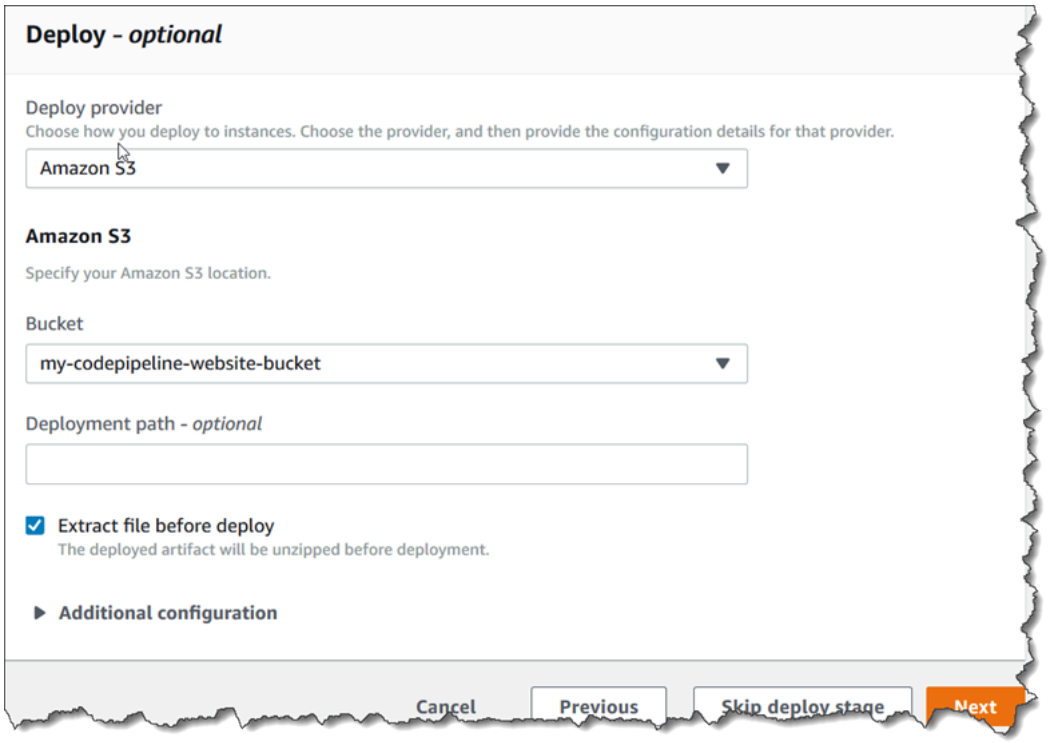
    a. En Deploy provider (Proveedor de implementación), elija Amazon S3.

    b. En Bucket, especifique el nombre del bucket público.

    c. Seleccione Extract file before deploy (Extraer el archivo antes de la implementación).

> **Note**
>
> Si no selecciona Extraer el archivo antes de la implementación, se produce un error en la implementación. Esto se debe a que la AWS CodeCommit acción de la canalización comprime los artefactos de origen y el archivo es un archivo ZIP.

Cuando se selecciona Extract file before deploy (Extraer el archivo antes de la implementación), se muestra Deployment path (Ruta de implementación). Especifique el nombre de la ruta que desea utilizar. Esto crea una estructura de carpetas en Amazon S3 en la que se extraen los archivos. Para este tutorial, deje en blanco este campo.




d. (Opcional) En Canned ACL puede aplicar un conjunto de garantías predefinidas, conocido como una ACL predefinida a los artefactos cargados.

e. (Opcional) En Cache control (Control del caché) introduzca los parámetros de caché. Puede establecerlo para controlar el comportamiento del caché para las solicitudes/respuestas. Para valores válidos, consulte el `Cache-Control`campo del encabezado para las operaciones HTTP

f. Elija Siguiente.

12. En el paso 7: Revisar, revise la información y, a continuación, seleccione Crear canalización.

13. Después de la canalización se ejecute correctamente, abra la consola de Amazon S3 y compruebe que los archivos aparecen en el bucket público, tal como se muestra a continuación:

```
index.html
main.css
graphic.jpg
```

14. Obtenga acceso al punto de enlace para probar el sitio web. El punto de enlace tiene este formato: `http://bucket-name.s3-website-region.amazonaws.com/`.

    Ejemplo de punto de enlace: `http://my-bucket.s3-website-us-west-2.amazonaws.com/`.

    Aparece la siguiente página web de muestra.

### Paso 3: Realizar un cambio en cualquier archivo de código fuente y verificar la implementación

Realice un cambio en los archivos código fuente y, a continuación, envíe ese cambio al repositorio. Esto desencadena la ejecución de la canalización. Compruebe que el sitio web se actualiza.

## Opción 2: Implementar archivos compilados archivados en Amazon S3 desde un bucket de origen de S3

En esta opción, los comandos de compilación de la fase de compilación compilan el TypeScript código en JavaScript código y despliegan el resultado en el bucket de destino de S3 en una carpeta independiente con fecha y hora. Primero, crea el TypeScript código y un archivo buildspec.yml. Tras combinar los archivos fuente en un archivo ZIP, se carga el archivo ZIP de origen en el bucket de origen de S3 y se utiliza un CodeBuild escenario para implementar un archivo ZIP de aplicación creado en el bucket de destino de S3. El código compilado se conserva en formato comprimido en el bucket de destino.

### Requisitos previos

Debe disponer de lo siguiente:

- Un bucket de origen de S3. Puede utilizar el bucket que creó en Tutorial: Crear una canalización simple (bucket de S3).
- Un bucket de destino de S3. Consulte Alojamiento de un sitio web estático en Amazon S3. Asegúrese de crear su bucket de la Región de AWS misma manera que la canalización que desea crear.

> **Note**
>
> En este ejemplo, se muestra la implementación de archivos en un bucket privado. No habilite el bucket de destino para el alojamiento de sitios web ni le asocie políticas que lo conviertan en un bucket público.

## Paso 1: Crear y cargar los archivos de código fuente en un bucket de código fuente de S3

En esta sección, debe crear y cargar los archivos de código fuente en el bucket que utiliza la canalización para la etapa de código fuente. En esta sección, se proporcionan instrucciones para la creación de los siguientes archivos de código fuente:

- Un `buildspec.yml` archivo que se utiliza para proyectos de CodeBuild construcción.
- Un archivo `index.ts`.

Para crear un archivo buildspec.yml

- Cree un archivo denominado `buildspec.yml` con el siguiente contenido. Estos comandos de compilación instalan TypeScript y utilizan el TypeScript compilador para reescribir el código en `index.ts` código. JavaScript

```
version: 0.2

phases:
  install:
    commands:
      - npm install -g typescript
  build:
    commands:
      - tsc index.ts
artifacts:
  files:
    - index.js
```

Para crear un archivo index.ts

- Cree un archivo denominado `index.ts` con el siguiente contenido.

```
interface Greeting {
    message: string;
}

class HelloGreeting implements Greeting {
    message = "Hello!";
}

function greet(greeting: Greeting) {
    console.log(greeting.message);
}

let greeting = new HelloGreeting();

greet(greeting);
```

Para cargar los archivos en un bucket de código fuente de S3

1. Sus archivos deberían ser parecidos a estos en su directorio local:

```
buildspec.yml
index.ts
```

   Comprima los archivos y asigne al archivo el nombre `source.zip`.

2. En la consola de Amazon S3, para el bucket de origen, elija Cargar. Elija Add files (Añadir archivos), a continuación, busque el archivo ZIP que ha creado.
3. Seleccione Cargar. Estos archivos son el artefacto de código fuente creado por el asistente Create Pipeline (Crear canalización) para la acción de implementación en CodePipeline. El archivo debería aparecer así en el bucket:

```
source.zip
```

## Paso 2: Crear la canalización

En esta sección, debe crear una canalización con las siguientes acciones:

- Una etapa de origen con una acción de Amazon S3 en la que los artefactos de origen son los archivos de la aplicación descargable.
- Una etapa de implementación con una acción de implementación de Amazon S3.

Para crear una canalización con el asistente

1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http://console.aws.amazon. com/codesuite/codepipeline/home.
2. En la página Bienvenido, Introducción o en la página Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyS3DeployPipeline`**.
5. En Función de servicio, elija Nueva función de servicio CodePipeline para poder crear una función de servicio en IAM.
6. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
7. En Paso 3: agregar la etapa de origen, en Proveedor de origen, elija Amazon S3. En Bucket, elija el nombre del bucket de código fuente. En S3 object key (Clave de objeto de S3), escriba el nombre del archivo ZIP de código fuente. Asegúrese de incluir la extensión de archivo .zip.

   Elija Siguiente.
8. En Paso 4: agregar la etapa de compilación:

   a. En Build provider (Proveedor de compilación), elija CodeBuild.

   b. Elija Crear el proyecto de compilación. En la página Create project (Crear proyecto):

   c. En Project name (Nombre de proyecto), escriba un nombre para este proyecto de compilación.

   d. En Environment (Entorno), elija Managed image (Imagen administrada). En Operating system (Sistema operativo), elija Ubuntu.

e. En Runtime, elija Standard (Estándar). Para la versión Runtime, elijaaws/codebuild/standard: 1.0.

f. En Image version (Version de la imagen), elija Always use the latest image for this runtime version (Utilizar siempre la imagen más reciente para esta versión del tiempo de ejecución).

g. En Rol de servicio, elija su rol CodeBuild de servicio o cree uno.

h. En Build specifications (Especificaciones de compilación), elija Use a buildspec file (Usar un archivo buildspec).

i. Selecciona Continuar a CodePipeline. Se muestra un mensaje si el proyecto se ha creado correctamente.

j. Elija Siguiente.

9. En Paso 5: agregar la etapa de implementación:

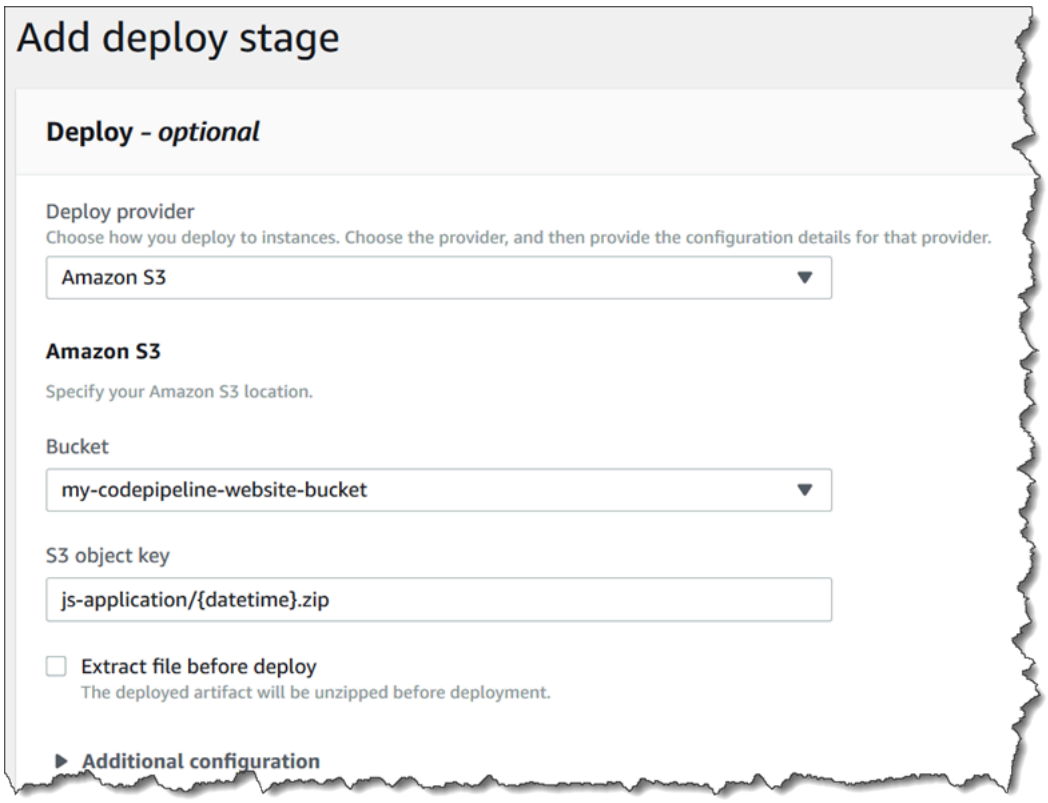
a. En Deploy provider (Proveedor de implementación), elija Amazon S3.

b. En Bucket, especifique el nombre del bucket de destino de S3.

c. Asegúrese de que no está seleccionada la opción Extract file before deploy (Extraer el archivo antes de la implementación).

Cuando no está seleccionada la opción Extract file before deploy (Extraer el archivo antes de la implementación), se muestra S3 object key (Clave de objeto de S3). Especifique el nombre de la ruta que desea utilizar: `js-application/{datetime}.zip`.

Esto crea una carpeta `js-application` en Amazon S3 en la que se extraen los archivos. En esta carpeta, la variable `{datetime}` crea una marca de tiempo en cada archivo de salida cuando se ejecuta la canalización.




d. (Opcional) En Canned ACL puede aplicar un conjunto de garantías predefinidas, conocido como una ACL predefinida a los artefactos cargados.

e. (Opcional) En Cache control (Control del caché) introduzca los parámetros de caché. Puede establecerlo para controlar el comportamiento del caché para las solicitudes/respuestas. Para valores válidos, consulte el `Cache-Control`campo del encabezado para las operaciones HTTP

f. Elija Siguiente.

10. En Step 6: Review, revise la información y después elija Create pipeline.

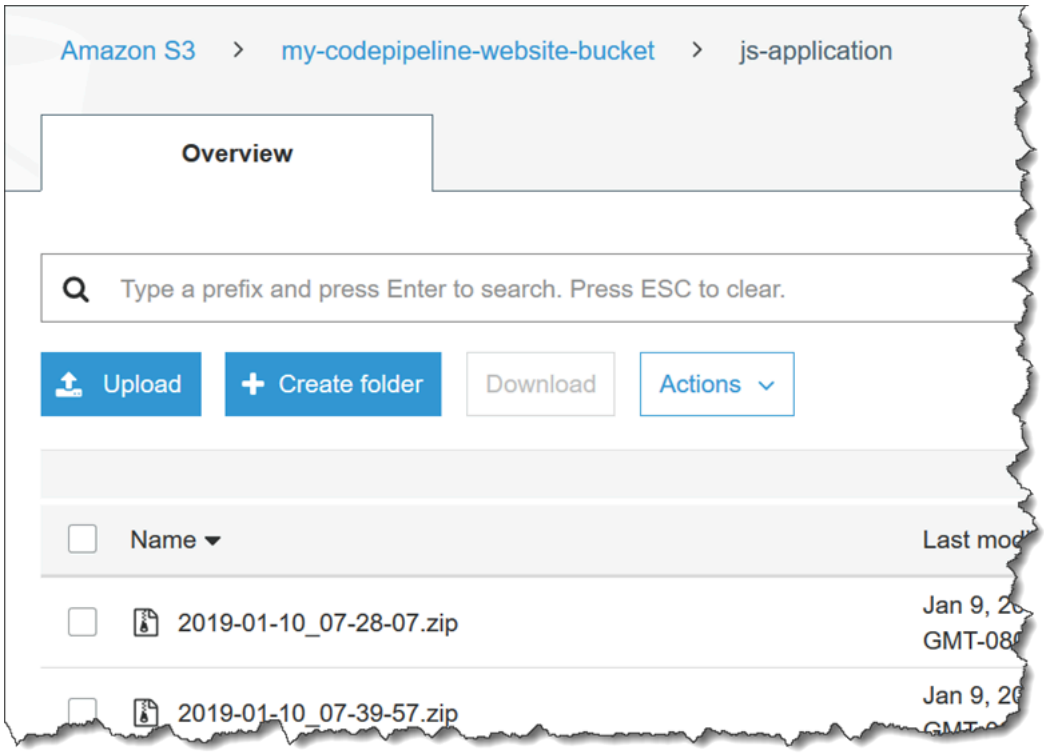
11. Después de que la canalización se ejecute correctamente, visualice el bucket en la consola de Amazon S3. Compruebe que el archivo ZIP implementado se muestre en el bucket de destino en la carpeta `js-application`. El JavaScript archivo contenido en el archivo ZIP debe ser`index.js`. El archivo `index.js` contiene la salida siguiente:

```
var HelloGreeting = /** @class */ (function () {
    function HelloGreeting() {
        this.message = "Hello!";
    }
    return HelloGreeting;
}());
```

```
function greet(greeting) {
    console.log(greeting.message);
}
var greeting = new HelloGreeting();
greet(greeting);
```

### Paso 3: Realizar un cambio en cualquier archivo de código fuente y verificar la implementación

Realice un cambio en los archivos de código fuente y, a continuación, cárguelos en el bucket de código fuente. Esto desencadena la ejecución de la canalización. Visualice el bucket de destino y verifique que los archivos de salida implementados estén disponibles en la carpeta `js-application`, tal y como se muestra a continuación:




# Tutorial: Cree una canalización que publique su aplicación sin servidor en el AWS Serverless Application Repository

Puede utilizarla AWS CodePipeline para entregar continuamente su aplicación AWS SAM sin servidor al. AWS Serverless Application Repository

> **Important**
>
> Como parte de la creación de una canalización, se utilizará un depósito de artefactos de S3 proporcionado por el cliente CodePipeline para almacenar artefactos. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y fiable.

En este tutorial se muestra cómo crear y configurar una canalización para crear una aplicación sin servidor alojada en ella GitHub y publicarla en ella automáticamente. AWS Serverless Application Repository La canalización GitHub se utiliza como proveedor de código fuente y CodeBuild como proveedor de compilación. Para publicar su aplicación sin servidor en AWS Serverless Application Repository, debe implementar una aplicación (desde AWS Serverless Application Repository) y asociar la función Lambda creada por esa aplicación como un proveedor de acciones de invocación en su canalización. De este modo, podrá enviar actualizaciones de la aplicación de forma continua a AWS Serverless Application Repository, sin necesidad de escribir ningún código.

> **Important**
>
> Muchas de las acciones que añada a la canalización en este procedimiento implican AWS recursos que debe crear antes de crear la canalización. AWS Los recursos para las acciones de origen siempre deben crearse en la misma AWS región en la que se creó la canalización. Por ejemplo, si creas tu canalización en la región EE.UU. Este (Ohio), tu CodeCommit repositorio debe estar en la región EE.UU. Este (Ohio).
>
> Puedes añadir acciones entre regiones al crear tu canalización. AWS los recursos para las acciones entre regiones deben estar en la misma AWS región en la que planeas ejecutar la acción. Para obtener más información, consulte Agregue una acción interregional en CodePipeline.

## Antes de empezar

En este tutorial, se presupone lo siguiente.

- Está familiarizado con AWS Serverless Application Model (AWS SAM) y con el AWS Serverless Application Repository.

- Tiene alojada una aplicación sin servidor GitHub que ha publicado en ella AWS Serverless Application Repository mediante la AWS SAM CLI. Para publicar una aplicación de ejemplo en la AWS Serverless Application Repository, consulte Inicio rápido: publicación de aplicaciones en la Guía para AWS Serverless Application Repository desarrolladores. Para publicar su propia aplicación en AWS Serverless Application Repository, consulte Publicar aplicaciones mediante la AWS SAM CLI en la Guía para AWS Serverless Application Model desarrolladores.

## Paso 1: Crear un archivo buildspec.yml

Cree un `buildspec.yml` archivo con el siguiente contenido y agréguelo al GitHub repositorio de su aplicación sin servidor. *`template.yml`* Sustitúyalo por la AWS SAM plantilla de la aplicación y *`bucketname`* por el depósito de S3 en el que se almacena la aplicación empaquetada.

```
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
```

## Paso 2: Crear y configurar la canalización

Siga estos pasos para crear su canalización en el Región de AWS lugar donde desee publicar la aplicación sin servidor.

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. Si es necesario, cambie al Región de AWS lugar en el que desee publicar la aplicación sin servidor.
3. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
4. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.

5. Seleccione Create pipeline. En la página Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba el nombre de la canalización.
6. En Tipo de canalización, seleccione V2. Para obtener más información, consulte Tipos de canalización. Elija Siguiente.
7. En Función de servicio, elija Nueva función de servicio CodePipeline para permitir la creación de una función de servicio en IAM.
8. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
9. En la página Paso 3: Añadir la fase de origen, en Proveedor de origen, seleccione GitHub.
10. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para la acción GitHub de origen, consulteGitHub conexiones.
11. En Repositorio, elige tu repositorio GitHub de origen.
12. En Branch, elige tu GitHub sucursal.
13. Deje los valores predeterminados restantes para la acción de origen. Elija Siguiente.
14. En la página Paso 4: agregación de la etapa de compilación, agregue una etapa de compilación:
    a. En Build provider (Proveedor de compilación), elija AWS CodeBuild. En Region (Región), utilice la región de la canalización.
    b. Elija Crear proyecto.
    c. En Project name (Nombre de proyecto), escriba un nombre para este proyecto de compilación.
    d. En Environment image (Imagen de entorno), elija Managed image (Imagen administrada). En Operating system (Sistema operativo), elija Ubuntu.
    e. En Runtime (Tiempo de ejecución) y Runtime version (Versión de tiempo de ejecución), elija el tiempo de ejecución y la versión necesarios para la aplicación sin servidor.
    f. En Service role (Rol de servicio), elija New service role (Nuevo rol de servicio).
    g. En Build specifications (Especificaciones de compilación), elija Use a buildspec file (Usar un archivo buildspec).
    h. Selecciona Continuar a CodePipeline. De este modo, se abre la CodePipeline consola y se crea un CodeBuild proyecto que lo utiliza `buildspec.yml` en su repositorio para la configuración. El proyecto de compilación utiliza un rol de servicio para administrar los permisos del Servicio de AWS . Es posible que este paso tarde un par de minutos.
    i. Elija Siguiente.

15. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.

16. En la página Paso 6: Añadir fase de despliegue, seleccione Omitir fase de despliegue y, a continuación, acepte el mensaje de advertencia seleccionando Omitir de nuevo. Elija Siguiente.
17. En el paso 7: Revisar, selecciona Crear canalización. Debería ver un diagrama con las etapas.
18. Conceda permiso al rol de CodeBuild servicio para acceder al bucket de S3 donde está almacenada la aplicación empaquetada.

    a. En la etapa Build (Compilación) de la canalización nueva, elija CodeBuild.
    b. Elija la pestaña Build details (Detalles de compilación).
    c. En Entorno, elija el rol CodeBuild de servicio para abrir la consola de IAM.
    d. Amplíe la selección para `CodeBuildBasePolicy` y elija Edit policy (Editar política).
    e. Elija JSON.
    f. Añada una instrucción de política nueva con el siguiente contenido. La declaración permite CodeBuild colocar objetos en el depósito de S3 donde se almacena la aplicación empaquetada. *`bucketname`*Sustitúyalo por el nombre del depósito de S3.

```
{
    "Effect": "Allow",
    "Resource": [
        "arn:aws:s3:::bucketname/*"
    ],
    "Action": [
        "s3:PutObject"
    ]
}
```

    g. Elija Revisar política.
    h. Seleccione Save changes (Guardar cambios).

## Paso 3: Implementar la aplicación de publicación

Siga estos pasos para implementar la aplicación que contiene la función de Lambda que lleva a cabo la publicación en el AWS Serverless Application Repository. Esta aplicación es aws-serverless-codepipeline-serverlessrepo-publish.

> **Note**
>
> Debes implementar la aplicación en el mismo lugar Región de AWS que tu canalización.

1. Vaya a la página de la aplicación y elija Deploy (Implementar).
2. Seleccione I acknowledge that this app creates custom IAM roles (Confirmo que esta aplicación puede crear roles de IAM personalizados).
3. Elija Implementar.
4. Selecciona View CloudFormation Stack para abrir la CloudFormation consola.
5. Amplíe la sección Resources (Recursos). Verás ServerlessRepoPublish, que es de ese tipo AWS::Lambda::Function. Anote el ID físico de este recurso para el siguiente paso. Lo utilizará al crear la acción de publicación nueva en CodePipeline.

## Paso 4: Crear la acción de publicación

Siga estos pasos para crear la acción de publicación en la canalización.

1. Abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la sección de navegación izquierda, elija la canalización que desea editar.
3. Elija Edit (Edición de).
4. Después de la última etapa de la canalización actual, elija + Add stage (Añadir etapa). En Stage name (Nombre de etapa), escriba un nombre, por ejemplo **`Publish`**, y elija Add stage (Añadir etapa).
5. En la nueva etapa, elija +Add action group (+Añadir grupo de acciones).
6. Introduzca un nombre para la acción. En Action provider (Proveedor de acción), en Invoke (Invocar), elija AWS Lambda.
7. En Artefactos de entrada, elija BuildArtifact.
8. En Nombre de función, elija el ID físico de la función de Lambda que ha anotado en el paso anterior.
9. Elija Save (Guardar) para la acción.
10. Elija Done (Listo) para la etapa.
11. En la parte superior derecha, elija Save (Guardar).

12. Para verificar tu canalización, realiza un cambio en tu aplicación en GitHub. Por ejemplo, cambia la descripción de la aplicación en la `Metadata` sección del archivo de AWS SAM plantilla. Confirma el cambio y envíalo a tu GitHub sucursal. Esto desencadena la ejecución de la canalización. Cuando la canalización esté completa, compruebe que la aplicación se ha actualizado con el cambio en el AWS Serverless Application Repository.

# Tutorial: Implementaciones de funciones Lambda con CodePipeline

Este tutorial le ayuda a crear una acción de despliegue CodePipeline que despliegue el código en la función que haya configurado en Lambda. Creará una función de Lambda de ejemplo en la que se crean un alias y una versión, se agrega la función de Lambda comprimida a la ubicación de origen y se ejecuta la acción de Lambda en la canalización.

> **Note**
>
> Como parte de la creación de una canalización en la consola, para los artefactos se utilizará un depósito de artefactos de CodePipeline S3. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y fiable.

> **Note**
>
> La acción de implementación de `Lambda` solo está disponible para canalizaciones de tipo V2.

## Requisitos previos

Para poder usar este tutorial para crear su propia canalización de implementación continua debe tener instalados algunos recursos. Esto es lo que necesita para empezar:

> **Note**
>
> Todos estos recursos deben crearse en la misma región. AWS

- Un repositorio de control de código fuente, por ejemplo GitHub, o un depósito de S3 de origen (en este tutorial se utiliza S3), donde almacenará un `.zip` archivo que cree para la función Lambda.
- Debe usar un rol de CodePipeline servicio existente que se haya actualizado con los permisos para esta acción. Para actualizar su rol de servicio, consulte Permisos de la política de rol de servicio para la acción de implementación de Lambda.

Una vez satisfechos estos requisitos previos, puede continuar con el tutorial y crear su canalización de implementación continua.

## Paso 1: creación de la función de Lambda de ejemplo

En este paso, se crea la función de Lambda en la que se realizará la implementación.

Para crear una función de Lambda

1. Acceda a la consola de Lambda y realice los pasos del siguiente tutorial para crear una función de Lambda de ejemplo: enlace.
2. En la barra de navegación superior, elija Crear y, en la parte superior de la página, seleccione Comenzar desde cero.
3. En Name (Nombre), escriba **`MyLambdaFunction`**.
4. Publique una nueva versión. Esta será la versión a la que apuntará el alias.
   a. Seleccione su función.
   b. Elija el menú desplegable Acciones.
   c. Elija Publicar nueva versión.
   d. (Opcional) Agregue una descripción en Descripción.
   e. Elija Publicar.
5. Cree un alias para su función, por ejemplo, `aliasV1`.
6. Asegúrese de que el alias apunte a la versión que acaba de crear (por ejemplo, 1).

> **Note**
>
> Si elige $LATEST, no puede usar las características de transferencia de tráfico porque Lambda no admite $LATEST para un alias que apunte a más de una versión.

## Paso 2: carga del archivo de la función en el repositorio

Descargue la función y guárdela como un archivo .zip. Cargue el archivo comprimido en el bucket de S3 siguiendo los pasos que se indican a continuación.

Para añadir un archivo **`.zip`** a su repositorio de código fuente

1. Abra su bucket de S3.
2. Seleccione Cargar.
3. Suba el archivo .zip que contiene su archivo `sample_lambda_source.zip` al bucket de origen.

   Tome nota de la ruta.

   ```
   object key
   ```

## Paso 3: creación de una canalización

Usa el CodePipeline asistente para crear las etapas de la canalización y conectar el repositorio de origen.

Para crear la canalización

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyPipeline`**.
5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios de CodePipeline, consulte Precios.
6. En Función de servicio, elija Usar la función de servicio existente y, a continuación, elija la función de CodePipeline servicio que se ha actualizado con los permisos necesarios para esta

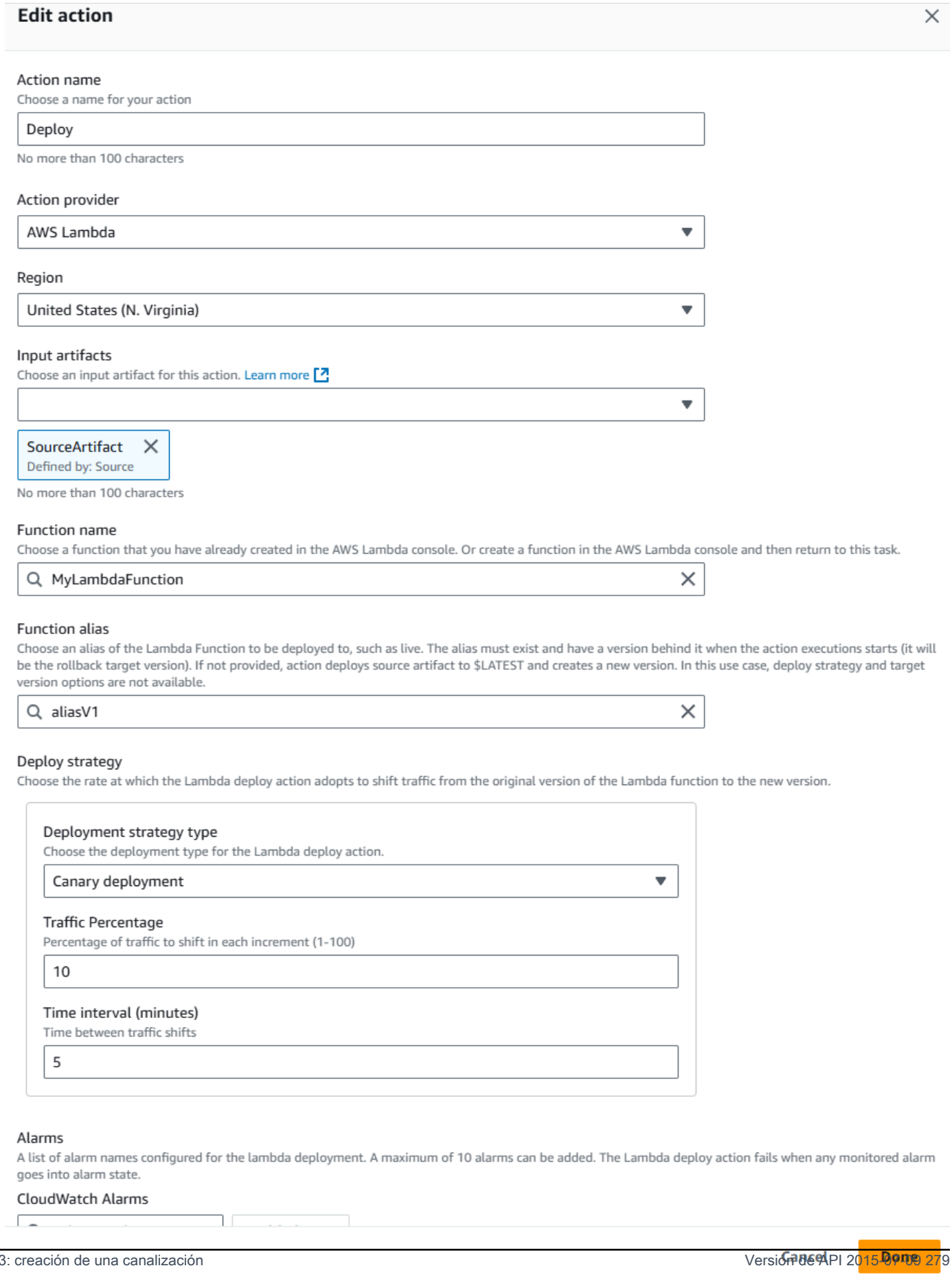
acción. Para configurar el rol CodePipeline de servicio para esta acción, consultePermisos de la política de rol de servicio para la acción de implementación de Lambda.

7. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
8. En la página Paso 3: agregar la etapa de origen, agregue una etapa de origen:

   a. En Source provider (Proveedor de código fuente), elija Amazon S3.

   b. En Clave de objeto, agregue el nombre del archivo .zip, incluida la extensión del archivo, por ejemplo, `sample_lambda_source.zip`.

   Elija Siguiente.
9. En la página Paso 4: agregación de la etapa de compilación, seleccione Omitir.
10. En la página Paso 5: agregación de la etapa de prueba, seleccione Omitir.
11. En la página Paso 6: agregación de la etapa de implementación, elija Lambda.

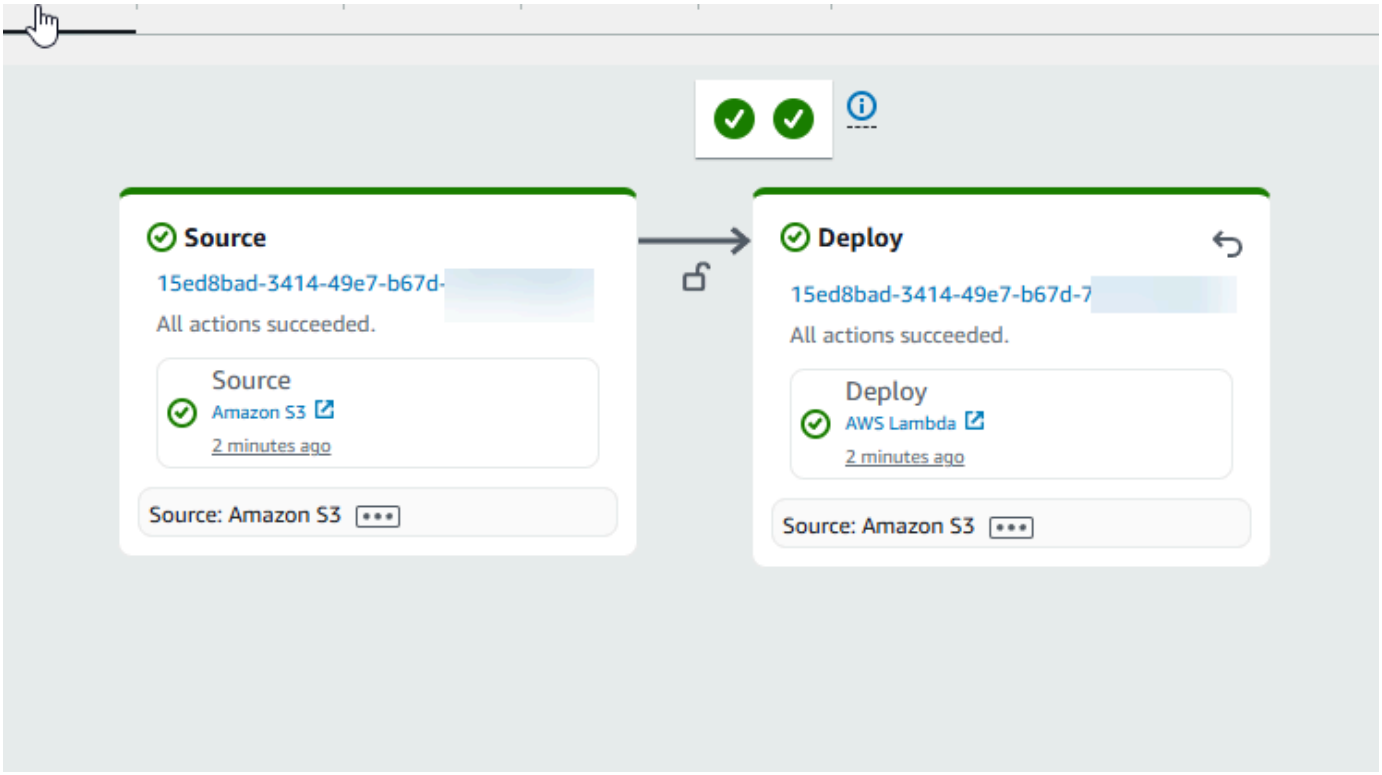

Edit action
Action name
Choose a name for your action
Deploy
No more than 100 characters
Action provider
AWS Lambda
Region
United States (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
Defined by: Source
No more than 100 characters
Function name
Choose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

a. Agregue el nombre de la función y el alias.

b. Elija su estrategia de implementación.

c. Elija Siguiente.

12. En la página Paso 7: Revisar, revisa la configuración de la canalización y selecciona Crear canalización para crear la canalización.




## Paso 4: Probar la canalización

Tu canalización debe tener todo lo necesario para ejecutar una implementación AWS continua end-to-end nativa. Ahora, pruebe su funcionalidad enviando un cambio de código al repositorio de código fuente.

Para probar la canalización

1. Realice una modificación del código en el repositorio de código fuente configurado, valide y envíe el cambio.
2. Abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
3. Seleccione su canalización de la lista.

4. Vea el progreso en la canalización a través de sus etapas. La canalización debería completarse y la acción implementarse en la función de Lambda.

## Más información

La acción de implementación de Lambda permite dos métodos de implementación. Un método es el cambio de tráfico por sí solo sin un artefacto de entrada de la acción de origen. El otro método consiste en actualizar el código de la función mediante un artefacto de entrada de la acción de origen y, a continuación, publicar una nueva versión basada en el código actualizado. Para el segundo método, si se proporciona el alias, también CodePipeline se realizará el cambio de tráfico. En este tutorial sobre la acción de implementación de Lambda, se muestra cómo actualizar la función mediante un artefacto de origen.

Para obtener más información sobre la acción, consulte la página de referencia de la acción en AWS Lambda Referencia de la acción de implementación de.

# Tutorial: Uso de variables con acciones de invocación de Lambda

Una acción de invocación de Lambda puede usar variables de otra acción como parte de su entrada y devolver nuevas variables junto con su salida. Para obtener información sobre las variables para las acciones en CodePipeline, consulteReferencia de variables.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará para los artefactos un depósito de artefactos de S3 proporcionado CodePipeline por el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

Al final de este tutorial, tendrá:

- Una acción de invocación de Lambda que:
  - Consume la `CommitId` variable de una acción de origen CodeCommit
  - Da como resultado tres nuevas variables: `dateTime`, `testRunId` y `region`

- Una acción de aprobación manual que consume las nuevas variables de la acción de invocación de Lambda para proporcionar una URL de prueba y un ID de ejecución de prueba
- Una canalización actualizada con las nuevas acciones

Temas


- Requisitos previos
- Paso 1: Crear una función de Lambda
- Paso 2: Añadir una acción de invocación de Lambda y una acción de aprobación manual a la canalización


## Requisitos previos

Antes de comenzar, debe disponer de lo siguiente:

- Puedes crear o usar la canalización con la CodeCommit fuente incluidaTutorial: Crear una canalización simple (CodeCommit repositorio).
- Edita tu canalización actual para que la acción CodeCommit de origen tenga un espacio de nombres. Asigne el espacio de nombres `SourceVariables` a la acción.

## Paso 1: Crear una función de Lambda

Realice los pasos siguientes para crear una función de Lambda y un rol de ejecución de Lambda. Agregue la acción de Lambda a la canalización después de crear la función de Lambda.

Crear una función de Lambda y rol de ejecución

1. Inicia sesión en Consola de administración de AWS y abre la AWS Lambda consola en https://console.aws.amazon.com/lambda/.
2. Seleccione Creación de función. Deje seleccionado Author from scratch (Crear desde cero).
3. En Function name (Nombre de la función), escriba el nombre de la función, como **`myInvokeFunction`**. En Runtime (Motor de ejecución), deje seleccionada la opción predeterminada.
4. Expanda Choose or create an execution role (Elegir o crear un rol de ejecución). Elija Create a new role with basic Lambda permissions (Crear un nuevo rol con permisos básicos de Lambda).
5. Seleccione Creación de función.

6. Para usar una variable de otra acción, tendrá que pasar `UserParameters` a la configuración de acción de invocación de Lambda. Configurarás la acción en tu canalización más adelante en el tutorial, pero añadirás el código asumiendo que se pasará la variable.

   Para producir nuevas variables, establezca una propiedad llamada `outputVariables` en la entrada a `putJobSuccessResult`. Tenga en cuenta que no puede producir variables como parte de `putJobFailureResult`.

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

   En la nueva función, en la pestaña Código, pegue el siguiente código de ejemplo en `index.mjs`.

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

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

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

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

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

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

```
                dateTime: Date(Date.now()).toString(),
                region: lambdaRegion
            }
        };

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

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

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

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

        // Your tests here

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

```
};
```

7. Permita que la función se guarde automáticamente.
8. Copie el nombre de recurso de Amazon (ARN) en el campo Función ARN situado en la parte superior de la pantalla.
9. Como último paso, abre la consola AWS Identity and Access Management (IAM) en https://console.aws.amazon.com/iam/. Modifique la función de ejecución de Lambda para agregar la siguiente política:. AWSCodePipelineCustomActionAccess Para consultar los pasos necesarios para crear un rol de ejecución de Lambda o modificar la política de rol, consulte Paso 2: Crear la función de Lambda.

## Paso 2: Añadir una acción de invocación de Lambda y una acción de aprobación manual a la canalización

En este paso, agregue una acción de invocación de Lambda a la canalización. Agregue la acción como parte de una etapa denominada Test (Prueba). El tipo de acción es una acción de invocación. A continuación, agregue una acción de aprobación manual después de la acción de invocación.

Para agregar una acción de Lambda y una acción de aprobación manual a la canalización

1. Abra la CodePipeline consola en. https://console.aws.amazon.com/codepipeline/

   Se muestran los nombres de todas las canalizaciones asociadas a su AWS cuenta. Elija la canalización en la que desea agregar la acción.
2. Agregue la acción de prueba de Lambda a su canalización.

   a. Para editar la canalización, elija Editar. Agregue una etapa después de la acción de origen en la canalización existente. Escriba un nombre para la etapa, por ejemplo **`Test`**.

   b. En la nueva etapa, elija +Agregar el grupo de acción para agregar una acción. En Action name (Nombre de la acción), escriba el nombre de la acción de invocación, como **`Test_Commit`**.

   c. En Proveedor de acción, seleccione AWS Lambda.

   d. En Input artifacts (Artefactos de entrada), elija el nombre del artefacto de salida de la acción de origen, como `SourceArtifact`.

   e. En FunctionName, añada el ARN de la función de Lambda que ha creado.

f. En el Variable namespace (Espacio de nombres de la variable), agregue el nombre del espacio de nombres, como **`TestVariables`**.

g. En Artefactos de salida, añada el nombre del artefacto de salida, como **`LambdaArtifact`**.

h. Seleccione Listo.

3. Agregue la acción de aprobación manual a la canalización.

a. Con la canalización todavía en modo de edición, agregue una etapa después de la acción de invocación. Escriba un nombre para la etapa, por ejemplo **`Approval`**.

b. En la nueva etapa, elija el icono para agregar una acción. En Action name (Nombre de la acción), escriba el nombre de la acción de aprobación, como **`Change_Approval`**.

c. En Action provider (Proveedor de acciones), elija Manual approval (Aprobación manual).

d. En URL for review (URL para revisión), cree la URL agregando la sintaxis de variable para la variable `region` y la variable `CommitId`. Asegúrese de utilizar los espacios de nombres asignados a las acciones que proporcionan las variables de salida.

En este ejemplo, la URL con la sintaxis variable de una CodeCommit acción tiene el espacio de nombres predeterminado. `SourceVariables` La variable de salida de región de Lambda tiene el espacio de nombres `TestVariables`. La URL tiene el siguiente aspecto.

```
https://#{TestVariables.region}.console.aws.amazon.com/codesuite/codecommit/
repositories/MyDemoRepo/commit/#{SourceVariables.CommitId}
```

En Comments (Comentarios), cree el texto del mensaje de aprobación agregando la sintaxis de la variable para la variable `testRunId`. Para este ejemplo, la dirección URL con la sintaxis de variable para la variable de salida de Lambda `testRunId` tiene el espacio de nombres `TestVariables`. Escriba el siguiente mensaje.

```
Make sure to review the code before approving this action. Test Run ID:
 #{TestVariables.testRunId}
```

4. Elija Done (Listo) para cerrar la pantalla de edición de la acción y, a continuación, elija Done (Listo) para cerrar la pantalla de edición de la etapa. Para guardar la canalización, elija Done (Listo). La canalización completada ahora contiene una estructura con etapas de origen, prueba, aprobación e implementación.

Elija Release change (Publicar modificación) para ejecutar el último cambio a través de la estructura de la canalización.

5. Cuando la canalización llegue a la etapa de aprobación manual, elija Review (Revisar). Las variables resueltas aparecen como la URL del ID de confirmación. El aprobador puede elegir la URL para consultar la confirmación.
6. Después de que la canalización se ejecute de forma satisfactoria, también puede consultar los valores de las variables en la página del historial de ejecución de acciones.

# Tutorial: Usa una acción de AWS Step Functions invocación en una canalización

Se puede utilizar AWS Step Functions para crear y configurar máquinas de estado. En este tutorial, se muestra cómo se agrega una acción de invocación a una canalización que activa las ejecuciones de máquinas de estados desde la canalización.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará un depósito de artefactos de S3 proporcionado por el cliente CodePipeline para fabricar artefactos. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y fiable.

En este tutorial, va a realizar las siguientes tareas:

- Crea un estado estándar en el que esté la máquina. AWS Step Functions
- Escribir directamente la entrada JSON de la máquina de estados. También puede cargar el archivo de entrada de la máquina de estados en un bucket de Amazon Simple Storage Service (Amazon S3).
- Actualizar la canalización agregando la acción de la máquina de estados.

Temas


- Requisito previo: cree o elija una canalización sencilla
- Paso 1: Crear la máquina de estados de ejemplo
- Paso 2: Agregar una acción de invocación de Step Functions a la canalización

## Requisito previo: cree o elija una canalización sencilla

En este tutorial, va a agregar una acción de invocación a una canalización existente. Puede utilizar la canalización que creó en Tutorial: Crear una canalización simple (bucket de S3) o Tutorial: Crear una canalización simple (CodeCommit repositorio).

Va a utilizar una canalización existente con una acción de origen y al menos una estructura de dos etapas, pero no se utilizan artefactos de origen en este ejemplo.

> **Note**
>
> Es posible que tenga que actualizar el rol de servicio que se utiliza en la canalización con otros permisos necesarios para ejecutar esta acción. Para ello, abra la consola AWS Identity and Access Management (IAM), busque el rol y, a continuación, añada los permisos a la política del rol. Para obtener más información, consulte Agrega permisos al rol de CodePipeline servicio.

## Paso 1: Crear la máquina de estados de ejemplo

En la consola de Step Functions, cree una máquina de estados utilizando la plantilla de ejemplo `HelloWorld`. Para obtener instrucciones, consulte Crea una máquina de estadosen la Guía para desarrolladores de AWS Step Functions .

## Paso 2: Agregar una acción de invocación de Step Functions a la canalización

Agregue una acción de invocación de Step Functions a la canalización del siguiente modo:

1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http://console.aws.amazon. com/codesuite/codepipeline/home.

   Se muestran los nombres de todas las canalizaciones asociadas a tu AWS cuenta.
2. En Nombre, elija el nombre de la canalización que desea editar. Esto abre una vista detallada de la canalización, que incluye el estado de cada una de las acciones en cada etapa de la canalización.
3. En la página de detalles de la canalización, elija Edit.

4. En la segunda etapa de la canalización sencilla, elija Editar la etapa. Elija Eliminar. De este modo, eliminará la segunda etapa, ahora que ya no la necesita.
5. En la parte inferior del diagrama, seleccione + Add stage (Añadir etapa).
6. En Nombre de la etapa, escriba un nombre, por ejemplo, **`Invoke`**, y elija Agregar la etapa.
7. Elija + Add action group (Añadir grupo de acciones).
8. En Nombre de la acción, escriba un nombre; por ejemplo, **`Invoke`**.
9. En Proveedor de acción, elija AWS Step Functions. En el campo Region (Región) conserve el valor predeterminado de la región de la canalización.
10. En Artefactos de entrada, elija `SourceArtifact`.
11. En ARN de máquina de estado, elija el nombre de recurso de Amazon (ARN) de la máquina de estados que creó anteriormente.
12. (Opcional) En Prefijo de nombre de ejecución, escriba un prefijo para agregarlo al ID de ejecución de la máquina de estados.
13. En Tipo de entrada, elija Literal.
14. En Entrada, especifique el JSON de entrada que la máquina de estados del ejemplo `HelloWorld` espera usar.

    > **Note**
    >
    > La entrada a la ejecución de la máquina de estados es diferente del término utilizado CodePipeline para describir los artefactos de entrada para las acciones.

    En este ejemplo, especifique el siguiente JSON:

    ```
    {"IsHelloWorldExample": true}
    ```

15. Seleccione Listo.
16. En la etapa que está editando, elija Listo. En el panel de AWS CodePipeline , elija Save (Guardar) y, a continuación, elija Save (Guardar) cuando aparezca el mensaje de advertencia.
17. Para enviar los cambios y comenzar una ejecución de la canalización, elija Release change (Publicar modificación) y, a continuación, Release (Publicar).
18. En la canalización completada, elija AWS Step Functions en la acción de invocación. En la AWS Step Functions consola, consulta el identificador de ejecución de la máquina de estado. El ID

indica el nombre de la máquina de estados `HelloWorld` y el ID de ejecución con el prefijo `my-prefix`.

```
arn:aws:states:us-west-2:account-ID:execution:HelloWorld:my-
prefix-0d9a0900-3609-4ebc-925e-83d9618fcca1
```

# Tutorial: Crear una canalización que se utilice AWS AppConfig como proveedor de despliegue

En este tutorial, configurará una canalización que entregue archivos de configuración de forma continua AWS AppConfig utilizándolos como proveedor de acciones de despliegue en la fase de despliegue.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará un depósito de artefactos de S3 proporcionado por el cliente CodePipeline para crear artefactos. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y fiable.

Temas


- Requisitos previos
- Paso 1: Crea tus recursos AWS AppConfig
- Paso 2: Cargar los archivos en un bucket de origen de S3
- Paso 3: Crear la canalización
- Paso 4: Realizar un cambio en cualquier archivo de origen y verificar la implementación


## Requisitos previos

Debe seguir los siguientes pasos antes de comenzar:

- En este ejemplo, se utiliza un origen de S3 para la canalización. Cree o utilice un bucket de Amazon S3 con el control de versiones habilitado. Puede seguir las instrucciones de Paso 1: creación de un bucket de origen de S3 para la aplicación para crear un bucket de S3.

## Paso 1: Crea tus recursos AWS AppConfig

En esta sección, se crean los recursos siguientes.

- Una aplicación AWS AppConfig es una unidad lógica de código que proporciona funciones a sus clientes.
- Un entorno AWS AppConfig es un grupo de AppConfig objetivos de despliegue lógico, como aplicaciones en un entorno beta o de producción.
- Un perfil de configuración es un conjunto de opciones que influyen en el comportamiento de la aplicación. El perfil de configuración permite acceder AWS AppConfig a la configuración en su ubicación almacenada.
- (Opcional) Una estrategia de implementación AWS AppConfig define el comportamiento de una implementación de configuración, por ejemplo, qué porcentaje de clientes deben recibir la nueva configuración implementada en un momento dado durante una implementación.

Para crear una aplicación, un entorno, un perfil de configuración y una estrategia de implementación

1. Inicie sesión en Consola de administración de AWS.
2. Siga los pasos de los siguientes temas para crear sus recursos en AWS AppConfig.
   - Crear una aplicación.
   - Crear un entorno.
   - Cree un perfil AWS CodePipeline de configuración.
   - (Opcional) Elegir una estrategia de implementación predefinida o crear una propia.

## Paso 2: Cargar los archivos en un bucket de origen de S3

En esta sección, cree su archivo o archivos de configuración. A continuación, comprima y coloque los archivos fuente en el bucket que la canalización utiliza para la etapa de origen.

Para crear archivos de configuración

1. Cree un archivo `configuration.json` para cada configuración de cada región. Incluya los siguientes contenidos:

```
Hello World!
```

2. Siga estos pasos para comprimir y cargar los archivos de configuración.

Para comprimir y cargar los archivos fuente

1. Cree un archivo .zip con los archivos y asigne un nombre al archivo .zip `configuration-files.zip`. Por ejemplo, el archivo.zip puede usar la siguiente estructura:

```
.
### appconfig-configurations
    ### MyConfigurations
        ### us-east-1
        #   ### configuration.json
        ### us-west-2
            ### configuration.json
```

2. En la consola de Amazon S3 para su bucket, elija Cargar y siga las instrucciones para cargar el archivo .zip.

## Paso 3: Crear la canalización

En esta sección, debe crear una canalización con las siguientes acciones:

- Una etapa de origen con una acción de Amazon S3 en la que los artefactos de origen son los archivos para su configuración.
- Una etapa de despliegue con una acción AppConfig de despliegue.

Para crear una canalización con el asistente

1. Inicie sesión Consola de administración de AWS y abra la CodePipeline consola en http:// console.aws.amazon. com/codesuite/codepipeline/home.
2. En la página Bienvenido, Introducción o Canalizaciones, elija Crear canalización.

3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyAppConfigPipeline`**.
5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios de CodePipeline, consulte Precios.
6. En Función de servicio, elija Nueva función de servicio CodePipeline para poder crear una función de servicio en IAM.
7. En Advanced settings (Configuración avanzada), deje los valores predeterminados y elija Next (Siguiente).
8. En Paso 3: agregar la etapa de origen, en Proveedor de origen, elija Amazon S3. En Bucket, elija el nombre del bucket de origen de S3.

   En Clave de objeto de S3, escriba el nombre del archivo .zip: `configuration-files.zip`.

   Elija Siguiente.
9. En Paso 4: agregar la etapa de compilación, elija Omitir la etapa de compilación y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

   Elija Siguiente.
10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.
11. En el paso 6: Añadir la etapa de despliegue:

    a. En Deploy provider, elija AWS AppConfig.

    b. En Aplicación, elija el nombre de la aplicación en la que creó AWS AppConfig. El campo muestra el ID de la aplicación.

    c. En Entorno, elija el nombre del entorno en el que creó AWS AppConfig. El campo muestra el ID de su entorno.

    d. En Perfil de configuración, elija el nombre del perfil de configuración en el que creó AWS AppConfig. El campo muestra el ID de su perfil de configuración.

e. En Estrategia de implementación, elija el nombre de su estrategia de implementación. Puede ser una estrategia de despliegue que haya creado AppConfig o una que haya elegido entre las estrategias de despliegue predefinidas AppConfig. El campo muestra el ID de su estrategia de implementación.

f. En Ruta de configuración del artefacto de entrada, introduzca la ruta del archivo. Asegúrese de que la ruta de configuración del artefacto de entrada coincida con la estructura de directorios del archivo.zip del bucket de S3. En este ejemplo, especifique la siguiente ruta de archivo: `appconfig-configurations/MyConfigurations/us-west-2/configuration.json`.

g. Elija Siguiente.

12. En el paso 7: Revisar, revise la información y, a continuación, seleccione Crear canalización.

## Paso 4: Realizar un cambio en cualquier archivo de origen y verificar la implementación

Realice un cambio en los archivos fuente y, a continuación, cárguelos en el bucket. Esto desencadena la ejecución de la canalización. Compruebe que la configuración esté disponible consultando la versión.

# Tutorial: Utilice un clon completo con una fuente de GitHub canalización

Puedes elegir la opción de clonación completa para tu acción GitHub fuente en CodePipeline. Usa esta opción para ejecutar CodeBuild comandos para los metadatos de Git en la acción de creación de tu canalización.

> **Note**
>
> La opción de clonación completa que se describe aquí se refiere a especificar si se CodePipeline deben clonar los metadatos del repositorio, que solo pueden ser utilizados por CodeBuild comandos. Para usar un token de acceso de GitHub usuario para usarlo en CodeBuild proyectos, siga los pasos que se indican a continuación para instalar el AWS Connector para la GitHub aplicación y, a continuación, deje el campo de instalación de la aplicación en blanco. AWS CodeConnections utilizará el token de acceso de usuario para la conexión.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará un depósito de artefactos de S3 proporcionado por el cliente CodePipeline para fabricar artefactos. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y fiable.

En este tutorial, crearás una canalización que se conecta a tu GitHub repositorio, usa la opción de clonación completa para los datos de origen y ejecuta una CodeBuild compilación que clona tu repositorio y ejecuta comandos de Git para el repositorio.

> **Note**
>
> Esta función no está disponible en las regiones de Asia Pacífico (Hong Kong), África (Ciudad del Cabo), Oriente Medio (Baréin), Europa (Zúrich) AWS GovCloud o US-West (). Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline. Para ver consideraciones sobre esta acción en la región de Europa (Milán), consulte la nota que aparece en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

Temas


- Requisitos previos
- Paso 1: Crear un archivo README
- Paso 2: Crear la canalización y compilar el proyecto
- Paso 3: actualice la política CodeBuild de roles de servicio para usar las conexiones
- Paso 4: Ver comandos del repositorio en el resultado de la compilación


## Requisitos previos

Antes de empezar, debe hacer lo siguiente:

- Crea un GitHub repositorio con tu GitHub cuenta.

- Ten tus GitHub credenciales preparadas. Cuando utilices el Consola de administración de AWS para configurar una conexión, se te pedirá que inicies sesión con tus GitHub credenciales.

## Paso 1: Crear un archivo README

Tras crear el GitHub repositorio, sigue estos pasos para añadir un archivo README.

1. Inicia sesión en tu GitHub repositorio y elige tu repositorio.
2. Para crear un nuevo archivo, seleccione Añadir archivo > Crear archivo nuevo. Asigne un nombre al archivo `README.md` y agregue el siguiente texto.

   ```
   This is a GitHub repository!
   ```

3. Seleccione Confirmar cambios.

   Asegúrese de que el archivo `README.md` está en el nivel raíz del repositorio.

## Paso 2: Crear la canalización y compilar el proyecto

En esta sección, debe crear una canalización con las siguientes acciones:

- Una etapa de origen con una conexión a tu GitHub repositorio y a tu acción.
- Una etapa de creación con una acción de AWS CodeBuild creación.

Para crear una canalización con el asistente

1. Inicia sesión en la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página Bienvenido, Introducción o en la página Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyGitHubPipeline`**.
5. En Tipo de canalización, elija V1 para los fines de este tutorial. También puede elegir V2; sin embargo, tenga en cuenta que los tipos de canalización difieren en cuanto a características y precio. Para obtener más información, consulte Tipos de canalización.
6. En Service role (Rol de servicio), elija New service role (Nuevo rol de servicio).

> **Note**
>
> Si opta por utilizar su función de CodePipeline servicio actual, asegúrese de haber añadido el permiso de `codestar-connections:UseConnection` IAM a su política de función de servicio. Para obtener instrucciones sobre la función CodePipeline de servicio, consulte Añadir permisos a la función CodePipeline de servicio.

7. Para Configuración avanzada deje los valores predeterminados. En Artifact store (Almacén de artefactos), elija Default location (Ubicación predeterminada) para utilizar el almacén de artefactos predeterminado, como el bucket de artefacto de Amazon S3 que se estableció como predeterminado, para la canalización en la región que seleccionó para esta.

   > **Note**
   >
   > Este no es el bucket de origen para su código fuente. Este es el almacén de artefactos de la canalización. Cada canalización debe tener su propio almacén de artefactos independiente, como un bucket de S3.

   Elija Siguiente.

8. En la página Paso 3: agregar la etapa de origen, agregue una etapa de origen:

   a. En Proveedor de origen, elija GitHub (a través de GitHub la aplicación).

   b. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para la acción GitHub de origen, consulteGitHub conexiones.

      Se instala una aplicación para todas las conexiones a un proveedor en particular. Si ya ha instalado el AWS conector para la GitHub aplicación, elíjalo y omita este paso.

      > **Note**
      >
      > Si desea crear un token de acceso de usuario, asegúrese de que ya ha instalado el AWS Connector para la GitHub aplicación y, a continuación, deje vacío el campo de instalación de la aplicación. AWS CodeConnections utilizará el token de acceso de usuario para la conexión. Para obtener más información, consulte Acceder a su proveedor de origen en CodeBuild.

c. En Repository name (Nombre de repositorio), elija el nombre de su repositorio de GitHub.

d. En Nombre de ramificación, elija la ramificación de repositorio que desea utilizar.

e. Asegúrese de que la opción Start the pipeline on source code change (Iniciar la canalización en el cambio del código fuente) está seleccionada.

f. En Formato de artefacto de salida, seleccione Clonación completa para habilitar la opción de clonación de Git para el repositorio de origen. Solo las acciones proporcionadas por CodeBuild pueden usar la opción de clonación de Git. Paso 3: actualice la política CodeBuild de roles de servicio para usar las conexionesEn este tutorial, utilizarás esta opción para actualizar los permisos de tu rol de servicio de CodeBuild proyectos.

Elija Siguiente.

9. En Paso 4: agregación de la etapa de compilación, agregue una etapa de compilación:

a. En Build provider (Proveedor de compilación), elija AWS CodeBuild. En el campo Region (Región) conserve el valor predeterminado de la región de la canalización.

b. Elija Crear proyecto.

c. En Project name (Nombre de proyecto), escriba un nombre para este proyecto de compilación.

d. En Environment image (Imagen de entorno), elija Managed image (Imagen administrada). En Operating system (Sistema operativo), elija Ubuntu.

e. En Runtime, elija Standard (Estándar). En Imagen, elija aws/codebuild/standard:5.0.

f. En Service role (Rol de servicio), elija New service role (Nuevo rol de servicio).

> **Note**
>
> Anote el nombre de su función de servicio. CodeBuild Necesitará el nombre del rol para el paso final de este tutorial.

g. En Buildspec, para Build specifications (Especificaciones de la compilación), elija Insert build commands (Insertar comandos de compilación). Elija Cambiar a editor y pegue lo siguiente en Comandos de compilación:

> **Note**
>
> En la sección `env` de la especificación de compilación, asegúrese de que el ayudante de credenciales para los comandos de git esté habilitado, como se muestra en este ejemplo.

```
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. Selecciona Continuar a CodePipeline. Esto vuelve a la CodePipeline consola y crea un CodeBuild proyecto que utiliza los comandos de compilación para la configuración. El proyecto de compilación usa un rol de servicio para administrar Servicio de AWS los permisos. Es posible que este paso tarde un par de minutos.

i. Elija Siguiente.

10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.

11. En la página Paso 6: Añadir fase de despliegue, seleccione Omitir fase de despliegue y, a continuación, acepte el mensaje de advertencia seleccionando Omitir de nuevo. Elija Siguiente.

12. En el paso 7: Revisar, selecciona Crear canalización.

## Paso 3: actualice la política CodeBuild de roles de servicio para usar las conexiones

La ejecución inicial de la canalización fallará porque la función de CodeBuild servicio debe actualizarse con permisos para usar las conexiones. Añada el permiso de IAM de `codestar-connections:UseConnection` a la política de roles de servicio. Para obtener instrucciones sobre cómo actualizar la política en la consola de IAM, consulte Añade CodeBuild GitClone permisos para las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com.

## Paso 4: Ver comandos del repositorio en el resultado de la compilación

1. Cuando tu rol de servicio se haya actualizado correctamente, selecciona Reintentar en la CodeBuild fase fallida.
2. Después de que la canalización se ejecute correctamente, en la etapa de implementación, elija Ver detalles.

En la página de detalles, elija la pestaña Registros. Vea el resultado de la CodeBuild compilación. Los comandos muestran el valor de la variable introducida.

Los comandos muestran el contenido del archivo `README.md`, muestran los archivos en el directorio, clonan el repositorio, visualizan el registro y archivan el repositorio como un archivo ZIP.

# Tutorial: Usa la clonación completa con una fuente de CodeCommit canalización

Puedes elegir la opción de clonación completa para tu acción CodeCommit de origen en CodePipeline. Usa esta opción para permitir el acceso CodeBuild a los metadatos de Git en tu acción de creación en proceso.

En este tutorial, crearás una canalización que acceda a tu CodeCommit repositorio, utilizarás la opción de clonación completa para los datos de origen y ejecutarás una CodeBuild compilación que clona tu repositorio y ejecuta los comandos de Git para el repositorio.

> **Note**
>
> CodeBuild las acciones son las únicas acciones posteriores que admiten el uso de los metadatos de Git disponibles con la opción de clonación de Git. Además, aunque tu canalización puede incluir acciones entre cuentas, la CodeCommit acción y la CodeBuild acción deben estar en la misma cuenta para que la opción de clonación completa funcione correctamente.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará para los artefactos un depósito de artefactos de S3 proporcionado CodePipeline por el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

Temas


- Requisitos previos
- Paso 1: Crear un archivo README
- Paso 2: Crear la canalización y compilar el proyecto
- Paso 3: Actualiza la política CodeBuild de roles de servicio para clonar el repositorio
- Paso 4: Ver comandos del repositorio en el resultado de la compilación


## Requisitos previos

Antes de empezar, debes crear un CodeCommit repositorio en la misma AWS cuenta y región que tu canalización.

## Paso 1: Crear un archivo README

Siga estos pasos para añadir un archivo README a su repositorio de origen. El archivo README proporciona un ejemplo de archivo fuente para que lo lea la acción CodeBuild posterior.

Para añadir un archivo README

1. Inicie sesión en su repositorio y elija su repositorio.
2. Para crear un nuevo archivo, seleccione Añadir archivo > Crear archivo. Asigne un nombre al archivo `README.md` y agregue el siguiente texto.

   ```
   This is a CodeCommit repository!
   ```
3. Seleccione Confirmar cambios.

   Asegúrese de que el archivo `README.md` está en el nivel raíz del repositorio.

## Paso 2: Crear la canalización y compilar el proyecto

En esta sección, debe crear una canalización con las siguientes acciones:

- Una etapa de origen con una acción de CodeCommit origen.
- Una etapa de construcción con una acción de AWS CodeBuild compilación.

Para crear una canalización con el asistente

1. Inicia sesión en la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página Bienvenido, Introducción o en la página Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyCodeCommitPipeline`**.
5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.
6. En Service role (Rol de servicio), realice una de las operaciones siguientes:
   - Elija Existing service role (Rol de servicio existente)
   - Elige tu rol CodePipeline de servicio actual. El rol debe tener el permiso de IAM de `codecommit:GetRepository` para la política de rol de servicio. Consulte Agregar permisos a la función CodePipeline de servicio.
7. Para Configuración avanzada deje los valores predeterminados. Elija Siguiente.
8. En la página Paso 3: agregar la etapa de origen, haga lo siguiente:

   a. En Source provider (Proveedor de código fuente), elija CodeCommit.

   b. En Nombre de repositorio, elija el nombre de su repositorio.

   c. En Nombre de ramificación, elija el nombre de su ramificación.

   d. Asegúrese de que la opción Start the pipeline on source code change (Iniciar la canalización en el cambio del código fuente) está seleccionada.

   e. En Formato de artefacto de salida, seleccione Clonación completa para habilitar la opción de clonación de Git para el repositorio de origen. Solo las acciones proporcionadas por CodeBuild pueden usar la opción de clonación de Git.

   Elija Siguiente.
9. En Paso 4: agregación de la etapa de compilación, haga lo siguiente:

   a. En Build provider (Proveedor de compilación), elija AWS CodeBuild. En el campo Region (Región) conserve el valor predeterminado de la región de la canalización.

b. Elija Crear proyecto.

c. En Project name (Nombre de proyecto), escriba un nombre para este proyecto de compilación.

d. En Environment image (Imagen de entorno), elija Managed image (Imagen administrada). En Operating system (Sistema operativo), elija Ubuntu.

e. En Runtime, elija Standard (Estándar). En  Image, elige  aws/codebuild  /standard:5.0.

f. En Service role (Rol de servicio), elija New service role (Nuevo rol de servicio).

> **Note**
>
> Anote el nombre de su rol de servicio. CodeBuild Necesitará el nombre del rol para el paso final de este tutorial.

g. En Buildspec, para Build specifications (Especificaciones de la compilación), elija Insert build commands (Insertar comandos de compilación). Elija Cambiar a editor y en Comandos de compilación pegue el código siguiente:

```
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. Elige Continuar hasta CodePipeline. De este modo, volverás a la CodePipeline consola y crearás un CodeBuild proyecto que usará tus comandos de compilación para la configuración. El proyecto de compilación utiliza un rol de servicio para administrar los permisos del Servicio de AWS . Es posible que este paso tarde un par de minutos.

i. Elija Siguiente.

10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.

11. En la página Paso 6: Agregar etapa de implementación, selecciona Omitir etapa de implementación y, a continuación, acepta el mensaje de advertencia seleccionando nuevamente Omitir. Elija Siguiente.

12. En el paso 7: revisar, selecciona Crear canalización.

## Paso 3: Actualiza la política CodeBuild de roles de servicio para clonar el repositorio

La ejecución inicial del proceso fallará porque tendrás que actualizar el rol de CodeBuild servicio con permisos para extraerlo de tu repositorio.

Añada el permiso de IAM de `codecommit:GitPull` a la política de roles de servicio. Para obtener instrucciones sobre cómo actualizar la política en la consola de IAM, consulte Agregue CodeBuild GitClone permisos para las acciones CodeCommit de origen.

### Paso 4: Ver comandos del repositorio en el resultado de la compilación

Para ver el resultado de la compilación

1. Cuando tu rol de servicio se actualice correctamente, elige Reintentar en la CodeBuild fase fallida.
2. Después de que la canalización se ejecute correctamente, en la etapa de implementación, elija Ver detalles.

   En la página de detalles, elija la pestaña Registros. Vea el resultado de la CodeBuild compilación. Los comandos muestran el valor de la variable introducida.

   Los comandos muestran el contenido del archivo `README.md`, muestran los archivos en el directorio, clonan el repositorio, visualizan el registro y ejecutan `git describe --all`.

## Tutorial: Crea una canalización con AWS CloudFormation StackSets acciones de despliegue

En este tutorial, utilizarás la AWS CodePipeline consola para crear una canalización con acciones de despliegue para crear un conjunto de pilas y crear instancias de pila. Cuando se ejecuta la canalización, la plantilla crea un conjunto de pilas y también crea y actualiza las instancias en las que se implementa el conjunto de pilas.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará para los artefactos un depósito de artefactos de S3 proporcionado CodePipeline por el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

Hay dos maneras de gestionar los permisos de un conjunto de pilas: roles de IAM autogestionados y AWS gestionados. En este tutorial, se proporcionan ejemplos con permisos autoadministrados.

Para usarlo StackSets de la manera más eficaz posible CodePipeline, debes tener un conocimiento claro de los conceptos subyacentes CloudFormation StackSets y de cómo funcionan. Consulte StackSets los conceptos en la Guía AWS CloudFormation del usuario.

Temas


- Requisitos previos
- Paso 1: Cargue la muestra AWS CloudFormation archivo de plantilla y parámetros
- Paso 2: Crear la canalización
- Paso 3: Ver la implementación inicial
- Paso 4: Añade una CloudFormationStackInstances acción
- Paso 5: Ver los recursos del conjunto de pilas para su implementación
- Paso 6: Actualizar el conjunto de pilas


## Requisitos previos

Para las operaciones de conjuntos apilados, se utilizan dos cuentas diferentes: una cuenta de administración y una cuenta de destino. Los conjuntos de pilas se crean en la cuenta de administrador. Se crean pilas individuales que pertenecen a un conjunto de pilas de la cuenta de destino.

Para crear un rol de administrador con su cuenta de administrador

- Siga las instrucciones de Configuración de permisos básicos para operaciones con conjuntos de pilas. Su rol debe llamarse **`AWSCloudFormationStackSetAdministrationRole`**.

Para crear el rol de servicio en la cuenta de destino

- Cree un rol de servicio en la cuenta de destino que confíe en la cuenta de administrador. Siga las instrucciones de Configuración de permisos básicos para operaciones con conjuntos de pilas. Su rol debe llamarse **`AWSCloudFormationStackSetExecutionRole`**.

# Paso 1: Cargue la muestra AWS CloudFormation archivo de plantilla y parámetros

Cree un bucket de origen para los archivos de parámetros y plantillas del conjunto de pilas. Descarga el archivo de AWS CloudFormation plantilla de muestra, configura un archivo de parámetros y, a continuación, comprime los archivos antes de subirlos a tu bucket fuente de S3.

> **Note**
>
> Asegúrese de comprimir los archivos de origen antes de subirlos a su bucket de origen de S3, incluso si el único archivo fuente es la plantilla.

Para crear un bucket de origen de S3

1. Inicie sesión en la consola de Amazon S3 Consola de administración de AWS y ábrala en https://console.aws.amazon.com/s3/.
2. Elija Crear bucket.
3. En Nombre del bucket, escriba un nombre para el bucket.

   En Región la región en la que desea crear la canalización. Elija Crear bucket.
4. Una vez creado el bucket, aparecerá un banner donde se indicará que la operación se ha realizado correctamente. Elija Go to bucket details (Acceder a los detalles del bucket).
5. En la pestaña Properties (Propiedades), elija Versioning (Control de versiones). Elija Enable versioning (Habilitar control de versiones) y haga clic en Save (Guardar).

Para crear el AWS CloudFormation archivo de plantilla

1. Descargue el siguiente archivo de plantilla de ejemplo para generar la CloudTrail configuración de los conjuntos de pilas:https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSCloudtrail.yml.
2. Guarde el archivo como `template.yml`.

Para crear el archivo parameters.txt

1. Cree un archivo con los parámetros para la implementación. Los parámetros son valores que desea actualizar en su pila en tiempo de ejecución. El siguiente archivo de ejemplo actualiza los parámetros de la plantilla del conjunto de pilas para permitir la validación del registro y los eventos globales.

```
[
  {
    "ParameterKey": "EnableLogFileValidation",
    "ParameterValue": "true"
  },
  {
    "ParameterKey": "IncludeGlobalEvents",
    "ParameterValue": "true"
  }
]
```

2. Guarde el archivo como `parameters.txt`.

Para crear el archivo accounts.txt

1. Cree un archivo con las cuentas en las que desee crear las instancias, como se muestra en el siguiente archivo de ejemplo.

```
[
    "111111222222","333333444444"
]
```

2. Guarde el archivo como `accounts.txt`.

Para crear y cargar los archivos de origen

1. Combine los archivos .zip en un solo archivo ZIP. Sus archivos deberían aparecer así en el archivo ZIP.

```
template.yml
parameters.txt
accounts.txt
```

2. Cargue el archivo ZIP en el bucket de S3. Este archivo es el artefacto de origen creado por el asistente Create Pipeline para su acción de implementación en CodePipeline.

## Paso 2: Crear la canalización

En esta sección, debe crear una canalización con las siguientes acciones:

- Una fase de origen con una acción de origen de S3 en la que el artefacto de origen es el archivo de plantilla y cualquier archivo de origen de respaldo.
- Una fase de despliegue con una acción de despliegue de conjuntos de CloudFormation pilas que crea el conjunto de pilas.
- Una etapa de despliegue con una acción de despliegue de instancias CloudFormation apiladas que crea las pilas e instancias dentro de las cuentas de destino.

Para crear una canalización con una acción CloudFormationStackSet

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En la página Bienvenido, Introducción o en la página Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyStackSetsPipeline`**.
5. En Tipo de canalización, elija V1 para los fines de este tutorial. También puede elegir V2; sin embargo, tenga en cuenta que los tipos de canalización difieren en cuanto a características y precio. Para obtener más información, consulte Tipos de canalización.
6. En Rol de servicio, elija Nuevo rol de servicio CodePipeline para poder crear un rol de servicio en IAM.
7. En Almacén de artefactos, deje los valores predeterminados.

> **Note**
>
> Este no es el bucket de origen para su código fuente. Este es el almacén de artefactos de la canalización. Cada canalización debe tener su propio almacén de artefactos independiente, como un bucket de S3. Al crear o editar una canalización, debes tener un

depósito de artefactos en la región de canalización y un depósito de artefactos por AWS región en la que ejecutes una acción.
Para obtener más información, consulte Artefactos de entrada y salida y CodePipeline referencia de estructura de tuberías.

Elija Siguiente.

8. En la página Paso 3: agregar la etapa de origen, en Proveedor de origen, elija Amazon S3.
9. En Bucket, introduzca el bucket de origen de S3 creado para este tutorial, por ejemplo `BucketName`. En Clave de objeto de S3, escriba la ruta y el nombre del archivo ZIP, como `MyFiles.zip`.
10. Elija Siguiente.
11. En Paso 4: agregar la etapa de compilación, elija Omitir la etapa de compilación y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.
12. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.
13. En el  paso 6: añade la etapa de despliegue:

    a. En Proveedor de implementación, elija Conjunto de pilas de AWS CloudFormation ).

    b. En Nombre de conjunto de pilas, escriba un nombre para el conjunto de pilas. Este es el nombre del conjunto de pilas que crea la plantilla.

    > **Note**
    >
    > Anote el nombre del conjunto de pilas. La usará cuando añada la segunda acción de StackSets despliegue a su canalización.

    c. En Ruta de plantilla, introduzca el nombre del artefacto y la ruta del archivo donde subió el archivo de plantilla. Por ejemplo, introduzca lo siguiente utilizando el el nombre del artefacto de origen por defecto `SourceArtifact`.

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

d. En Destinos de implementación, introduzca el nombre del artefacto y la ruta del archivo donde subió el archivo de cuentas. Por ejemplo, introduzca lo siguiente utilizando el el nombre del artefacto de origen por defecto `SourceArtifact`.

```
SourceArtifact::accounts.txt
```

e. En Objetivo de  despliegue Regiones de AWS, introduce una región para el despliegue de tu instancia de pila inicial, por ejemplo`us-east-1`.

f. Expanda Opciones de implementación. En Parámetros, introduzca el nombre del artefacto y la ruta del archivo donde cargó el archivo de parámetros. Por ejemplo, introduzca lo siguiente utilizando el el nombre del artefacto de origen por defecto `SourceArtifact`.

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

Para introducir los parámetros como una entrada literal en lugar de como una ruta de archivo, introduzca lo siguiente:

```
ParameterKey=EnableLogFileValidation,ParameterValue=true
ParameterKey=IncludeGlobalEvents,ParameterValue=true
```

g. En Capacidades, elija CAPABILITY_IAM y CAPABILITY_NAMED_IAM.

h. En Modelo de permiso, elija SELF_MANAGED.

i. En Porcentaje de tolerancia a errores, introduzca `20`.

j. En Porcentaje máximo simultáneo, introduzca `25`.

k. Elija Siguiente.

l. En Paso 7: revisión, elija Crear canalización. Aparece su canalización.

m. Permita que su canalización se ejecute.

## Paso 3: Ver la implementación inicial

Vea los recursos y el estado de su implementación inicial. Tras comprobar que la implementación ha creado correctamente el conjunto de pilas, puede añadir la segunda acción a la fase Deploy / Implementación).

Para ver los recursos

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En Canalizaciones, elija la canalización y luego Ver. El diagrama muestra las etapas de código fuente e implementación de la canalización.
3. Elige la CloudFormation acción en la CloudFormationStackSet acción de tu cartera. La plantilla, los recursos y los eventos de tu conjunto de pilas se muestran en la CloudFormation consola.
4. En el panel de navegación de la izquierda, elige StackSets. En la lista, elija el nuevo conjunto de pilas.
5. Elija la pestaña Instancias de la pila. Verifique que se haya creado una instancia de pila para cada cuenta que haya proporcionado en la región us-east-1. Compruebe que el estado de cada instancia de pila sea `CURRENT`.

## Paso 4: Añade una CloudFormationStackInstances acción

Crea una próxima acción en tu proceso que te permita CloudFormation StackSets crear las instancias de pila restantes.

Para crear una acción siguiente en su canalización

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.

   En Canalizaciones, elija la canalización y luego Ver. El diagrama muestra las etapas de código fuente e implementación de la canalización.
2. Elija esta opción para editar la canalización. La canalización se muestra en modo de edición.
3. En la etapa Implementar, elija Editar.
4. En la acción de implementación de Conjunto de pilas de AWS CloudFormation CloudFormation, seleccione Añadir grupo de acciones.
5. En la página Editar acción, añada los detalles de la acción:

   a. En Nombre de la acción, escriba un nombre para la acción.

   b. En Proveedor de acciones, elija Instancias de pila de AWS CloudFormation ).

   c. En Artefactos de entrada, elija SourceArtifact.

   d. En Nombre de conjunto de pilas, introduzca el nombre para el conjunto de pilas. Este es el nombre del conjunto de pilas proporcionado en la primera acción.

e. En Destinos de implementación, introduzca el nombre del artefacto y la ruta del archivo donde subió el archivo de cuentas. Por ejemplo, introduzca lo siguiente utilizando el el nombre del artefacto de origen por defecto `SourceArtifact`.

```
SourceArtifact::accounts.txt
```

f. En Objetivo de despliegue Regiones de AWS, introduce las regiones en las que se desplegarán las instancias de pila restantes, `eu-central-1` como `us-east-2` las siguientes:

```
us-east2, eu-central-1
```

g. En Porcentaje de tolerancia a errores, introduzca `20`.

h. En Porcentaje máximo simultáneo, introduzca `25`.

i. Seleccione Save.

j. Publica un cambio de forma manual. La canalización actualizada se muestra con dos acciones en la etapa de implementación.

## Paso 5: Ver los recursos del conjunto de pilas para su implementación

Puede ver los recursos y el estado de su implementación del conjunto de pilas.

Para ver los recursos

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En Canalizaciones, elija la canalización y luego Ver. El diagrama muestra las etapas de código fuente e implementación de la canalización.
3. Elige la CloudFormation acción en la **`AWS CloudFormation Stack Instances`** acción de tu cartera. La plantilla, los recursos y los eventos de tu conjunto de pilas se muestran en la CloudFormation consola.
4. En el panel de navegación de la izquierda, elige StackSets. En la lista, elija el conjunto de pilas.
5. Elija la pestaña Instancias de la pila. Compruebe que todas las instancias de pila restantes de cada cuenta que haya proporcionado se hayan creado o actualizado en las regiones esperadas. Compruebe que el estado de cada instancia de pila sea `CURRENT`.

## Paso 6: Actualizar el conjunto de pilas

Actualice su conjunto de pilas e impleméntela en las instancias. En este ejemplo, también cambia los objetivos de implementación que desea designar para la actualización. Las instancias que no forman parte de la actualización pasan a un estado desactualizado.

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En Canalizaciones, elija la canalización y luego Editar. En la etapa Implementar, elija Editar.
3. Elija editar la acción Conjunto de pila de AWS CloudFormation ) en su canalización. En Descripción, sobrescriba la descripción existente por una nueva descripción para el conjunto de pilas.
4. Elija editar la acción Instancias de pila de AWS CloudFormation ) en su canalización. En Objetivo de implementación Regiones de AWS, borre el `us-east-2` valor que se ingresó cuando se creó la acción.
5. Guarde los cambios. Elija Publicar modificación para ejecutar su canalización.
6. Abra su acción en CloudFormation. Elige la pestaña de StackSet información. En la StackSet descripción, comprueba que se muestre la nueva descripción.
7. Elija la pestaña Instancias de la pila. En Estado, verifique que el estado de las instancias de la pila en us-east-2 sea `OUTDATED`.

# Tutorial: Cómo crear una regla de verificación de variables para una canalización como una condición de entrada

En este tutorial, configurará una canalización que entregue archivos de forma continua GitHub utilizándolos como proveedor de acciones de origen en la fase de origen. La canalización completa detecta cambios cuando se realiza un cambio en los archivos de código fuente en el repositorio de código fuente. La canalización se ejecuta y, a continuación, compara las variables de salida con el nombre del repositorio de origen y el nombre de la ramificación proporcionados en la condición de entrada a la etapa de compilación.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará para los artefactos un depósito de artefactos de S3 proporcionado CodePipeline por el cliente. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una

cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y confiable.

**Important**

Muchas de las acciones que añades a tu canalización en este procedimiento implican AWS recursos que debes crear antes de crear la canalización. AWS Los recursos para tus acciones de origen siempre deben crearse en la misma AWS región en la que creas tu canalización. Por ejemplo, si creas tu canalización en la región Este de EE. UU. (Ohio), tu CodeCommit repositorio debe estar en la región Este de EE. UU. (Ohio).
Puedes añadir acciones interregionales al crear tu canalización. AWS Los recursos para las acciones interregionales deben estar en la misma AWS región en la que planeas ejecutar la acción. Para obtener más información, consulte Agregue una acción interregional en CodePipeline.

En este ejemplo, se utiliza la canalización del ejemplo con una acción fuente GitHub (versión 2) y una acción de CodeBuild compilación en las que la condición de entrada para la fase de creación es comprobar si hay variables.

## Requisitos previos

Antes de empezar, debe hacer lo siguiente:

- Crea un GitHub repositorio con tu GitHub cuenta.
- Ten tus GitHub credenciales listas. Cuando utilice el Consola de administración de AWS para configurar una conexión, se le pedirá que inicie sesión con sus GitHub credenciales.
- Una conexión a tu repositorio para configurarla GitHub (a través de la GitHub aplicación) como la acción de origen de tu canalización. Para crear una conexión con tu GitHub repositorio, consultaGitHub conexiones.

## Paso 1: Cree un archivo fuente de muestra y agréguelo a su GitHub repositorio

En esta sección, debe crear y agregar los archivos de código fuente al repositorio que utiliza la canalización para la etapa de origen. Para este ejemplo, va a generar y agregar lo siguiente:

- Un archivo `README.md`.

Después de crear tu GitHub repositorio, sigue estos pasos para añadir tu archivo README.

1. Inicia sesión en tu GitHub repositorio y elige tu repositorio.
2. Para crear un nuevo archivo, seleccione Agregar el archivo y, a continuación, Crear nuevo archivo. Asigne un nombre al archivo `README.md` y agregue el siguiente texto.

   ```
   This is a GitHub repository!
   ```

3. Seleccione Confirmar cambios. Para los fines de este tutorial, agregue un mensaje de confirmación que contenga la palabra “Update” en mayúscula, como en el siguiente ejemplo:

   ```
   Update to source files
   ```

   > **Note**
   >
   > La verificación de reglas para cadenas distingue entre mayúsculas y minúsculas.

   Asegúrese de que el archivo `README.md` está en el nivel raíz del repositorio.

## Paso 2: Crear la canalización

En esta sección, debe crear una canalización con las siguientes acciones:

- Una fase fuente con una conexión a tu GitHub repositorio y a tu acción.
- Una etapa de CodeBuild compilación en la que la etapa tiene una condición On Entry configurada para la regla de verificación de variables.

Para crear una canalización con el asistente

1. Inicia sesión en la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página Bienvenido, Introducción o en la página Canalizaciones, elija Crear canalización.
3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.

4. En Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba **`MyVarCheckPipeline`**.
5. CodePipeline proporciona tuberías de tipo V1 y V2, que difieren en características y precio. El tipo V2 es el único tipo que puede elegir en la consola. Para obtener más información, consulte Tipos de canalización. Para obtener información sobre los precios CodePipeline, consulte Precios.
6. En Service role (Rol de servicio), elija New service role (Nuevo rol de servicio).

   > **Note**
   >
   > Si, en cambio, optas por usar tu rol de CodePipeline servicio actual, asegúrate de haber agregado el permiso de `codeconnections:UseConnection` IAM a tu política de roles de servicio. Para obtener instrucciones sobre la función CodePipeline de servicio, consulta Añadir permisos a la función CodePipeline de servicio.

7. Para Configuración avanzada deje los valores predeterminados.

   Elija Siguiente.
8. En la página Paso 3: agregar la etapa de origen, agregue una etapa de origen:

   a. En Proveedor de origen, elija GitHub (a través de GitHub la aplicación).

   b. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para tu acción GitHub de origen, consultaGitHub conexiones.

   c. En Repository name (Nombre de repositorio), elija el nombre de su repositorio de GitHub .

   d. En Nombre de ramificación, elija la ramificación de repositorio que desea utilizar.

   e. Compruebe que la opción Sin desencadenador esté seleccionada.

   Elija Siguiente.
9. En Paso 4: agregación de la etapa de compilación, agregue una etapa de compilación:

   a. En Build provider (Proveedor de compilación), elija AWS CodeBuild. En el campo Region (Región) conserve el valor predeterminado de la región de la canalización.

   b. Elija Crear proyecto.

   c. En Project name (Nombre de proyecto), escriba un nombre para este proyecto de compilación.

d. En Environment image (Imagen de entorno), elija Managed image (Imagen administrada). En Operating system (Sistema operativo), elija Ubuntu.

e. En Runtime, elija Standard (Estándar). En Image, elija aws/codebuild /standard:5.0.

f. En Service role (Rol de servicio), elija New service role (Nuevo rol de servicio).

> **Note**
>
> Anote el nombre de su rol de servicio. CodeBuild Necesitará el nombre del rol para el paso final de este tutorial.

g. En Buildspec, para Build specifications (Especificaciones de la compilación), elija Insert build commands (Insertar comandos de compilación). Elija Cambiar a editor y pegue lo siguiente en Comandos de compilación:

```
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. Elige Continuar hasta CodePipeline. Esto vuelve a la CodePipeline consola y crea un CodeBuild proyecto que usa tus comandos de compilación para la configuración. El proyecto de compilación usa un rol de servicio para administrar Servicio de AWS los permisos. Es posible que este paso tarde un par de minutos.

i. Elija Siguiente.

10. En Paso 5: agregación de la etapa de prueba, elija Omitir la etapa de prueba y, a continuación, acepte el mensaje de advertencia eligiendo Omitir una vez más.

    Elija Siguiente.

11. En la página Paso 6: Agregar la fase de implementación, elija Omitir la etapa de implementación y, a continuación, acepte el mensaje de advertencia seleccionando nuevamente Omitir. Elija Siguiente.

12. En el paso 7: revisar, selecciona Crear canalización.

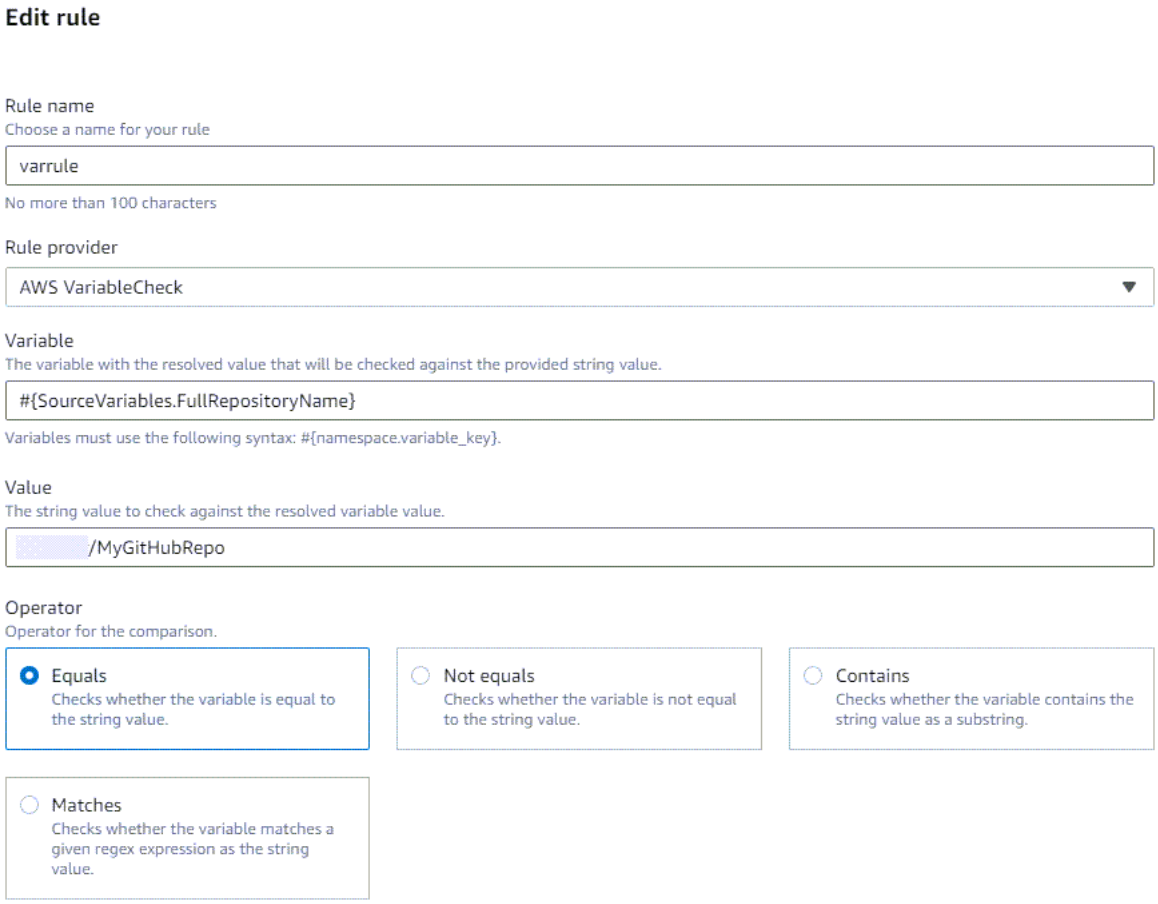
## Paso 2: edición de la etapa de compilación para agregar la condición y la regla

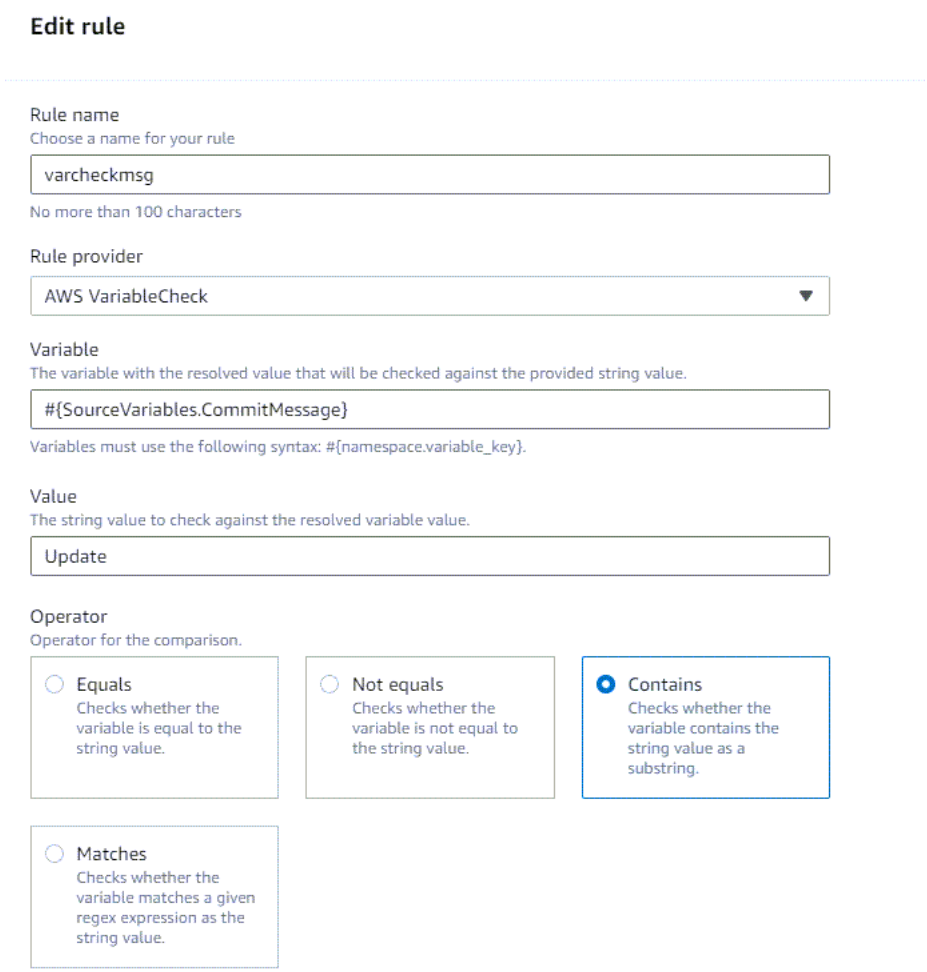
En este paso, va a editar la etapa para agregar una condición de entrada a la regla de verificación de variables.

1. Seleccione su canalización y, a continuación, elija Editar. Elija agregar una regla de entrada en la etapa de compilación.

   En Proveedor de reglas, elige VariableCheck.

2. En Variable, introduzca la variable o variables que desee verificar. En Valor, introduzca el valor de la cadena para compararlo con la variable resuelta. En las siguientes pantallas de ejemplo, se crea una regla para una verificación “igual a” y otra regla para una verificación “contiene”.





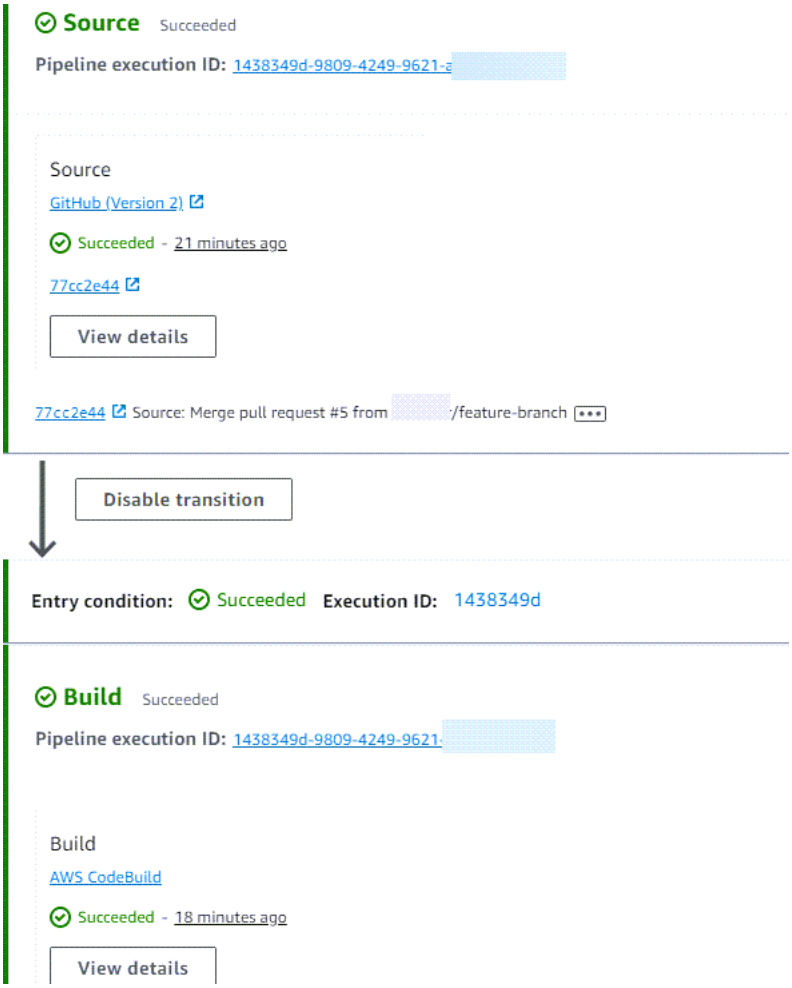

3. Seleccione Save.

   Seleccione Listo.

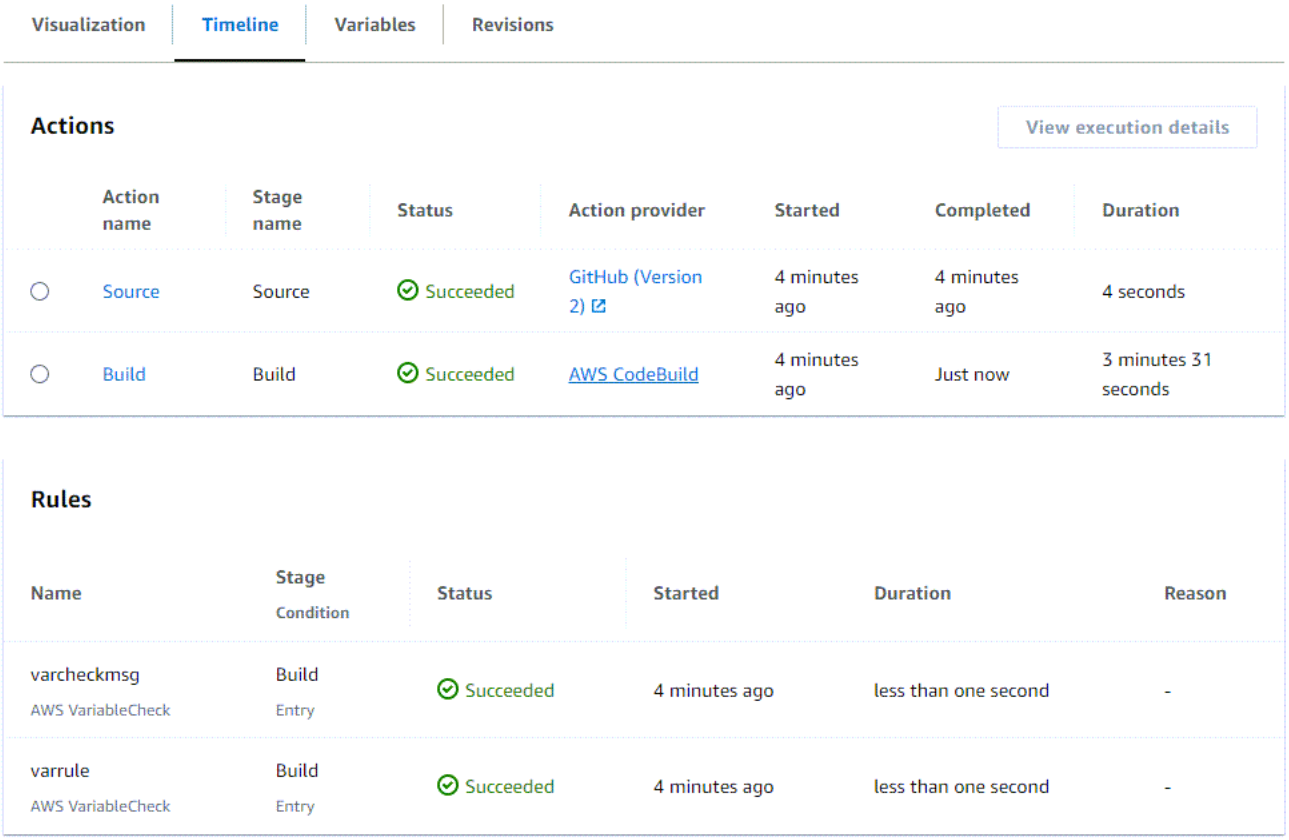
## Paso 3: ejecución de la canalización y visualización de las variables resueltas

En este paso, va a ver los valores resueltos y los resultados de la regla de verificación de variables.

1. Vea la ejecución resuelta una vez que la verificación de reglas se haya realizado correctamente, tal y como se muestra en el siguiente ejemplo.




2. Vea la información de las variables en la pestaña Línea temporal.


Visualization
Timeline
Variables
Revisions
Actions
View execution details
Action name
Stage name
Status
Action provider
Started
Completed
Duration
Source
Source
Succeeded
GitHub (Version 2)
4 minutes ago
4 minutes ago
4 seconds
Build
Build
Succeeded
AWS CodeBuild
4 minutes ago
Just now
3 minutes 31 seconds
Rules
Name
Stage
Condition
Status
Started
Duration
Reason
varcheckmsg
AWS VariableCheck
Build
Entry
Succeeded
4 minutes ago
less than one second
-
varrule
AWS VariableCheck
Build
Entry
Succeeded
4 minutes ago
less than one second
-

# CodePipeline casos de uso

En las siguientes secciones se describen los casos de uso de CodePipeline.

Temas


- Casos de uso para CodePipeline


## Casos de uso para CodePipeline

Puede crear canalizaciones que se integren con otras Servicios de AWS. Pueden ser Servicios de AWS, como Amazon S3, o productos de terceros, como GitHub. En esta sección se proporcionan ejemplos CodePipeline para automatizar las versiones de código mediante distintas integraciones de productos. Para obtener una lista completa de las integraciones CodePipeline organizadas por tipo de acción, consulte. CodePipeline referencia de estructura de tuberías

Temas


- Úselo CodePipeline con Amazon S3, AWS CodeCommit, y AWS CodeDeploy
- Úselo CodePipeline con proveedores de acciones de terceros (GitHub y Jenkins)
- Se usa CodePipeline para compilar, compilar y probar código con CodeBuild
- Úselo CodePipeline con Amazon ECS para la entrega continua de aplicaciones basadas en contenedores a la nube
- Úselo CodePipeline con Elastic Beanstalk para la entrega continua de aplicaciones web a la nube
- Úselo con CodePipeline AWS Lambda para la entrega continua Lambda-based de aplicaciones sin servidor
- Úselo CodePipeline con CloudFormation plantillas para la entrega continua a la nube


### Úselo CodePipeline con Amazon S3, AWS CodeCommit, y AWS CodeDeploy

Cuando crea una canalización, CodePipeline se integra con AWS productos y servicios que actúan como proveedores de acciones en cada etapa de la canalización. Cuando elija etapas en el asistente, debe elegir una etapa de origen y al menos una etapa de compilación o implementación. El asistente crea las etapas automáticamente con nombres predeterminados que no se pueden

cambiar. Estos son los nombres de etapa creados al configurar una canalización completa de tres etapas en el asistente:

- Una etapa de acción de origen con un nombre predeterminado de “Source”.
- Una etapa de acción de compilación con un nombre predeterminado de “Build“.
- Una etapa de acción de implementación con un nombre predeterminado de “Staging“ (Ensayo).

Puede utilizar los tutoriales de esta guía para crear canalizaciones y especificar etapas:

- Los pasos en Tutorial: Crear una canalización simple (bucket de S3) le ayudan a utilizar el asistente para crear una canalización con dos etapas predeterminadas: “Source“ (origen) y “Staging“ (Ensayo), donde su repositorio de Amazon S3 es el proveedor de origen. En este tutorial se crea una canalización que se utiliza AWS CodeDeploy para implementar una aplicación de muestra desde un bucket de Amazon S3 en instancias de Amazon EC2 que ejecutan Amazon Linux.
- Los pasos que se indican le Tutorial: Crear una canalización simple (CodeCommit repositorio) ayudan a usar el asistente para crear una canalización con una etapa de «código fuente» que utilice su AWS CodeCommit repositorio como proveedor de código fuente. En este tutorial se crea una canalización que se utiliza AWS CodeDeploy para implementar una aplicación de muestra desde un AWS CodeCommit repositorio en una instancia de Amazon EC2 que ejecute Amazon Linux.

## Úselo CodePipeline con proveedores de acciones de terceros (GitHub y Jenkins)

Puedes crear canalizaciones que se integren con productos de terceros, como Jenkins GitHub . Los pasos de Tutorial: Crear una canalización de cuatro etapas le muestran cómo crear una canalización que:

- Obtiene el código fuente de un repositorio, GitHub
- Utilice Jenkins para crear y probar el código fuente,
- Se utiliza AWS CodeDeploy para implementar el código fuente creado y probado en instancias de Amazon EC2 que ejecutan Amazon Linux o Microsoft Windows Server.

## Se usa CodePipeline para compilar, compilar y probar código con CodeBuild

CodeBuild es un servicio de compilación gestionado en la nube que le permite crear y probar su código sin necesidad de un servidor o un sistema. Úselo CodePipeline con CodeBuild para automatizar la ejecución de revisiones durante el proceso de entrega continua de compilaciones de software siempre que se produzca un cambio en el código fuente. Para obtener más información, consulte Usar CodePipeline con CodeBuild para probar el código y ejecutar compilaciones.

## Úselo CodePipeline con Amazon ECS para la entrega continua de aplicaciones basadas en contenedores a la nube

Amazon ECS es un servicio de administración de contenedores que le permite implementar aplicaciones basadas en contenedor en instancias de Amazon ECS en la nube. Úselo CodePipeline con Amazon ECS para automatizar la ejecución de revisiones a lo largo del proceso para el despliegue continuo de aplicaciones basadas en contenedores siempre que se produzca un cambio en el repositorio de imágenes de origen. Para obtener más información, consulte Tutorial: Implementación continua con CodePipeline.

## Úselo CodePipeline con Elastic Beanstalk para la entrega continua de aplicaciones web a la nube

Elastic Beanstalk es un servicio informático que le permite implementar servicios y aplicaciones web para servidores web. Úselo CodePipeline con Elastic Beanstalk para el despliegue continuo de aplicaciones web en su entorno de aplicaciones. También se puede utilizar AWS CodeStar para crear una canalización con una acción de despliegue de Elastic Beanstalk.

## Úselo con CodePipeline AWS Lambda para la entrega continua Lambda-based de aplicaciones sin servidor

Puede usarlo AWS Lambda con CodePipeline para invocar una AWS Lambda función, tal y como se describe en Implementación de aplicaciones sin servidor. También puede usar AWS Lambda y AWS CodeStar crear una canalización para implementar aplicaciones sin servidor.

# Úselo CodePipeline con CloudFormation plantillas para la entrega continua a la nube

Puede utilizarlas CodePipeline para CloudFormation la entrega continua y la automatización. Para obtener más información, consulte Entrega continua con CodePipeline. CloudFormation también se utiliza para crear las plantillas de las canalizaciones creadas en AWS CodeStar.

# Úselo CodePipeline con Amazon Virtual Private Cloud

AWS CodePipeline ahora es compatible con los puntos de conexión Amazon Virtual Private Cloud (Amazon VPC) con la tecnología de. AWS PrivateLink Esto significa que puede conectarse directamente a CodePipeline través de un punto final privado en su VPC, manteniendo todo el tráfico dentro de su VPC y de la red. AWS

Amazon VPC es una Servicio de AWS que puede utilizar para lanzar AWS recursos en una red virtual que usted defina. Con una VPC, tiene control sobre los ajustes de red, como por ejemplo:

- Rango de direcciones IP
- Subredes
- Tablas de enrutamiento
- Gateways de red

Los puntos finales de VPC de interfaz cuentan con una AWS tecnología que facilita la comunicación privada entre el Servicios de AWS uso de una interfaz de red elástica con direcciones IP privadas. AWS PrivateLink Para conectar su VPC CodePipeline, debe definir un punto final de VPC de interfaz para. CodePipeline Este tipo de punto de conexión le permite conectar su VPC a servicios de Servicios de AWS. El punto final proporciona una conectividad fiable y escalable CodePipeline sin necesidad de una puerta de enlace a Internet, una instancia de traducción de direcciones de red (NAT) ni una conexión VPN. Para obtener más información acerca de cómo configurar una VPC, consulte la Guía del usuario de VPC.

## Disponibilidad.

CodePipeline actualmente es compatible con los puntos finales de VPC de las siguientes maneras: Regiones de AWS

- Este de EE. UU. (Ohio)
- Este de EE. UU. (Norte de Virginia)
- Oeste de EE. UU. (Norte de California)
- Oeste de EE. UU. (Oregón)
- Canadá (centro)
- Europa (Fráncfort)

- Europa (Irlanda)
- Europa (Londres)
- Europa (Milán)*
- Europa (París)
- Europa (Estocolmo)
- Asia-Pacífico (Hong Kong)*
- Asia-Pacífico (Mumbai)
- Asia-Pacífico (Tokio)
- Asia-Pacífico (Seúl)
- Asia-Pacífico (Singapur)
- Asia-Pacífico (Sídney)
- América del Sur (São Paulo)
- AWS GovCloud (EE. UU.-Oeste)

* Debe habilitar esta región antes de poder utilizarla.

## Cree un punto final de VPC para CodePipeline

Puede utilizar la consola de Amazon VPC para crear com.amazonaws. ***region***Punto final de VPC .codepipeline. En la consola, *region* es el identificador de región de una región Región de AWS compatible con CodePipeline, por ejemplo, la región EE.UU. Este (Ohio). `us-east-2` Para obtener más información, consulte Creación de un punto de conexión de tipo interfaz en la Guía del usuario de Amazon VPC.

El punto de conexión ya contiene la región que especificó al iniciar sesión en AWS. Si inicia sesión en otra región, el punto de enlace de la VPC se actualizará con la nueva región.

> **Note**
>
> Es posible Servicios de AWS que otras que ofrezcan compatibilidad con VPC y se integren con ellas CodePipeline, por ejemplo, no admitan el uso de puntos de enlace de Amazon VPC para esa integración. CodeCommit Por ejemplo, el tráfico entre CodePipeline y CodeCommit no se puede restringir al rango de subredes de la VPC.

# Solución de problemas con la configuración de la VPC

Al solucionar problemas con la VPC, utilice la información que aparece en los mensajes de error sobre la conexión a Internet para ayudarle a identificar, diagnosticar y resolver los problemas.

1. Asegúrese de que la gateway de Internet está asociada a su VPC.
2. Asegúrese de que la tabla de enrutamiento de su subred pública apunta a la gateway de Internet.
3. Asegúrese de que su red ACLs permita que el tráfico fluya.
4. Asegúrese de que los grupos de seguridad permiten el tráfico.
5. Asegúrese de que la tabla de enrutamiento de las subredes privadas apunta a la puerta de enlace privada virtual.
6. Asegúrese de que la función de servicio utilizada por CodePipeline tenga los permisos adecuados. Por ejemplo, si CodePipeline no tiene los permisos de Amazon EC2 necesarios para trabajar con una Amazon VPC, es posible que reciba un mensaje de error que diga: «Error de EC2 inesperado:». UnauthorizedOperation

# Defina CI/CD las canalizaciones con etapas y acciones

Para definir un proceso de publicación automatizado AWS CodePipeline, debe crear una canalización, que es una construcción de flujo de trabajo que describe la forma en que los cambios de software pasan por un proceso de publicación. Una canalización se compone de etapas y acciones que usted configura.

> **Note**
>
> Al añadir etapas de compilación, implementación, prueba o invocación, además de las opciones predeterminadas incluidas CodePipeline, puede elegir acciones personalizadas que ya haya creado para usarlas con sus canalizaciones. Las acciones personalizadas se pueden utilizar para tareas como ejecutar un proceso de creación desarrollado internamente o un conjunto de pruebas. Se incluyen identificadores de versión para ayudarle a diferenciar las distintas versiones de una acción personalizada en las listas de proveedores. Para obtener más información, consulte Crea y añade una acción personalizada en CodePipeline.

Para crear una canalización, primero debe completar los pasos descritos en Empezar con CodePipeline.

Para obtener más información sobre las canalizaciones, consulte CodePipeline conceptosCodePipeline tutoriales, y, si quiere utilizarlas AWS CLI para crear una canalización,. Creación de una canalización, etapas y acciones Para ver una lista de canalizaciones, consulte Vea las canalizaciones y los detalles en CodePipeline.

Temas


- Creación de una canalización, etapas y acciones
- Editar una canalización en CodePipeline
- Vea las canalizaciones y los detalles en CodePipeline
- Eliminar una canalización en CodePipeline
- Cree una canalización en la CodePipeline que se utilicen los recursos de otra AWS inscrita
- Migrar los canales de sondeo para utilizar la detección de cambios basada en eventos
- Crear el rol CodePipeline de servicio
- Etiquetado de recursos

- Etiquetar una canalización CodePipeline
- Creación de una regla de notificación

# Creación de una canalización, etapas y acciones

Puede utilizar la AWS CodePipeline consola o la AWS CLI para crear una canalización. Las canalizaciones deben tener dos etapas como mínimo. La primera etapa de una canalización debe ser una etapa de origen. La canalización debe tener al menos otra etapa que sea una etapa de compilación o implementación.

> **Important**
>
> Como parte de la creación de una canalización, se utilizará un depósito de artefactos de S3 proporcionado por el cliente CodePipeline para los artefactos. (Es diferente del bucket que se usa para una acción de origen de S3). Si el depósito de artefactos de S3 está en una cuenta diferente a la de tu canalización, asegúrate de que el depósito de artefactos de S3 pertenezca a una Cuentas de AWS persona segura y fiable.

Puedes añadir acciones a tu canalización que estén en una canalización Región de AWS diferente a la tuya. Una acción interregional es aquella en la que un Servicio de AWS es el proveedor de una acción y el tipo de acción o el tipo de proveedor se encuentra en una AWS región diferente a la de tu cartera. Para obtener más información, consulte Agregue una acción interregional en CodePipeline.

También puede crear canalizaciones que creen e implementen aplicaciones basadas en contenedores utilizando Amazon ECS como proveedor de implementación. Antes de crear una canalización que implemente aplicaciones basadas en contenedores con Amazon ECS, debe crear un archivo de definiciones de imágenes, tal y como se describe en Referencia del archivo de definiciones de imágenes.

CodePipeline utiliza métodos de detección de cambios para iniciar tu canalización cuando se introduce un cambio en el código fuente. Estos métodos de detección se basan en el tipo de código fuente:

- CodePipeline usa Amazon CloudWatch Events para detectar cambios en el repositorio y la rama de CodeCommit origen o en el bucket de código fuente de S3.

> **Note**
>
> Si utiliza la consola para crear o editar una canalización, los recursos de detección de cambios se crean automáticamente. Si usa la AWS CLI para crear la canalización, debe crear los recursos adicionales usted mismo. Para obtener más información, consulte CodeCommit acciones de origen y EventBridge.

Temas


- Creación de una canalización personalizada (consola)
- Crear una canalización (CLI)
- Creación de una canalización a partir de plantillas estáticas


## Creación de una canalización personalizada (consola)

Para crear una canalización personalizada en la consola, necesita proporcionar la ubicación del archivo de código fuente e información acerca de los proveedores que utilizará para las acciones.

Cuando se usa la consola para crear una canalización, se debe incluir una etapa de código fuente y una de las siguientes etapas, o ambas:

- Una etapa de compilación.
- Una etapa de implementación.

Cuando utiliza el asistente de canalización, CodePipeline crea los nombres de las etapas (fuente, compilación, puesta en escena). Estos nombres no se pueden cambiar. Puede usar nombres más específicos (por ejemplo, BuildToGamma o DeployToProd) para las etapas que añada más adelante.

Paso 1: Crear la canalización y asignarle un nombre

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En la página Welcome, elija Create pipeline.

   Si es la primera vez que la usas CodePipeline, selecciona Comenzar.

3. En la página Paso 1: elección de la opción de creación, en Opciones de creación, seleccione la opción Crear una canalización personalizada. Elija Siguiente.
4. En la página Paso 2: elegir la configuración de la canalización, en Nombre de la canalización, escriba el nombre de la canalización.

   En una sola AWS cuenta, cada canalización que crees en una AWS región debe tener un nombre único. Los nombres se pueden reutilizar para canalizaciones de regiones diferentes.

   > **Note**
   >
   > Después de crear una canalización, no podrá cambiar el nombre. Para obtener información acerca de otras limitaciones, consulte Cuotas en AWS CodePipeline.

5. En Tipo de canalización, elija una de las siguientes opciones: Los tipos de canalización difieren en características y precio. Para obtener más información, consulte Tipos de canalización.

   - Las canalizaciones de tipo V1 tienen una estructura JSON que contiene parámetros estándares de canalización, etapa y nivel de acción.
   - Las canalizaciones de tipo V2 tienen la misma estructura que las de tipo V1, además de compatibilidad con parámetros adicionales, como los desencadenadores en las etiquetas de Git y las variables a nivel de canalización.

6. En Service role (Rol de servicio), realice una de las operaciones siguientes:

   - Elija Nueva función de servicio para CodePipeline poder crear una nueva función de servicio en IAM.
   - Elija Existing service role (Rol de servicio existente) para utilizar un rol de servicio que ya se ha creado en IAM. En Role ARN (ARN del rol), elija en la lista el ARN del rol de servicio.

   > **Note**
   >
   > En función de cuándo se creó el rol de servicio, es posible que tenga que actualizar sus permisos para admitir otros Servicios de AWS. Para obtener información, consulte Agrega permisos al rol de CodePipeline servicio.

   Para obtener más información acerca del rol de servicio y su instrucción de política, consulte Administre la función CodePipeline de servicio.

7. (Opcional) En Variables, seleccione Añadir variable para añadir variables a nivel de canalización.

   Para obtener más información acerca de las variables a nivel de canalización, consulte Referencia de variables. Para ver un tutorial con una variable a nivel de canalización que se transfiere en el momento de la ejecución de la canalización, consulte Tutorial: Uso de variables a nivel de canalización.

   > **Note**
   >
   > Si bien es opcional añadir variables a nivel de canalización, en el caso de una canalización especificada con variables a nivel de canalización en la que no se proporcionen valores, la ejecución de la canalización fallará.

8. (Opcional) Expanda Advanced settings (Configuración avanzada).
9. En Artifact store (Almacén de artefactos), ejecute una de estas acciones:

   a. Elige la ubicación predeterminada para usar el almacén de artefactos predeterminado, como el depósito de artefactos de S3 designado como predeterminado, para tu canalización en la Región de AWS que hayas seleccionado para tu canalización.

   b. Elija Custom location (Ubicación personalizada) si ya dispone de un almacén de artefactos (por ejemplo, un bucket de artefactos de S3) en la misma región que la canalización. En Bucket, elija el nombre del bucket.

   > **Note**
   >
   > Este no es el bucket de origen para su código fuente. Este es el almacén de artefactos de la canalización. Cada canalización debe tener su propio almacén de artefactos independiente, como un bucket de S3. Al crear o editar una canalización, debes tener una cubeta de artefactos en la región de la canalización y una cubeta de artefactos por cada AWS región en la que vayas a ejecutar una acción.
   > Para obtener más información, consulte Artefactos de entrada y salida y CodePipeline referencia de estructura de tuberías.

10. (Opcional) En Encryption key (Clave de cifrado), realice una de las siguientes operaciones:

a. Para usar el CodePipeline valor predeterminado AWS KMS key para cifrar los datos del almacén de artefactos de la canalización (depósito S3), elige la clave gestionada predeterminada. AWS

b. Si desea utilizar su clave administrada por el cliente para cifrar los datos del almacén de artefactos de canalización (bucket de S3), elija Clave administrada por el cliente. Elija el ID de clave, el ARN de clave o el ARN de alias.

11. Elija Siguiente.

## Paso 2: Crear una etapa de código fuente

1. En la página Paso 3: agregar la etapa de origen, en Proveedor de origen, elija el tipo de repositorio en el que se almacena el código fuente y, a continuación, especifique las opciones obligatorias. Los campos adicionales se muestran según el proveedor de origen seleccionado de la siguiente manera.

   - Para Bitbucket Cloud GitHub (mediante una GitHub aplicación), GitHub Enterprise Server o GitLab.com autogestionado: GitLab

     1. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para tu acción de GitHub origen, consulta. GitHub conexiones
     2. Elija el repositorio que desea usar como ubicación de origen para la canalización.

        Elija agregar un desencadenador o filtrar los tipos de desencadenadores para iniciar la canalización. Para obtener más información acerca de cómo trabajar con desencadenadores, consulte Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código. Para obtener más información acerca de filtros con patrones de glob, consulte Trabajar con patrones glob en la sintaxis.
     3. En Formato del artefacto de salida, debe elegir el formato de los artefactos.
        - Para almacenar los artefactos de salida de la GitHub acción mediante el método predeterminado, elija CodePipelinedefault. La acción accede a los archivos del GitHub repositorio y almacena los artefactos en un archivo ZIP en el almacén de artefactos de Pipeline.
        - Para almacenar un archivo JSON que contiene una referencia URL al repositorio de manera que las acciones posteriores puedan ejecutar comandos Git directamente, elija Clonación completa. Esta opción solo la pueden utilizar las acciones posteriores de CodeBuild .

Si eliges esta opción, tendrás que actualizar los permisos de tu rol de servicio del CodeBuild proyecto, tal y como se muestra en la siguiente. Solución de problemas CodePipeline Para ver un tutorial que muestra cómo utilizar la opción Clonación completa, consulte Tutorial: Utilice un clon completo con una fuente de GitHub canalización.

- Para Amazon S3:

  1. En Amazon S3 location (Ubicación de Amazon S3), proporcione el nombre del bucket de S3 y la ruta del objeto de un bucket con control de versiones habilitado. El formato del nombre de bucket y la ruta tiene un aspecto similar a este:

     ```
     s3://bucketName/folderName/objectName
     ```

     > **Note**
     >
     > Si Amazon S3 es el proveedor de origen de la canalización, debe comprimir los archivos de origen en un solo archivo.zip y cargarlo en el bucket de origen. También puede cargar un archivo sin comprimir; sin embargo, se producirá un error en las acciones posteriores que esperan un archivo.zip.

  2. Tras elegir el bucket de origen de S3, CodePipeline crea la regla de Amazon CloudWatch Events y la AWS CloudTrail ruta que se va a crear para esta canalización. En Change detection options, deje los valores predeterminados. Esto te CodePipeline permite usar Amazon CloudWatch Events y AWS CloudTrail detectar cambios en tu nueva canalización. Elija Siguiente.

- Para AWS CodeCommit:

  - En Nombre del repositorio, elige el nombre del CodeCommit repositorio que quieres usar como ubicación de origen de tu canalización. En Branch name, en la lista desplegable, elija la ramificación que desee usar.
  - En Formato del artefacto de salida, debe elegir el formato de los artefactos.
    - Para almacenar los artefactos de salida de la CodeCommit acción mediante el método predeterminado, selecciona CodePipelinepredeterminado. La acción accede a los archivos del CodeCommit repositorio y almacena los artefactos en un archivo ZIP en el almacén de artefactos de Pipeline.
    - Para almacenar un archivo JSON que contiene una referencia URL al repositorio de manera que las acciones posteriores puedan ejecutar comandos Git directamente, elija

Clonación completa. Esta opción solo la pueden utilizar las acciones posteriores de CodeBuild .

Si eliges esta opción, tendrás que añadir el `codecommit:GitPull` permiso a tu función de CodeBuild servicio, tal y como se muestra en la siguiente. Agregue CodeBuild GitClone permisos para las acciones CodeCommit de origen También tendrá que añadir los `codecommit:GetRepository` permisos a su función de CodePipeline servicio, como se muestra enAgrega permisos al rol de CodePipeline servicio. Para ver un tutorial que muestra cómo utilizar la opción Clonación completa, consulte Tutorial: Utilice un clon completo con una fuente de GitHub canalización.

- Tras elegir el nombre y la sucursal del CodeCommit repositorio, aparece un mensaje en las opciones de detección de cambios en el que se muestra la regla de Amazon CloudWatch Events que se va a crear para esta canalización. En Change detection options, deje los valores predeterminados. Esto te CodePipeline permite usar Amazon CloudWatch Events para detectar cambios en tu nueva canalización.

- Para Amazon ECR:
  - En Nombre de repositorio, elija el nombre de su repositorio de Amazon ECR.
  - En Image tag (Etiqueta de imagen), especifique el nombre de la imagen y la versión, si es diferente de LATEST.
  - En Output Artifacts, elija el artefacto de salida predeterminado, por ejemplo MyApp, el que contenga el nombre de la imagen y la información del URI del repositorio que desee utilizar en la siguiente etapa.

    Para ver un tutorial sobre cómo crear una canalización para Amazon ECS con implementaciones CodeDeploy azul-verde que incluya una etapa de origen de Amazon ECR, consulte. Tutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR

  Cuando se incluye una etapa de origen de Amazon ECR en la canalización, la acción de origen genera un archivo `imageDetail.json` como artefacto de salida al confirmar un cambio. Para obtener más información sobre el archivo `imageDetail.json`, consulte Detail.json archivo de imagen para las acciones de blue/green implementación de Amazon ECS.

> **Note**
>
> El objeto y el tipo de archivo deben ser compatibles con el sistema de despliegue que vaya a utilizar (por ejemplo, Elastic Beanstalk o). CodeDeploy Algunos tipos de archivos compatibles son .zip, .tar, y .tgz. Para obtener más información acerca de los tipos de contenedores compatibles con Elastic Beanstalk, consulte Personalización y configuración de entornos y Plataformas compatibles. Para obtener más información sobre cómo implementar las revisiones con CodeDeploy, consulte Cargar la revisión de su aplicación y Preparar una revisión.

2. Para configurar la etapa para el reintento automático, seleccione Habilitar el reintento automático en caso de fallo de una etapa. Para obtener más información acerca del reintento automático, consulte Configuración de una etapa para el reintento automático en caso de fallo.
3. Elija Siguiente.

### Paso 4: creación de una etapa de compilación

Este paso es opcional si tiene previsto crear una etapa de implementación.

1. En la página Paso 4: agregación de una etapa de compilación, realice una de las siguientes acciones y, después, elija Siguiente:
   - Elija Omitir etapa de compilación si tiene previsto crear una etapa de prueba o implementación.
   - Para elegir la acción de Comandos para la etapa de compilación, elija Comandos.

     > **Note**
     >
     > Si ejecuta la acción de Comandos, se le cobrarán cargos por separado en AWS CodeBuild. Si planea insertar comandos de compilación como parte de una CodeBuild acción, elija Otros proveedores de compilación y, a continuación, seleccione CodeBuild.

En Comandos, introduzca los comandos del intérprete de comandos para su acción. Para obtener más información acerca de la acción de Comandos, consulte Referencia de la acción de Comandos.

- Para elegir otros proveedores de compilación CodeBuild, por ejemplo, elija Otros proveedores. En Build provider (Proveedor de compilación), elija el proveedor de acciones personalizadas de servicios de compilación y proporcione los detalles de la configuración para ese proveedor. Si desea ver un ejemplo de cómo añadir Jenkins como proveedor de compilación, consulte Tutorial: Crear una canalización de cuatro etapas.
- En Build provider (Proveedor de compilación), elija AWS CodeBuild.

  En Región, elige la AWS región en la que se encuentra el recurso. El campo Región designa dónde se crean los AWS recursos para este tipo de acción y tipo de proveedor. Este campo solo se muestra en el caso de las acciones en las que el proveedor de la acción es un Servicio de AWS. El campo Región se establece de forma predeterminada en la misma AWS región que tu canalización.

  En Project name (Nombre del proyecto), elija el proyecto de compilación. Si ya ha creado un proyecto de construcción en CodeBuild, elíjalo. O bien, puede crear un proyecto de construcción CodeBuild y, a continuación, volver a esta tarea. Siga las instrucciones de Crear una canalización que use CodeBuild en la Guía del usuario de CodeBuild.

  En las especificaciones de compilación, el archivo CodeBuild buildspec es opcional y, en su lugar, puede introducir comandos. En Insertar comandos de compilación, especifique los comandos del intérprete de comandos que correspondan a su acción. Para obtener más información acerca de las consideraciones para el uso de comandos de compilación, consulte Referencia de la acción de Comandos. Elija Usar un archivo buildspec si desea ejecutar comandos en otras fases o si tiene una lista larga de comandos.

  En Variables de entorno, para añadir variables de CodeBuild entorno a la acción de compilación, selecciona Añadir variable de entorno. Cada variable se compone de tres entradas:

  - En Name (Nombre), escriba el nombre de la clave o de la variable de entorno.
  - En Value (Valor), escriba el valor de la variable de entorno. Si elige Parámetro para el tipo de variable, asegúrese de que este valor sea el nombre de un parámetro que ya haya almacenado en el Almacén de parámetros de AWS Systems Manager.

> **Note**
>
> Se desaconseja encarecidamente el uso de variables de entorno para almacenar valores confidenciales, especialmente AWS las credenciales. Cuando utiliza la CodeBuild consola o la AWS CLI, las variables de entorno se muestran en texto plano. Para valores confidenciales, se recomienda utilizar el tipo Parameter (Parámetro) en su lugar.

- (Opcional) En Type (Tipo), especifique el tipo de variable de entorno. Los valores válidos son Texto sin formato o Parámetro. El valor predeterminado es Texto sin formato.

(Opcional) En Tipo de compilación, seleccione una de las siguientes opciones:

- Para ejecutar cada compilación en una sola ejecución de acción de compilación, elija Compilación única.
- Para ejecutar varias compilaciones en la misma ejecución de acción de compilación, elija Compilación por lote.

(Opcional) Si opta por ejecutar compilaciones por lotes, puede elegir Combinar todos los artefactos del lote en una sola ubicación para colocar todos los artefactos de compilación en un único artefacto de salida.

2. Para configurar la etapa para el reintento automático, seleccione Habilitar el reintento automático en caso de fallo de una etapa. Para obtener más información acerca del reintento automático, consulte Configuración de una etapa para el reintento automático en caso de fallo.
3. Elija Siguiente.

### Paso 5: creación de una etapa de prueba

Este paso es opcional si tiene previsto crear una etapa de compilación o implementación.

1. En la página Paso 5: agregación de una etapa de prueba, realice una de las siguientes acciones y después elija Siguiente:
   - Elija Omitir etapa de prueba si tiene previsto crear una etapa de implementación.
   - En Proveedor de la prueba, elija el proveedor de acciones de prueba y complete los campos correspondientes.
2. Elija Siguiente.

### Paso 6: Crear una etapa de despliegue

Este paso es opcional si ya ha creado una etapa de compilación.

1. En la página Paso 6: agregación de una etapa de implementación, realice una de las siguientes acciones y después elija Siguiente:

   - Elija Omitir etapa de implementación si ha creado una etapa de compilación o de prueba en los pasos anteriores.

     > **Note**
     >
     > Esta opción no aparece si ya ha omitido la etapa de compilación o de prueba.

   - En Deploy provider (Proveedor de implementación), elija una acción personalizada que haya creado para un proveedor de implementación.

     En Región, solo para acciones entre regiones, elija la AWS región en la que se crea el recurso. El campo Región designa dónde se crean los recursos de AWS para este tipo de acción y de proveedor. Este campo solo se muestra en el caso de las acciones en las que el proveedor de la acción es un Servicio de AWS. El campo Región se establece de forma predeterminada en la misma AWS región que tu canalización.

   - En Deploy provider (Proveedor de implementación), hay campos disponibles para los proveedores predeterminados, tal como se indica a continuación:

     - CodeDeploy

       En Nombre de la aplicación, ingresa o elige el nombre de una CodeDeploy aplicación existente. En Deployment group (Grupo de implementaciones), escriba el nombre de un grupo de implementaciones para la aplicación. Elija Siguiente. También puede crear una aplicación, un grupo de implementaciones o ambas cosas en la consola de CodeDeploy.

     - AWS Elastic Beanstalk

       En Nombre de aplicación, escriba o seleccione el nombre de una aplicación de Elastic Beanstalk disponible. En Environment name (Nombre de entorno), especifique un entorno para la aplicación. Elija Siguiente. También puede crear una aplicación, un entorno o ambas cosas en la consola de Elastic Beanstalk.

     - AWS OpsWorks Stacks

En Stack (Pila), escriba o elija el nombre de la pila que desea utilizar. En Layer, elija la capa a la que pertenecen las instancias de destino. En Aplicación, elija la aplicación que desea actualizar e implementar. Si necesita crear una aplicación, elija Crear una nueva en AWS OpsWorks.

Para obtener información sobre cómo añadir una aplicación a una pila y a una capa AWS OpsWorks, consulte Añadir aplicaciones en la Guía del AWS OpsWorks usuario.

Para ver un ejemplo completo de cómo utilizar una canalización simple CodePipeline como fuente del código que se ejecuta en AWS OpsWorks capas, consulte Utilización CodePipeline con AWS OpsWorks Stacks.

- AWS CloudFormation

  Realice una de las siguientes acciones:

  - En el modo Acción, elija Crear o actualizar una pila, introduzca un nombre de pila y un nombre de archivo de plantilla y, a continuación, elija el nombre del rol que AWS CloudFormation desee asumir. Si lo prefiere, escriba el nombre de un archivo de configuración y elija una opción de capacidad de IAM.
  - En el modo Acción, elija Crear o reemplazar un conjunto de cambios, introduzca un nombre de pila y un nombre de conjunto de cambios y, a continuación, elija el nombre del rol que AWS CloudFormation desee asumir. Si lo prefiere, escriba el nombre de un archivo de configuración y elija una opción de capacidad de IAM.

  Para obtener información sobre la integración de AWS CloudFormation funciones en una canalización CodePipeline, consulte Continuous Delivery with CodePipeline en la Guía del AWS CloudFormation usuario.

- Amazon ECS

  En Nombre de clúster, escriba o seleccione el nombre de un clúster de Amazon ECS disponible. En Service name (Nombre de servicio), escriba o elija el nombre del servicio que se ejecuta en el clúster. También puede crear un clúster y un servicio. En Image filename (Nombre de archivo de imagen), escriba el nombre del archivo de definiciones de imágenes que describe el contenedor y la imagen de su servicio.

> **Note**
>
> La acción de implementación de Amazon ECS requiere un archivo `imagedefinitions.json` como entrada. El nombre de archivo predeterminado es imagedefinitions.json. Si utiliza otro nombre de archivo, debe proporcionarlo al crear la etapa de implementación de la canalización. Para obtener más información, consulte Archivo imagedefinitions.json para las acciones de implementación estándar de.

Elija Siguiente.

> **Note**
>
> Asegúrese de que su clúster de Amazon ECS se ha configurado con dos o más instancias. Los clústeres de Amazon ECS deben incluir al menos dos instancias para poder mantener una como la instancia principal y usar la otra para alojar las nuevas implementaciones.

Si desea ver un tutorial sobre la implementación de aplicaciones basadas en contenedores con la canalización, consulte Tutorial: Implementación continua con CodePipeline.

- Amazon ECS (Blue/Green)

  Introduzca el grupo de CodeDeploy aplicaciones y despliegues, la definición de la tarea de Amazon ECS y la información AppSpec del archivo y, a continuación, seleccione Siguiente.

  > **Note**
  >
  > La acción Amazon ECS (Blue/Green) requiere un Detail.json archivo de imagen como artefacto de entrada para la acción de implementación. Dado que la acción de origen de Amazon ECR crea este archivo, no es necesario que las canalizaciones con una acción de origen de Amazon ECR proporcionen un archivo `imageDetail.json`. Para obtener más información, consulte Detail.json archivo de imagen para las acciones de blue/green implementación de Amazon ECS.

Para ver un tutorial sobre cómo crear una canalización para las implementaciones azul-verde en un clúster de Amazon ECS con CodeDeploy, consulte. Tutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR

- AWS Service Catalog

  Elija Enter deployment configuration (Especificar configuración de implementación) si desea utilizar campos de la consola para especificar la configuración, o elija Configuration file (Archivo de configuración) si tiene un archivo de configuración. Especifique la información de configuración y del producto y, a continuación, elija Next (Siguiente).

  Si desea ver un tutorial sobre la implementación de cambios de producto en con la canalización, consulte Tutorial: Crear una canalización que se implemente en Service Catalog.

- Alexa Skills Kit

  En Alexa Skill ID (ID de habilidad de Alexa), escriba el ID de habilidad para la habilidad de Alexa. En Client ID (ID de cliente) y en Client secret (Secreto de cliente), escriba las credenciales generadas con un perfil de seguridad de Login with Amazon (LWA). En Refresh token (Token de actualización), especifique el token de actualización que ha generado mediante el comando de la CLI de ASK para recuperar un token de actualización. Elija Siguiente.

  Si desea ver un tutorial sobre la implementación de habilidades de Alexa con una canalización y sobre cómo generar las credenciales de LWA, consulte Tutorial: Crear una canalización que implemente una habilidad de Amazon Alexa.

- Amazon S3

  En Bucket, escriba el nombre del bucket de S3 que desea utilizar. Elija Extract file before deploy (Extraer el archivo antes de la implementación) si el artefacto de entrada a la etapa de implementación es un archivo ZIP. Si selecciona Extract file before deploy (Extraer el archivo antes de la implementación), es posible que también pueda introducir un valor para la Deployment path (Ruta de la implementación) en la que se descomprimirá el archivo ZIP. Si no está seleccionada, debe introducir un valor en la clave de objeto de S3.

> **Note**
>
> La mayoría de los artefactos de salida de las etapas de código fuente y compilación están comprimidos. Todos los proveedores de origen de la canalización excepto Amazon S3 comprimirán los archivos de origen antes de suministrárselos al artefacto de entrada de la siguiente acción.

(Opcional) En ACL predefinida, introduzca la ACL predefinida para aplicarla al objeto que se implementa en Amazon S3.

> **Note**
>
> La aplicación de una ACL predefinida sobrescribe cualquier ACL existente aplicada al objeto.

(Opcional) En Cache control (Control de caché), especifique los parámetros de control de caché de las solicitudes para descargar objetos del bucket. Para una lista de valores válidos, consulte el `Cache-Control`campo del encabezado para las operaciones HTTP Para introducir varios valores en CacheControl, utilice una coma entre cada valor. Puede añadir un espacio después de cada coma (opcional), tal y como se muestra en este ejemplo.




La entrada de ejemplo anterior se muestra en la CLI de la siguiente manera:

```
"CacheControl": "public, max-age=0, no-transform"
```

Elija Siguiente.

Para obtener un tutorial que explica cómo crear una canalización con un proveedor de acción de implementación de Amazon S3, consulte Tutorial: Crear una canalización que utilice Amazon S3 como proveedor de implementación.

2. Para configurar la etapa para el reintento automático, seleccione Habilitar el reintento automático en caso de fallo de una etapa. Para obtener más información acerca del reintento automático, consulte Configuración de una etapa para el reintento automático en caso de fallo.
3. Para configurar la etapa para la reversión automática, seleccione Configurar la reversión automática en caso de fallo en una etapa. Para obtener más información acerca de la reversión automática, consulte Configuración de una etapa para la reversión automática.
4. Elija Next Step (Paso siguiente).

Paso 7: Revise la canalización

- En la página Paso 7: Revisar, revisa la configuración de la canalización y, a continuación, selecciona Crear canalización para crear la canalización o Anterior para volver atrás y editar las opciones elegidas. Para salir del asistente sin crear una canalización, elija Cancel.

Ahora que ha creado su canalización, puede verla en la consola. La canalización comienza a ejecutarse una vez creada. Para obtener más información, consulte Vea las canalizaciones y los detalles en CodePipeline. Para obtener más información acerca de cómo realizar cambios en la canalización, consulte Editar una canalización en CodePipeline.

## Crear una canalización (CLI)

Para utilizarla AWS CLI para crear una canalización, debes crear un archivo JSON para definir la estructura de la canalización y, a continuación, ejecutar el create-pipeline comando con el `--cli-input-json` parámetro.

> **Important**
>
> No puedes usar el AWS CLI para crear una canalización que incluya las acciones de los socios. En su lugar, debes usar la CodePipeline consola.

Para obtener más información sobre la estructura de la canalización, consulta CodePipeline referencia de estructura de tuberías y create-pipeline en la referencia de la CodePipeline API.

Para crear un archivo JSON, utilice el archivo JSON de canalización de ejemplo, edítelo y, a continuación, llame a ese archivo cuando ejecute el comando create-pipeline.

Requisitos previos:

Necesita el ARN del rol de servicio para CodePipeline el que creó. Empezar con CodePipeline
Al ejecutar el comando, se utiliza el ARN del rol de CodePipeline servicio en el archivo JSON de canalización. create-pipeline Para obtener más información sobre cómo crear un rol de servicio, consulte Crear el rol CodePipeline de servicio. A diferencia de la consola, al ejecutar el create-pipeline comando en el AWS CLI no existe la opción de crear el rol de CodePipeline servicio automáticamente. El rol de servicio debe existir previamente.

Necesita el nombre de un bucket de S3 donde se almacenarán los artefactos de la canalización. Este bucket debe estar en la misma región que la canalización. El nombre del bucket se utiliza en el archivo JSON de la canalización al ejecutar el comando create-pipeline. A diferencia de la consola, al ejecutar el create-pipeline comando en la AWS CLI no se crea un depósito de S3 para almacenar artefactos. El bucket debe existir previamente.

> **Note**
>
> También puede usar el comando get-pipeline para obtener una copia de la estructura JSON de esa canalización y, después, modificar esa estructura en un editor de texto sin formato.

Para crear el archivo JSON

1. En un terminal (Linux, macOS o Unix) o en el símbolo del sistema (Windows), cree un nuevo archivo de texto en un directorio local.
2. (Opcional) Puede agregar una o varias variables a nivel de canalización. Puede hacer referencia a este valor en la configuración de CodePipeline las acciones. Puede añadir los nombres y valores de las variables al crear la canalización, y también puede optar por asignar valores al iniciar la canalización en la consola.

   > **Note**
   >
   > Si bien es opcional añadir variables a nivel de canalización, en el caso de una canalización especificada con variables a nivel de canalización en la que no se proporcionen valores, la ejecución de la canalización fallará.

   Una variable a nivel de canalización se resuelve en el tiempo de ejecución de la canalización. Todas las variables son inmutables, lo que significa que no se pueden actualizar después de

asignar un valor. Las variables a nivel de canalización con valores resueltos se mostrarán en el historial de cada ejecución.

Las variables se proporcionan a nivel de canalización mediante el atributo de variables de la estructura de canalización. En el siguiente ejemplo, la variable `Variable1` tiene un valor de `Value1`.

```
      "variables": [
           {
               "name": "Timeout",
               "defaultValue": "1000",
               "description": "description"
           }
       ]
```

Agrega esta estructura al JSON de su canalización o al JSON de ejemplo en el siguiente paso. Para obtener más información acerca de las variables, incluida la información de los espacios de nombres, consulte Referencia de variables.

3. Abra el archivo en un editor de texto sin formato y edite los valores para que reflejen la estructura que desea crear. Como mínimo, deberá cambiar el nombre de la canalización. Considere también la posibilidad de cambiar:

   - El bucket de S3 en el que se almacenan los artefactos de esta canalización.
   - La ubicación de código fuente del código.
   - El proveedor de implementación.
   - Cómo desea implementar el código.
   - Las etiquetas de la canalización.

   La siguiente estructura de canalización de dos etapas refleja los valores que debería plantearse modificar en la canalización. Probablemente, su canalización tenga más de dos etapas:

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

```
                {
                    "inputArtifacts": [],
                    "name": "Source",
                    "actionTypeId": {
                        "category": "Source",
                        "owner": "AWS",
                        "version": "1",
                        "provider": "S3"
                    },
                    "outputArtifacts": [
                        {
                            "name": "MyApp"
                        }
                    ],
                    "configuration": {
                        "S3Bucket": "amzn-s3-demo-source-bucket",
                        "S3ObjectKey": "ExampleCodePipelineSampleBundle.zip",
                        "PollForSourceChanges": "false"
                    },
                    "runOrder": 1
                }
            ]
        },
        {
            "name": "Staging",
            "actions": [
                {
                    "inputArtifacts": [
                        {
                            "name": "MyApp"
                        }
                    ],
                    "name": "Deploy-CodeDeploy-Application",
                    "actionTypeId": {
                        "category": "Deploy",
                        "owner": "AWS",
                        "version": "1",
                        "provider": "CodeDeploy"
                    },
                    "outputArtifacts": [],
                    "configuration": {
                        "ApplicationName": "CodePipelineDemoApplication",
                        "DeploymentGroupName": "CodePipelineDemoFleet"
                    },
```

```
                        "runOrder": 1
                    }
                ]
            }
        ],
        "artifactStore": {
            "type": "S3",
            "location": "codepipeline-us-east-2-250656481468"
        },
        "name": "MyFirstPipeline",
        "version": 1,
        "variables": [
            {
                "name": "Timeout",
                "defaultValue": "1000",
                "description": "description"
            }
          ]
        },
        "triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "tags": {
                                "includes": [
                                    "v1"
                                ],
                                "excludes": [
                                    "v2"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    "metadata": {
        "pipelineArn": "arn:aws:codepipeline:us-
east-2:80398EXAMPLE:MyFirstPipeline",
        "updated": 1501626591.112,
        "created": 1501626591.112
```

```
    },
    "tags": [{
      "key": "Project",
      "value": "ProjectA"
    }]
}
```

Este ejemplo etiqueta la canalización incluyendo la clave de etiqueta `Project` y el valor `ProjectA` en la canalización. Para obtener más información sobre cómo etiquetar los recursos CodePipeline, consulte[Etiquetado de recursos](#).

Asegúrese de que el parámetro `PollForSourceChanges` del archivo JSON se ha establecido de la siguiente manera:

```
                "PollForSourceChanges": "false",
```

CodePipeline usa Amazon CloudWatch Events para detectar cambios en el repositorio y la rama de CodeCommit origen o en el bucket de código fuente de S3. El siguiente paso incluye instrucciones para crear manualmente estos recursos para la canalización. Al definir la marca como `false`, se deshabilitan las comprobaciones periódicas, que no son necesarias cuando se utilizan los métodos de detección de cambios recomendados.

4. Para crear una acción de compilación, prueba o implementación en una región diferente a la de la canalización, debe añadir lo siguiente a la estructura de canalización. Para obtener instrucciones, consulte [Agregue una acción interregional en CodePipeline](#).

   - Añada el parámetro `Region` a la estructura de la canalización de su acción.
   - Utilice el `artifactStores` parámetro para especificar un depósito de artefactos para cada AWS región en la que tenga una acción.

5. Cuando esté satisfecho con la estructura, guarde el archivo con un nombre como **`pipeline.json`**.

Creación de una canalización

1. Ejecute el comando create-pipeline y use el parámetro `--cli-input-json` para especificar el archivo JSON creado anteriormente.

Para crear una canalización *MySecondPipeline* con el nombre de un archivo JSON denominado pipeline.json que incluya el nombre «*MySecondPipeline*» como valor name en el JSON, el comando tendría el siguiente aspecto:

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

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

Este comando devuelve la estructura de toda la canalización que ha creado.

2. Para ver la canalización, abre la CodePipeline consola y selecciónela de la lista de canalizaciones, o usa el comando. get-pipeline-state Para obtener más información, consulte Vea las canalizaciones y los detalles en CodePipeline.
3. Si utiliza la CLI para crear una canalización, debe crear manualmente los recursos de detección de cambios recomendados para la canalización:

   - Para una canalización con un CodeCommit repositorio, debe crear manualmente la regla de CloudWatch eventos, tal y como se describe enCrea una EventBridge regla para una CodeCommit fuente (CLI).
   - Para una canalización con una fuente de Amazon S3, debe crear manualmente la regla y el seguimiento de CloudWatch eventos, AWS CloudTrail tal y como se describe enConectarse a las acciones de origen de Amazon S3 que utilizan EventBridge y AWS CloudTrail.

## Creación de una canalización a partir de plantillas estáticas

Es posible crear en la consola una canalización que utilice una plantilla para configurar una canalización con el código fuente y las propiedades que especifique. Debe proporcionar la ubicación del archivo de código fuente e información acerca de los proveedores que utilizará para las acciones. Puede especificar una acción de origen para Amazon ECR o cualquier repositorio de terceros compatible con AWS CodeConnections, por ejemplo. GitHub

La plantilla creará una pila CloudFormation para tu canalización que incluye los siguientes recursos:

- Se crea una canalización con el tipo de canalización V2. En Tipo de canalización, elija una de las siguientes opciones: Los tipos de canalización difieren en características y precio. Para obtener más información, consulte Tipos de canalización.
- Se crea un rol de servicio para la canalización y se hace referencia a él en la plantilla.
- Un almacén de artefactos se crea mediante el almacén de artefactos predeterminado, como el bucket de artefactos de S3 designado por defecto, para la canalización en la Región de AWS que haya seleccionado para dicha canalización.

Para ver la colección de plantillas iniciales de código abierto que se utilizan para el asistente de creación de plantillas estáticas, consulta el repositorio en https://github.com/aws/codepipeline-starter-templates.

Cuando utiliza plantillas estáticas para crear una canalización, la estructura de la canalización se configura en cada plantilla según las necesidades del caso de uso. Por ejemplo, la plantilla de un despliegue en CloudFormation se utiliza como ejemplo en este procedimiento. La plantilla genera una canalización denominada DeployToCloudFormationServicecon la siguiente estructura:

- Una etapa de compilación que contiene una acción de origen con la configuración que se especifica en el asistente.
- Una etapa de implementación con una acción de implementación y una pila de recursos asociada en CloudFormation.

Cuando utilizas una plantilla estática para crear una canalización, CodePipeline crea los nombres de las etapas (fuente, compilación, puesta en escena). Estos nombres no se pueden cambiar. Puedes usar nombres más específicos (por ejemplo, BuildToGamma o DeployToProd) para las etapas que añadas más adelante.

Paso 1: acceso a la consola

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En la página Welcome, elija Create pipeline.

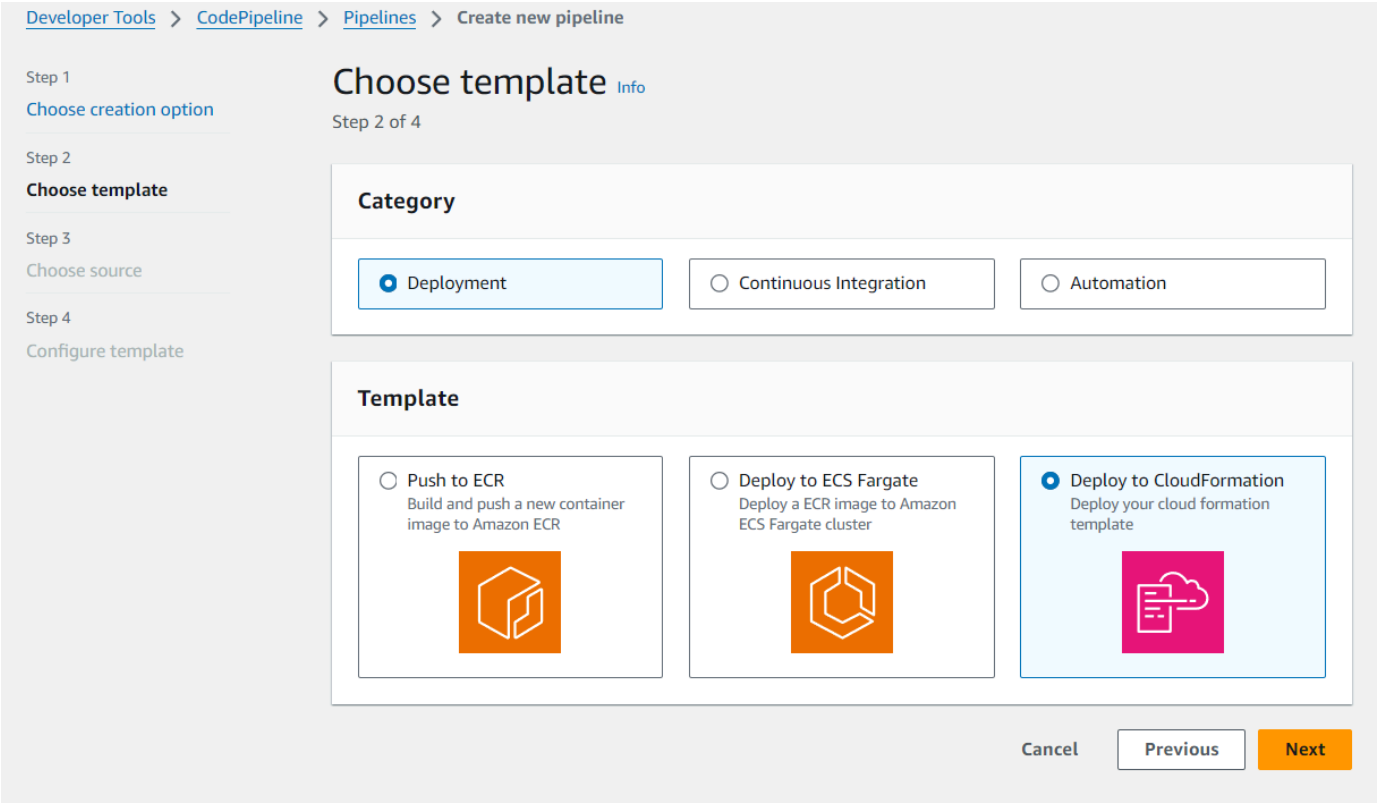
   Si es la primera vez que la usas CodePipeline, selecciona Comenzar.

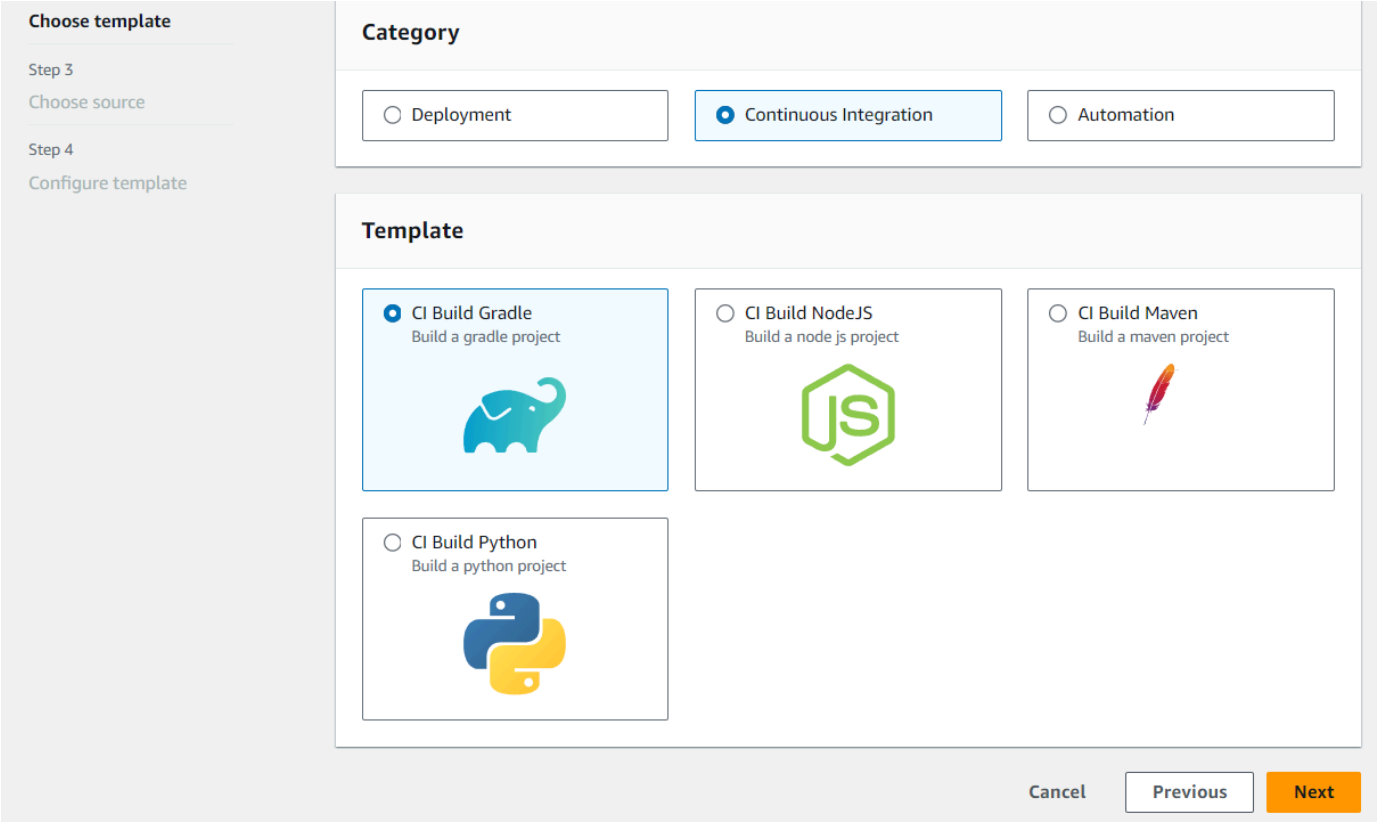
## Paso 2: elección de plantilla

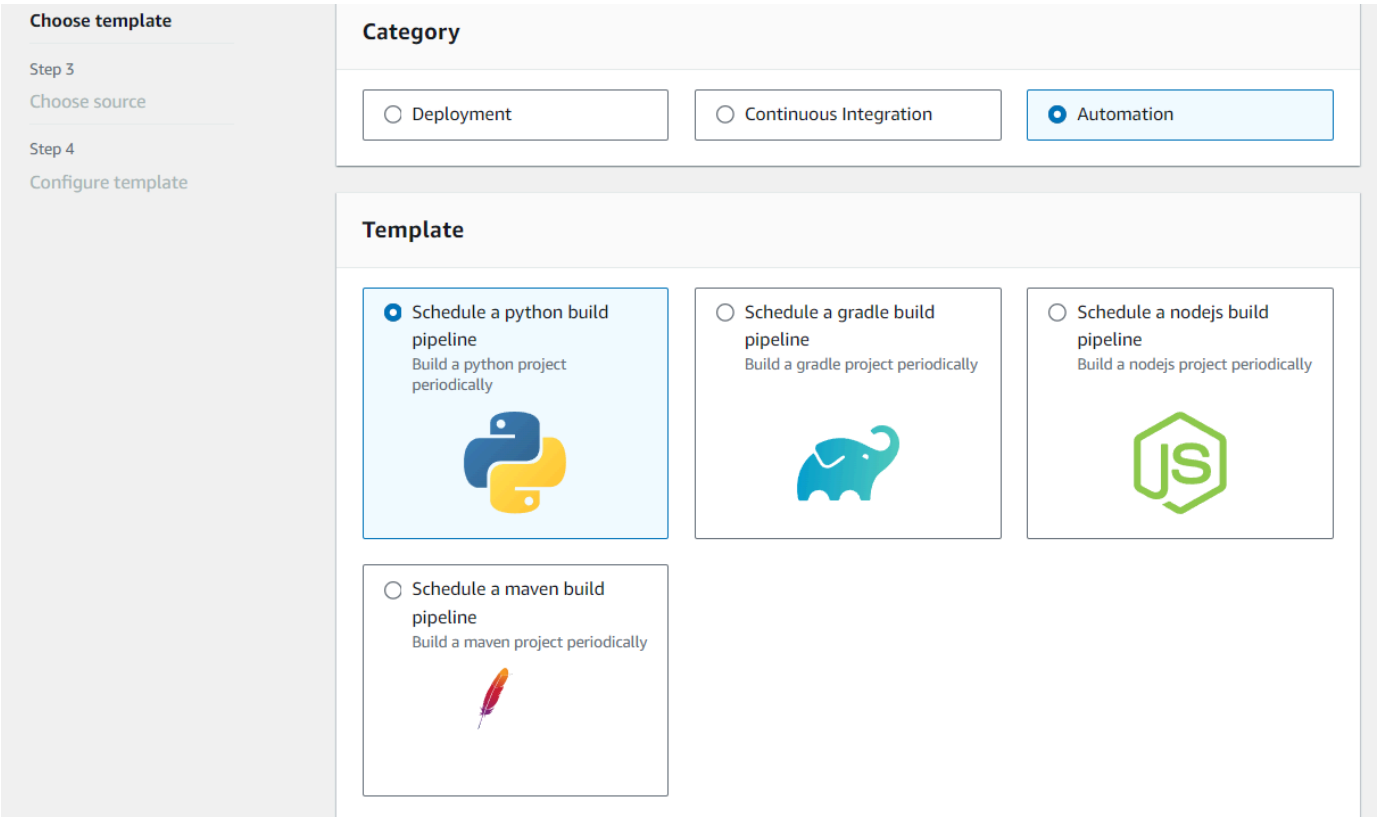
Elija una plantilla para crear una canalización con una etapa de implementación, una automatización o una canalización de CI.

1. En la página Paso 1: elección de la opción de creación realice una de las siguientes acciones y, a continuación, seleccione Siguiente:

   - Elija Implementación si planea crear una etapa de implementación. Vea las opciones de plantillas que se implementan en ECR o CloudFormation. Para este ejemplo, elija Implementación y, a continuación, elija implementar CloudFormation en.
   - Elija Integración continua si planea crear una canalización de CI. Consulte las opciones de canalización de CI, como la compilación en Gradle.
   - Elija Automatización si planea crear una canalización automatizada. Consulte las opciones de automatización, como programar una compilación en Python.

2. 


Choose template
Step 3
Choose source
Step 4
Configure template
Category
Deployment
Continuous Integration
Automation
Template
CI Build Gradle
Build a gradle project
CI Build NodeJS
Build a node js project
CI Build Maven
Build a maven project
CI Build Python
Build a python project
Cancel
Previous
Next


Choose template
Step 3
Choose source
Step 4
Configure template
Category
Deployment
Continuous Integration
Automation
Template
Schedule a python build pipeline
Build a python project periodically
Schedule a gradle build pipeline
Build a gradle project periodically
Schedule a nodejs build pipeline
Build a nodejs project periodically
Schedule a maven build pipeline
Build a maven project periodically

Paso 3: elección del código fuente

- En la página Paso 3: elección del código fuente, en Proveedor de origen, elija el proveedor en el que se almacena el código fuente, especifique las opciones obligatorias y, a continuación, elija Paso siguiente.

  - Para Bitbucket Cloud GitHub (mediante una GitHub aplicación), GitHub Enterprise Server o GitLab.com autogestionado: GitLab

    1. En Conexión, seleccione una conexión existente o cree una nueva. Para crear o gestionar una conexión para tu acción de GitHub origen, consulta. GitHub conexiones
    2. Elija el repositorio que desea usar como ubicación de origen para la canalización.

       Elija agregar un desencadenador o filtrar los tipos de desencadenadores para iniciar la canalización. Para obtener más información acerca de cómo trabajar con desencadenadores, consulte Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código. Para obtener más información acerca de filtros con patrones de glob, consulte Trabajar con patrones glob en la sintaxis.
    3. En Formato del artefacto de salida, debe elegir el formato de los artefactos.
       - Para almacenar los artefactos de salida de la GitHub acción mediante el método predeterminado, elija CodePipelinedefault. La acción accede a los archivos del GitHub repositorio y almacena los artefactos en un archivo ZIP en el almacén de artefactos de Pipeline.
       - Para almacenar un archivo JSON que contiene una referencia URL al repositorio de manera que las acciones posteriores puedan ejecutar comandos Git directamente, elija Clonación completa. Esta opción solo la pueden utilizar las acciones posteriores de CodeBuild .

         Si eliges esta opción, tendrás que actualizar los permisos de tu rol de servicio del CodeBuild proyecto, tal y como se muestra en la siguiente. Solución de problemas CodePipeline Para ver un tutorial que muestra cómo utilizar la opción Clonación completa, consulte Tutorial: Utilice un clon completo con una fuente de GitHub canalización.

  - Para Amazon ECR:
    - En Nombre de repositorio, elija el nombre de su repositorio de Amazon ECR.
    - En Image tag (Etiqueta de imagen), especifique el nombre de la imagen y la versión, si es diferente de LATEST.

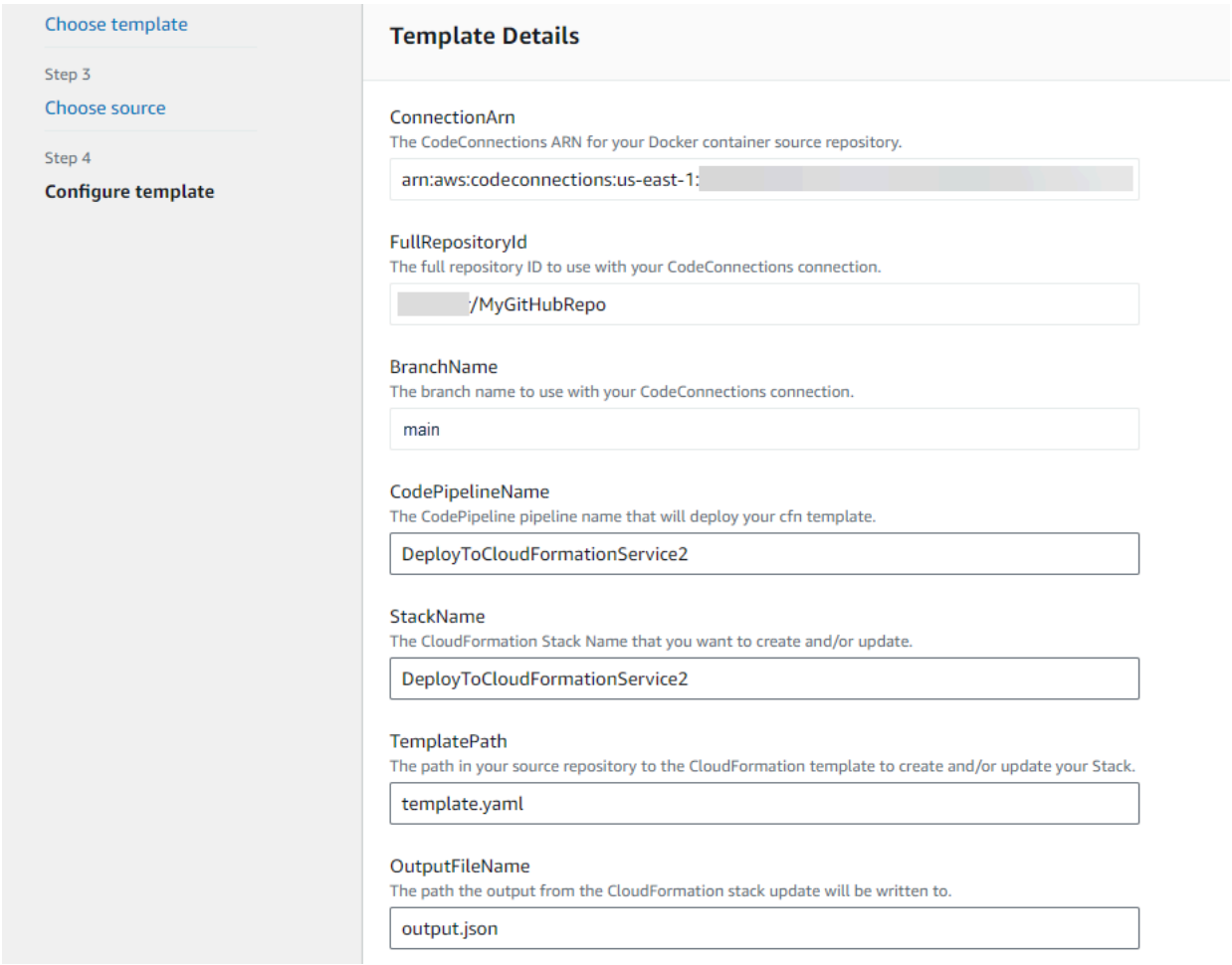
- En Artefactos de salida, elija el artefacto de salida predeterminado, por ejemplo MyApp, el que contenga el nombre de la imagen y la información del URI del repositorio que desee utilizar en la siguiente etapa.

Cuando se incluye una etapa de origen de Amazon ECR en la canalización, la acción de origen genera un archivo `imageDetail.json` como artefacto de salida al confirmar un cambio. Para obtener más información sobre el archivo `imageDetail.json`, consulte Detail.json archivo de imagen para las acciones de blue/green implementación de Amazon ECS.

> **Note**
>
> El objeto y el tipo de archivo deben ser compatibles con el sistema de despliegue que vaya a utilizar (por ejemplo, Elastic Beanstalk o). CodeDeploy Algunos tipos de archivos compatibles son .zip, .tar, y .tgz. Para obtener más información acerca de los tipos de contenedores compatibles con Elastic Beanstalk, consulte Personalización y configuración de entornos y Plataformas compatibles. Para obtener más información sobre cómo implementar las revisiones con CodeDeploy, consulte Cargar la revisión de su aplicación y Preparar una revisión.

Paso 4: configuración de plantilla

Para este ejemplo, se CloudFormation seleccionó la implementación en. En este paso, añada la configuración de la plantilla.




1. En Paso 4: configuración de plantilla, en Nombre de la pila, introduzca un nombre para la canalización.
2. Edite la política de IAM del marcador de posición para los permisos que se aplican a la plantilla.
3. Elija Crear la canalización desde una plantilla.
4. Aparecerá un mensaje en el que se indicará que se están creando los recursos de la canalización.

## Paso 5: visualización de la canalización

- Ahora que has creado tu canalización, puedes verla en la CodePipeline consola y ver la pila en ella CloudFormation. La canalización comienza a ejecutarse una vez creada. Para obtener más información, consulte Vea las canalizaciones y los detalles en CodePipeline. Para obtener más información acerca de cómo realizar cambios en la canalización, consulte Editar una canalización en CodePipeline.

# Editar una canalización en CodePipeline

Las canalizaciones describen el proceso de lanzamiento que desea que siga AWS CodePipeline e incluyen las etapas y las acciones que es necesario completar. Puede editar una canalización para añadir o eliminar dichos elementos. Sin embargo, al editar una canalización, ciertos valores como, por ejemplo, el nombre o los metadatos de la canalización no pueden modificarse.

Puede editar el tipo de canalización, las variables y los desencadenadores mediante la página de edición de la canalización. También puede agregar o cambiar etapas y acciones de la canalización.

A diferencia de lo que sucede al crearla, cuando se edita una canalización no se vuelve a ejecutar en ella la revisión más reciente. Si desea ejecutar la versión más reciente en una canalización que acaba de editar, tendrá que volver a ejecutarla manualmente. De lo contrario, la canalización editada se ejecuta la próxima vez que realice un cambio en una ubicación de origen configurada en la etapa de origen. Para obtener más información, consulte Iniciar la canalización manualmente.

Puede añadir a una canalización acciones que se encuentren en una región de AWS diferente a la de la canalización. Cuando un Servicio de AWS es el proveedor de una acción y este tipo de acción o de proveedor se encuentran en una región de AWS diferente a la de la canalización, se trata de una acción entre regiones. Para obtener más información sobre las acciones entre regiones, consulte Agregue una acción interregional en CodePipeline.

CodePipeline utiliza métodos de detección de cambios para comenzar la canalización cuando se envía un cambio de código fuente. Estos métodos de detección se basan en el tipo de código fuente:

- CodePipeline utiliza Eventos de Amazon CloudWatch para detectar cambios en el repositorio de origen de CodeCommit o en el bucket de origen de Amazon S3.

> **Note**
>
> Los recursos de detección de cambios se crean automáticamente cuando se utiliza la consola. Si utiliza la consola para crear o editar una canalización, los recursos adicionales se crean automáticamente. Si utiliza la AWS CLI para crear la canalización, debe crear los recursos adicionales. Para obtener más información sobre la creación o actualización de una canalización de CodeCommit, consulte Crea una EventBridge regla para una CodeCommit fuente (CLI). Para obtener más información acerca de la creación o actualización de una canalización de Amazon S3 mediante la CLI, consulte Cree una EventBridge regla para una fuente de Amazon S3 (CLI).

Temas


- Editar una canalización (consola)
- Editar una canalización (AWS CLI)


# Editar una canalización (consola)

Puede utilizar la consola de CodePipeline para añadir, editar o eliminar etapas de una canalización y para añadir, editar o eliminar acciones en una etapa.

Cuando actualiza una canalización, CodePipeline completa correctamente todas las acciones en ejecución y, a continuación, falla en las etapas y ejecuciones de la canalización en las que se completaron las acciones en ejecución. Cuando se actualice una canalización, tendrá que volver a ejecutarla. Para obtener más información sobre la ejecución de una canalización, consulte Iniciar la canalización manualmente.

Para editar una canalización

1. Inicie sesión en la Consola de administración de AWS y abra la consola de CodePipeline en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres de todas las canalizaciones asociadas con su cuenta de AWS.
2. En Nombre, elija el nombre de la canalización que desea editar. Esto abre una vista detallada de la canalización, que incluye el estado de cada una de las acciones en cada etapa de la canalización.
3. En la página de detalles de la canalización, elija Edit.
4. Para editar el tipo de canalización, elija Editar en la tarjeta Propiedades de Editar: canalización. Elija una de las siguientes opciones y, a continuación, elija Listo.

   - Las canalizaciones de tipo V1 tienen una estructura JSON que contiene parámetros estándares de canalización, etapa y nivel de acción.
   - Las canalizaciones de tipo V2 tienen la misma estructura que las de tipo V1, además de compatibilidad con parámetros adicionales, como los desencadenadores y las variables a nivel de canalización.

   Los tipos de canalización difieren en características y precio. Para obtener más información, consulte Tipos de canalización.

5. Para editar las variables de la canalización, seleccione Editar variables en la tarjeta Editar: variables. Agregue o cambie variables para el nivel de canalización y, a continuación, seleccione Listo.

   Para obtener más información acerca de las variables a nivel de canalización, consulte Referencia de variables. Para ver un tutorial con una variable a nivel de canalización que se transfiere en el momento de la ejecución de la canalización, consulte Tutorial: Uso de variables a nivel de canalización.

   > **Note**
   >
   > Si bien es opcional añadir variables a nivel de canalización, en el caso de una canalización especificada con variables a nivel de canalización en la que no se proporcionen valores, la ejecución de la canalización fallará.

6. Para editar los desencadenadores de la canalización, elija Editar desencadenadores en la tarjeta Editar: desencadenadores. Agregue o cambie los desencadenadores y, a continuación, seleccione Listo.

   Para obtener más información sobre cómo agregar desencadenadores, consulte los pasos para crear una conexión a Bitbucket Cloud, GitHub (mediante la aplicación de GitHub), GitHub Enterprise Server, GitLab.com o GitLab autoadministrado, como GitHub conexiones.

7. Para editar las etapas y las acciones en la página Editar, realice una de las siguientes operaciones:

   - Para editar una etapa, elija Edit stage (Editar etapa) Puede añadir acciones en serie y en paralelo con las acciones existentes:

     También puede editar las acciones en esta vista eligiendo el icono de edición correspondiente. Para eliminar una acción, elija el icono de eliminación de esa acción.
   - Para editar una acción, elija el icono de edición de dicha acción y modifique los valores que desee en Edit action. Los elementos marcados con un asterisco (*) son obligatorios.
     - En el caso de nombres de repositorio y ramificaciones de , aparece un mensaje que muestra la regla de Eventos de Amazon CloudWatch que se va a crear para esta canalización. Si elimina el origen de CodeCommit , aparece un mensaje que muestra la regla de Eventos de Amazon CloudWatch que se eliminará.

    - En el caso de un bucket de origen de Amazon S3, aparece un mensaje que muestra la regla de Eventos de Amazon CloudWatch y el registro de seguimiento de AWS CloudTrail que se vaya a crear para esta canalización. Si elimina el origen de Amazon S3 , aparece un mensaje que muestra la regla de Eventos de Amazon CloudWatch y el registro de seguimiento de AWS CloudTrail que se eliminará. Si otras canalizaciones están usando el registro de seguimiento de AWS CloudTrail, no se quita el registro de seguimiento y se elimina el evento de datos.
- Para añadir una etapa, elija + Add stage (Añadir etapa) en el lugar de la canalización en el que desea añadirla. Dé un nombre a la etapa y después añada al menos una acción. Los elementos marcados con un asterisco (*) son obligatorios.
- Para eliminar una etapa, elija el icono de eliminación en esa etapa. Se eliminan la etapa y todas sus acciones.
- Para configurar una etapa para que se revierta automáticamente en caso de fallo, seleccione Editar la etapa y, a continuación, seleccione la casilla Configurar la reversión automática en caso de fallo en una etapa.

Por ejemplo, si desea agregar una acción en serie a una etapa de una canalización:

1. En la etapa a la que desee añadir la acción, elija Edit stage (Editar etapa) y, a continuación, añádala haciendo clic en + Add action group (Añadir grupo de acciones).
2. En el cuadro emergente Edit action (Editar acción), en Action name (Nombre de acción), escriba el nombre de la acción. En la lista Action provider (Proveedor de acciones) se muestran opciones organizadas por categoría. Identifique la categoría que quiera, por ejemplo, Deploy (Implementación). Una vez allí, elija el proveedor que quiera, (por ejemplo, AWS CodeDeploy). En Región, elija la región de AWS en la que se crea o tiene previsto crear el recurso. El campo Región designa dónde se crean los recursos de AWS para este tipo de acción y de proveedor. Este campo solo se muestra en el caso de las acciones en las que el proveedor de la acción es un Servicio de AWS. El campo Región se establece de forma predeterminada en la misma región de AWS que la canalización.

   Para ver ejemplos de la adición de proveedores de acciones y del uso de campos predeterminados para cada uno, consulte Creación de una canalización personalizada (consola).

Para añadir CodeBuild como acción de compilación o acción de prueba a una etapa, consulte Usar CodePipeline con CodeBuild para probar el código y ejecutar compilaciones en la Guía del usuario de CodeBuild.

> **Note**
>
> Algunos proveedores de acción, tales como GitHub, requieren que se conecte al sitio web del proveedor antes de completar la configuración de la acción. Al conectarse al sitio web de un proveedor, asegúrese de usar las credenciales correspondientes a dicho sitio web. No use las credenciales de AWS.

3. Cuando haya terminado de configurar la acción, elija Save (Guardar).

> **Note**
>
> No puede cambiar el nombre de una etapa en la vista de la consola. Puede agregar una etapa con el nombre que desea cambiar y, después, eliminar la antigua. Asegúrese de añadir las acciones con las que desea contar en esa etapa antes de eliminar las antiguas.

8. Cuando haya terminado de editar la canalización, elija Save (Guardar) para volver a la página de resumen.

> **Important**
>
> Una vez guardados los cambios, estos no se pueden deshacer. Deberá volver a editar la canalización. Si se ejecuta una revisión en la canalización al guardar los cambios, dicha ejecución no se completa. Si desea una confirmación específica o cambiar la ejecución en la canalización editada, deberá ejecutarla manualmente en la canalización. De lo contrario, la siguiente confirmación o cambio se ejecuta automáticamente en la canalización.

9. Para probar la acción, seleccione Liberar cambio para procesar la confirmación a través de la canalización y confirmar un cambio en la fuente especificada en la etapa de origen de la canalización. O bien, siga los pasos que se indican en Iniciar la canalización manualmente para lanzar manualmente un cambio mediante la AWS CLI.

# Editar una canalización (AWS CLI)

Puede usar el comando update-pipeline para editar una canalización.

Cuando actualiza una canalización, CodePipeline completa correctamente todas las acciones en ejecución y, a continuación, falla en las etapas y ejecuciones de la canalización en las que se completaron las acciones en ejecución. Cuando se actualice una canalización, tendrá que volver a ejecutarla. Para obtener más información sobre la ejecución de una canalización, consulte Iniciar la canalización manualmente.

> **Important**
>
> Es posible utilizar la AWS CLI para editar canalizaciones que incluyan acciones de socios, pero no se debe editar manualmente la codificación JSON de una acción de socio. De lo contrario, la acción del socio produce un error al actualizar la canalización.

Para editar una canalización

1. Abra una sesión de terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows) y ejecute el comando get-pipeline para copiar la estructura de canalización en un archivo JSON. Por ejemplo, para una canalización llamada **`MyFirstPipeline`**, escriba el siguiente comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando no devuelve nada, pero el archivo creado debería aparecer en el directorio en el que se ejecutó el comando.

2. Abra el archivo JSON en cualquier editor de texto sin formato y modifique la estructura del archivo para que refleje los cambios que desea hacer en la canalización. Por ejemplo, puede añadir o eliminar etapas, o añadir otra acción a una etapa existente.

   El siguiente ejemplo muestra cómo añadir otra etapa de implementación en el archivo pipeline.json. Esta etapa se ejecuta después de la primera etapa de implementación denominada *`Staging (Ensayo)`*.

> **Note**
>
> Esto es solo un fragmento del archivo, no toda la estructura. Para obtener más información, consulte CodePipeline referencia de estructura de tuberías.

```
,
        {
            "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
                    }
                ]
            }
        ]
}
```

Para obtener información sobre el modo utilizar la CLI para añadir una acción de aprobación a una canalización, consulte Añada una acción de aprobación manual a una canalización en CodePipeline.

Asegúrese de que el parámetro `PollForSourceChanges` del archivo JSON se ha establecido de la siguiente manera:

```
                "PollForSourceChanges": "false",
```

CodePipeline utiliza Eventos de Amazon CloudWatch para detectar cambios en el repositorio y la ramificación de origen en CodeCommit o en el bucket de origen de Amazon S3. El siguiente paso incluye instrucciones para crear estos recursos manualmente. Al definir la marca como `false`, se deshabilitan las comprobaciones periódicas, que no son obligatorias cuando utiliza los métodos de detección de cambios recomendados.

3. Para añadir una acción de prueba, compilación o implementación en una región diferente a la de la canalización, debe añadir lo siguiente a su estructura de la canalización. Para obtener instrucciones detalladas, consulte Agregue una acción interregional en CodePipeline.

   - Añada el parámetro `Region` a la estructura de la canalización de su acción.
   - Utilice el parámetro `artifactStores` para especificar un bucket de artefactos para cada región en la que tenga una acción.

4. Si está trabajando con la estructura de la canalización recuperada mediante el comando get-pipeline, debe modificar la estructura del archivo JSON. Debe eliminar las líneas `metadata` del archivo para que el comando update-pipeline pueda utilizarlo. Quite la sección de la estructura de canalizaciones del archivo JSON (las líneas `"metadata": { }` y los campos `"created"`, `"pipelineARN"` y `"updated"` que contenga).

   Por ejemplo, quite las siguientes líneas de la estructura:

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

   Guarde el archivo.

5. Si utiliza la CLI para editar una canalización, debe administrar manualmente los recursos de detección de cambios recomendados para dicha canalización:

   - Para los repositorios de CodeCommit, debe crear la regla de Eventos de CloudWatch, tal y como se describe en Crea una EventBridge regla para una CodeCommit fuente (CLI).
   - Para un origen de Amazon S3, debe crear la regla de Eventos de CloudWatch y el registro de seguimiento de AWS CloudTrail, tal y como se describe en Conectarse a las acciones de origen de Amazon S3 que utilizan EventBridge y AWS CloudTrail.

6. Para aplicar los cambios, ejecute el comando update-pipeline especificando el archivo JSON:

   > **Important**
   >
   > Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

   Este comando devuelve la estructura completa de la canalización editada.

> **Note**
>
> El comando update-pipeline detiene la canalización. Si se está ejecutando una revisión en la canalización al ejecutar el comando update-pipeline, dicha ejecución se detiene. Debe comenzar la canalización actualizada manualmente para ejecutar dicha revisión en ella.

7. Abra la consola de CodePipeline y elija la canalización que acaba de editar.

   La canalización muestra los cambios. La próxima vez que haga un cambio en la ubicación de origen, la canalización ejecuta dicha revisión a través de la estructura revisada de la canalización.

8. Para ejecutar manualmente la última revisión a través de la estructura revisada de la canalización, ejecute el comando start-pipeline-execution. Para obtener más información, consulte Iniciar la canalización manualmente.

Para obtener más información sobre la estructura de una canalización y los valores previstos, consulte CodePipeline referencia de estructura de tuberías y AWS CodePipeline API Reference.

# Vea las canalizaciones y los detalles en CodePipeline

Puedes usar la AWS CodePipeline consola o la AWS CLI para ver los detalles de las canalizaciones asociadas a tu AWS cuenta.

Temas

- Ver canalizaciones (consola)
- Ver los detalles de las acciones en una canalización (consola)
- Ver el ARN de la canalización y el ARN del rol de servicio (consola)
- Visualización de los detalles y el historial de la canalización (CLI)
- Visualización de resultados de reglas para condiciones de etapa en el historial de ejecución

## Ver canalizaciones (consola)

Puede ver el estado, las transiciones y las actualizaciones de los artefactos de una canalización.

> **Note**
>
> Al cabo de una hora, la vista detallada de una canalización deja de actualizarse automáticamente en el navegador. Para ver la información actual, actualice la página.

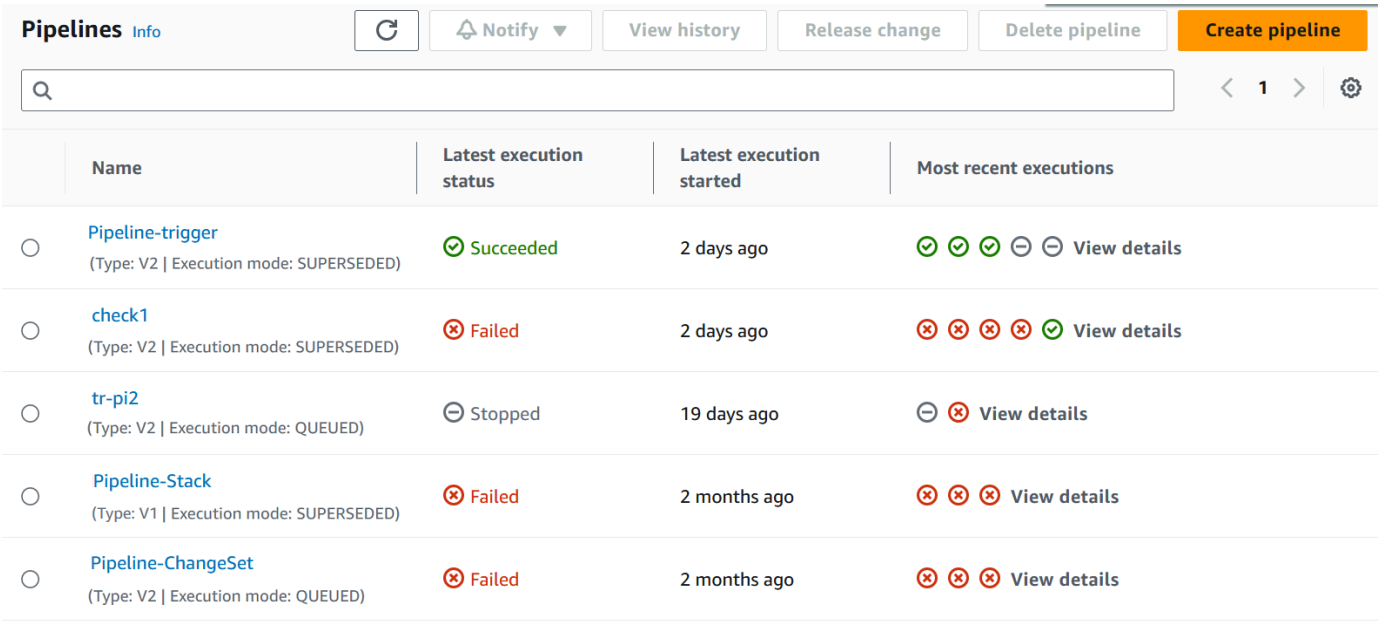
Para ver canalizaciones

1. Inicia sesión en la CodePipeline consola Consola de administración de AWS y ábrela en http:// console.aws.amazon. com/codesuite/codepipeline/home.

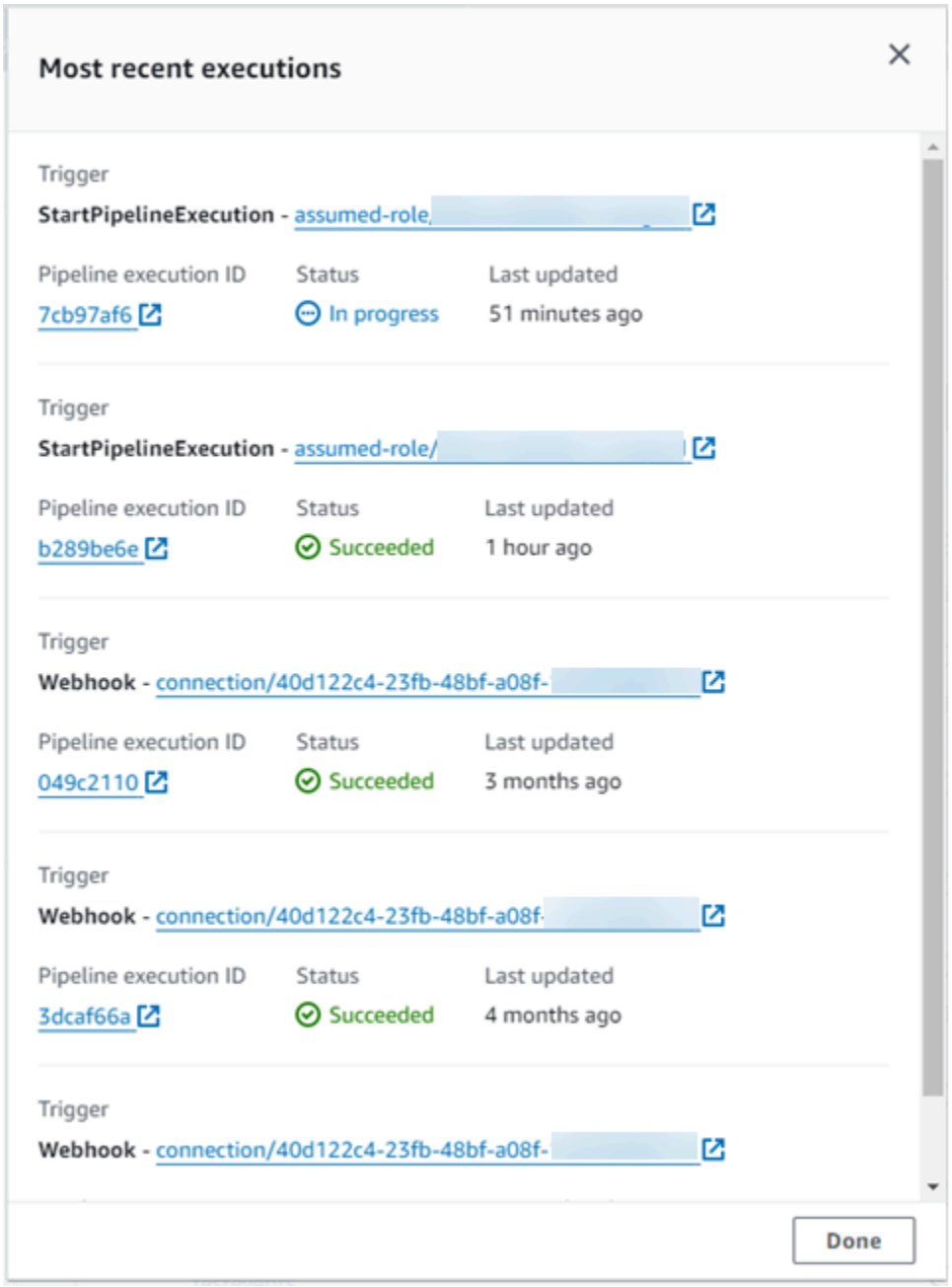
   Aparece la página Canalizaciones. Se muestra una lista de todas las canalizaciones de esa región.

   Se muestran el nombre, el tipo, el estado, la versión, la fecha de creación y la fecha de la última modificación de todas las canalizaciones asociadas a tu AWS cuenta, junto con la hora de ejecución iniciada más recientemente.

2. Se muestra el estado de las cinco ejecuciones más recientes.




   Elija Ver detalles junto a una fila específica para mostrar un cuadro de diálogo de detalles con las ejecuciones más recientes.

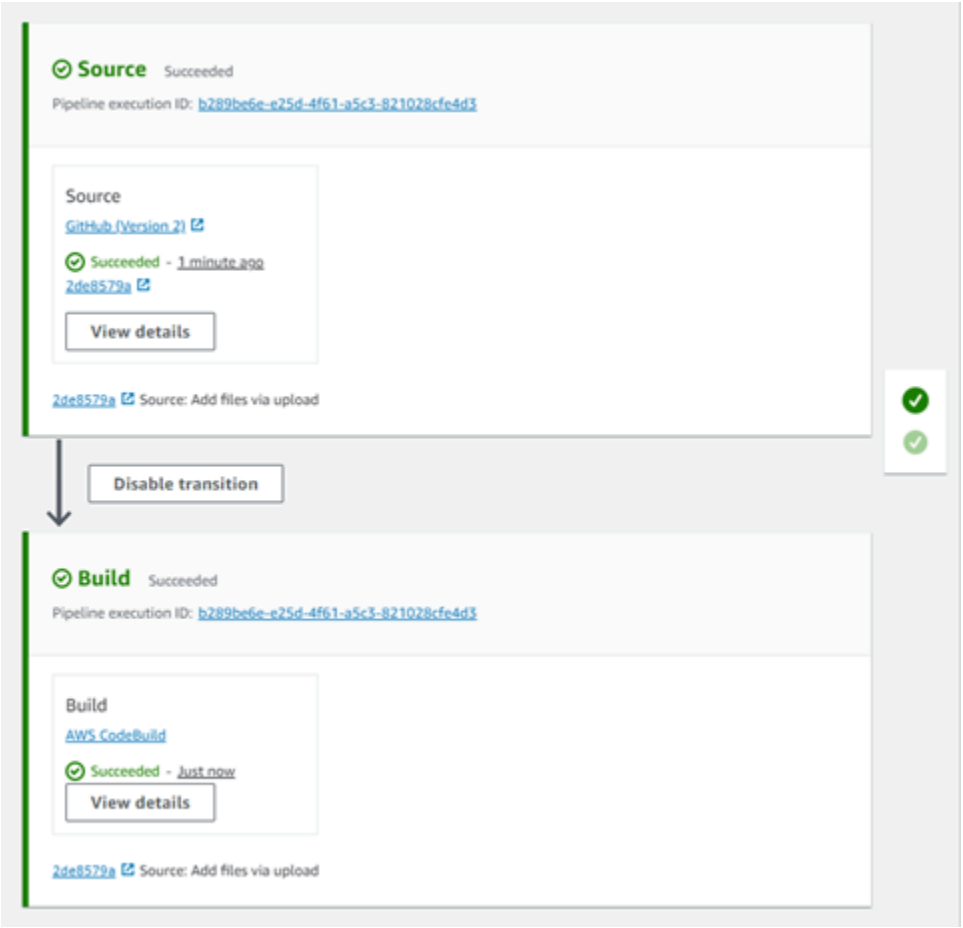



Para ver detalles acerca de las ejecuciones más recientes de la canalización, elija View history (Ver historial). En el caso de las ejecuciones anteriores, puedes ver los detalles de las revisiones asociados a los artefactos de origen, como la ejecución IDs, el estado, las horas de inicio y finalización, la duración, la confirmación IDs y los mensajes.

> **Note**
>
> En el caso de una canalización en modo de ejecución PARALELO, la vista de canalización principal no muestra la estructura de la canalización ni las ejecuciones en

curso. En el caso de una canalización en modo de ejecución PARALELO, para acceder a la estructura de la canalización hay que seleccionar el ID de la ejecución que se quiere ver en la página del historial de ejecuciones. Elija Historial en el menú de navegación de la izquierda, elija el ID de ejecución para la ejecución en paralelo y, a continuación, consulte la canalización en la pestaña Visualización.

3. Para ver los detalles de una canalización, elíjala en Name. Se muestra una vista detallada de la canalización, incluido el estado de cada acción en cada etapa y el estado de las transiciones.




La vista gráfica muestra la siguiente información para cada etapa:

- El nombre de la etapa
- Las acciones configuradas para la etapa.
- El estado de las transiciones entre etapas (habilitadas o deshabilitadas), que se indica mediante el aspecto de la flecha entre etapas. Una transición habilitada se indica con una flecha con un botón Disable transition (Deshabilitar transición) junto a ella. Las transiciones

deshabilitadas tienen una flecha con un tachado y un botón Enable transition (Habilitar transición) al lado.

- Una barra de color que indica el estado de la etapa:
  - Gris: sin ejecuciones
  - Azul: en progreso
  - Verde: se realizó correctamente
  - Rojo: error

La vista gráfica también muestra la siguiente información acerca de las acciones en cada etapa:

- El nombre de la acción.
- El proveedor de la acción, por ejemplo CodeDeploy.
- Cuando se ejecutó por última vez la acción.
- Si la acción se ha realizado correctamente o si se ha producido un error.
- Enlaces a otros detalles acerca de la última ejecución de la acción, si están disponibles.
- Detalles sobre las revisiones de origen que se están ejecutando durante la última ejecución de la fase o, en el caso de las CodeDeploy implementaciones, las últimas revisiones de fuente que se implementaron en las instancias de destino.
- Un botón Ver detalles que abre un cuadro de diálogo con detalles sobre la ejecución de la acción, los registros y la configuración de la acción.

> **Note**
>
> La pestaña Registros está disponible para CloudFormation las acciones que se hayan ejecutado en la cuenta de la canalización CodeBuild y las acciones que se hayan ejecutado en ella.

4. Para ver los detalles del proveedor de la acción, elija al proveedor. Por ejemplo, en el ejemplo anterior de la canalización, si selecciona CodeDeploy la fase de preparación o la fase de producción, se mostrará la página de CodeDeploy consola del grupo de despliegue configurado para esa fase.
5. Para ver el progreso de una acción mostrado junto a una acción en curso (se indica con el mensaje En curso). Si la acción se está realizando, verá el progreso y los pasos o las acciones a medida que se producen.

6. Para aprobar o rechazar las acciones que se han configurado para la aprobación manual, elija Review.
7. Para volver a intentar acciones en una etapa que no se completó correctamente, elija Retry.
8. El estado desde la última vez que se ejecutó la acción, incluidos los resultados de dicha acción, Realizados correctamente o No realizados.

## Ver los detalles de las acciones en una canalización (consola)

Puede ver los detalles de una canalización, incluidos los detalles de las acciones en cada etapa.

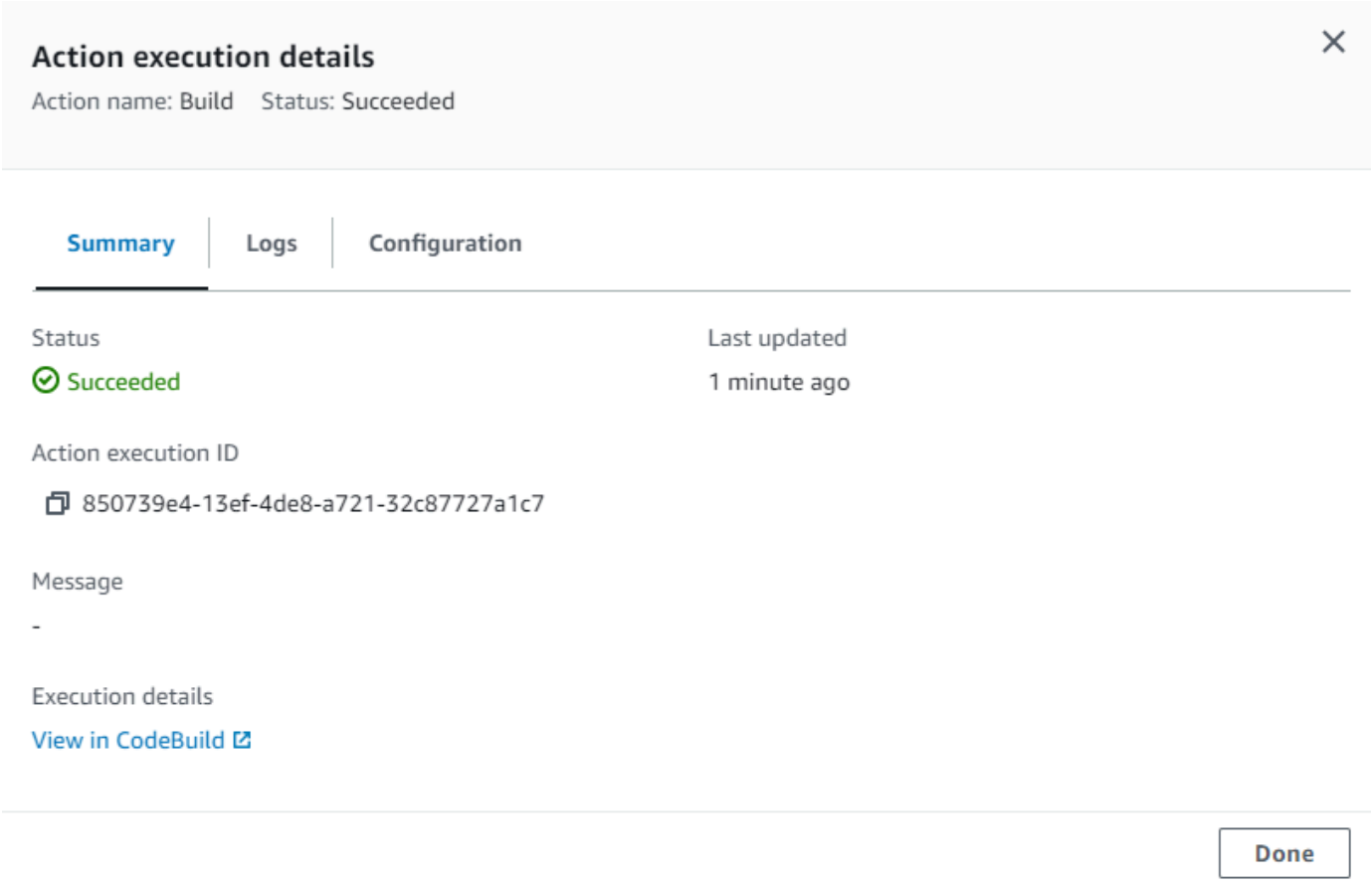
> **Note**
>
> Al cabo de una hora, la vista detallada de una canalización deja de actualizarse automáticamente en el navegador. Para ver la información actual, actualice la página.

Ver los detalles de las acciones en una canalización

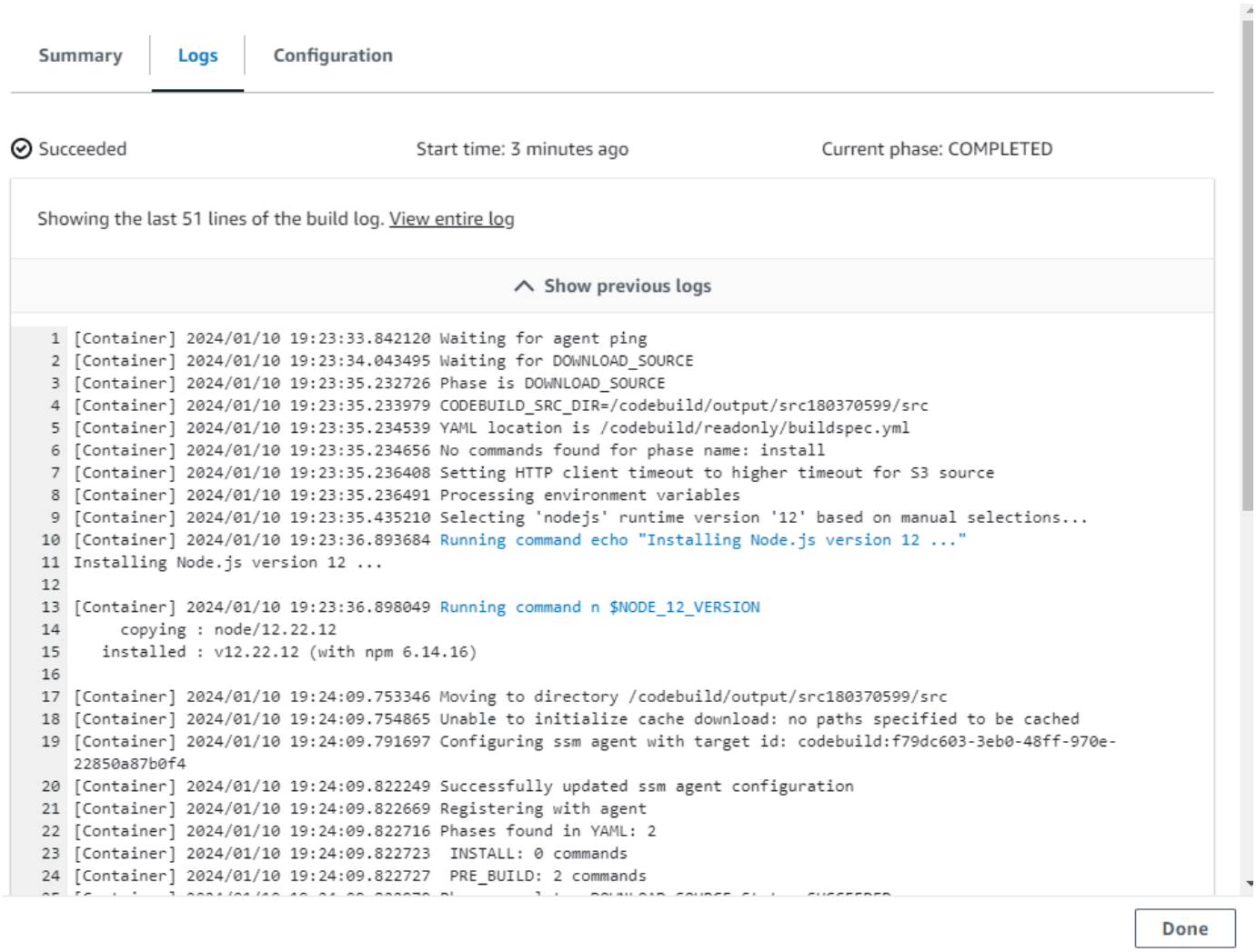
1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http:// console.aws.amazon. com/codesuite/codepipeline/home.

   Aparece la página Canalizaciones.
2. En cualquier acción, elija Ver detalles para abrir un cuadro de diálogo con detalles sobre la ejecución de la acción, los registros y la configuración de la acción.

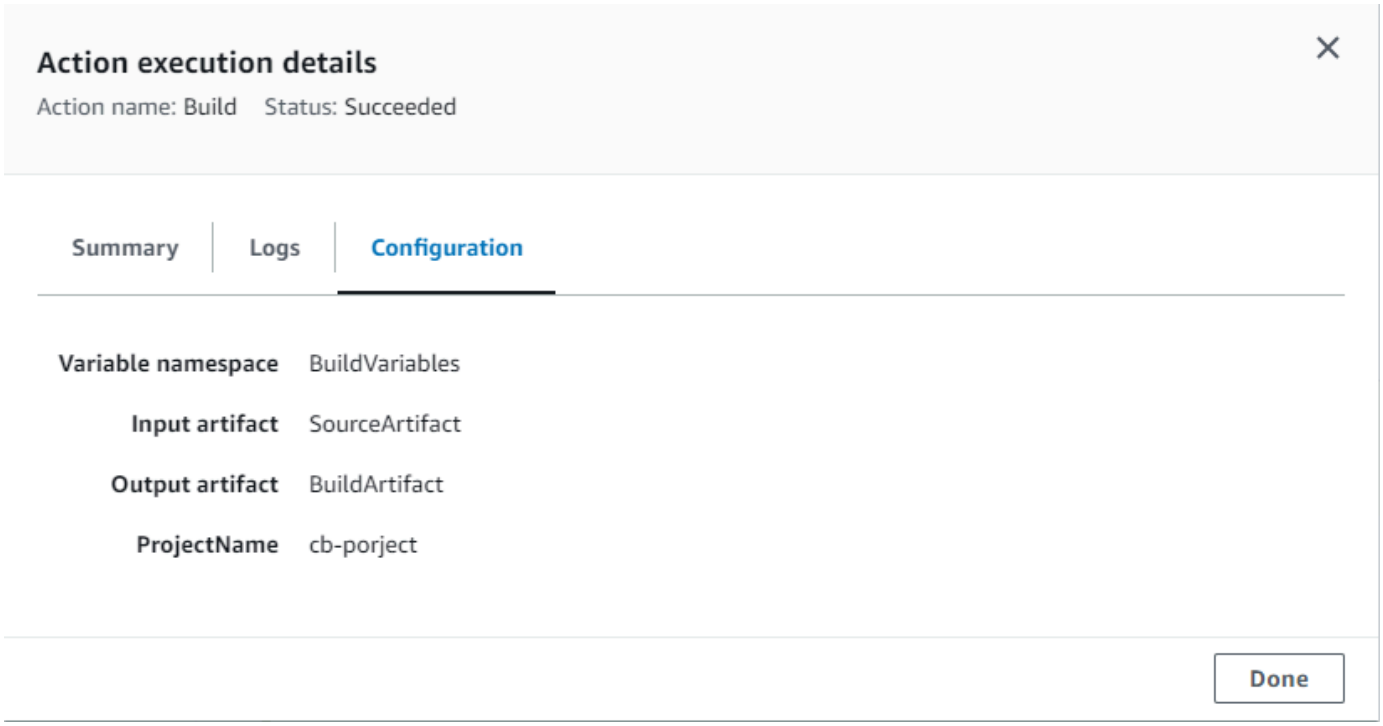
   > **Note**
   >
   > La pestaña Registros está disponible para CloudFormation las acciones CodeBuild y las acciones.
3. Para ver el resumen de una acción en una etapa de una canalización, elija Ver detalles de la acción y, a continuación, elija la pestaña Resumen.




4. Para ver los registros de acciones de una acción con registros, elija Ver detalles de la acción y, a continuación, elija la pestaña Registros.




5. Para ver los detalles de configuración de una acción, elija la pestaña Configuración.




## Ver el ARN de la canalización y el ARN del rol de servicio (consola)

Puede usar la consola para ver la configuración de la canalización, como el ARN de la canalización, el ARN del rol de servicio y el almacén de artefactos de la canalización.

1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http://console.aws.amazon. com/codesuite/codepipeline/home.

   Aparecerán los nombres de todas las canalizaciones asociadas a tu AWS cuenta.
2. En el panel de navegación izquierdo, elija Configuración. La página muestra lo siguiente:

   - El nombre de la canalización.
   - El nombre de recurso de Amazon (ARN) .

     El ARN de canalización se crea con el siguiente formato:

     arn:aws:codepipeline::: *region account pipeline-name*

     ARN de canalización de muestra:

     `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`
   - El ARN del rol de CodePipeline servicio de tu canalización

- La versión de la canalización.
- El nombre y la ubicación del almacén de artefactos de la canalización

## Visualización de los detalles y el historial de la canalización (CLI)

Puede ejecutar los siguientes comandos para ver los detalles acerca de sus canalizaciones y las ejecuciones de canalizaciones:

- list-pipelinescomando para ver un resumen de todas las canalizaciones asociadas a tu AWS cuenta.
- get-pipeline para repasar los detalles de una canalización individual.
- list-pipeline-executions para ver resúmenes de las ejecuciones más recientes de una canalización.
- get-pipeline-execution para ver información sobre la ejecución de una canalización, como los detalles de los artefactos, el ID de ejecución de la canalización y el nombre, la versión y el estado de la canalización.
- get-pipeline-state para ver el estado de la canalización, de las etapas y de las acciones.
- list-action-executions para ver los detalles de ejecución de las acciones de una canalización.

1. Abre una terminal (Linux, macOS o Unix) o una línea de comandos (Windows) y usa el comando AWS CLI para ejecutar el list-pipelines comando:

   ```
   aws codepipeline list-pipelines
   ```

   Este comando devuelve una lista de todas las canalizaciones asociadas a su cuenta de AWS .

2. Para ver los detalles acerca de una canalización, ejecute el comando get-pipeline especificando el nombre único de la canalización. Por ejemplo, para ver detalles acerca de una canalización denominada *MyFirstPipeline*, escriba lo siguiente:

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

   Este comando devuelve la estructura de la canalización.

# Visualización de resultados de reglas para condiciones de etapa en el historial de ejecución

Puede ver los resultados de las reglas para una ejecución en la que la condición de una etapa ejecutara una regla y activara un resultado para dicha etapa, como una reversión o un error.

Los valores de estado válidos para las condiciones y las reglas son los siguientes: `InProgress | Failed | Errored | Succeeded | Cancelled | Abandoned | Overridden`

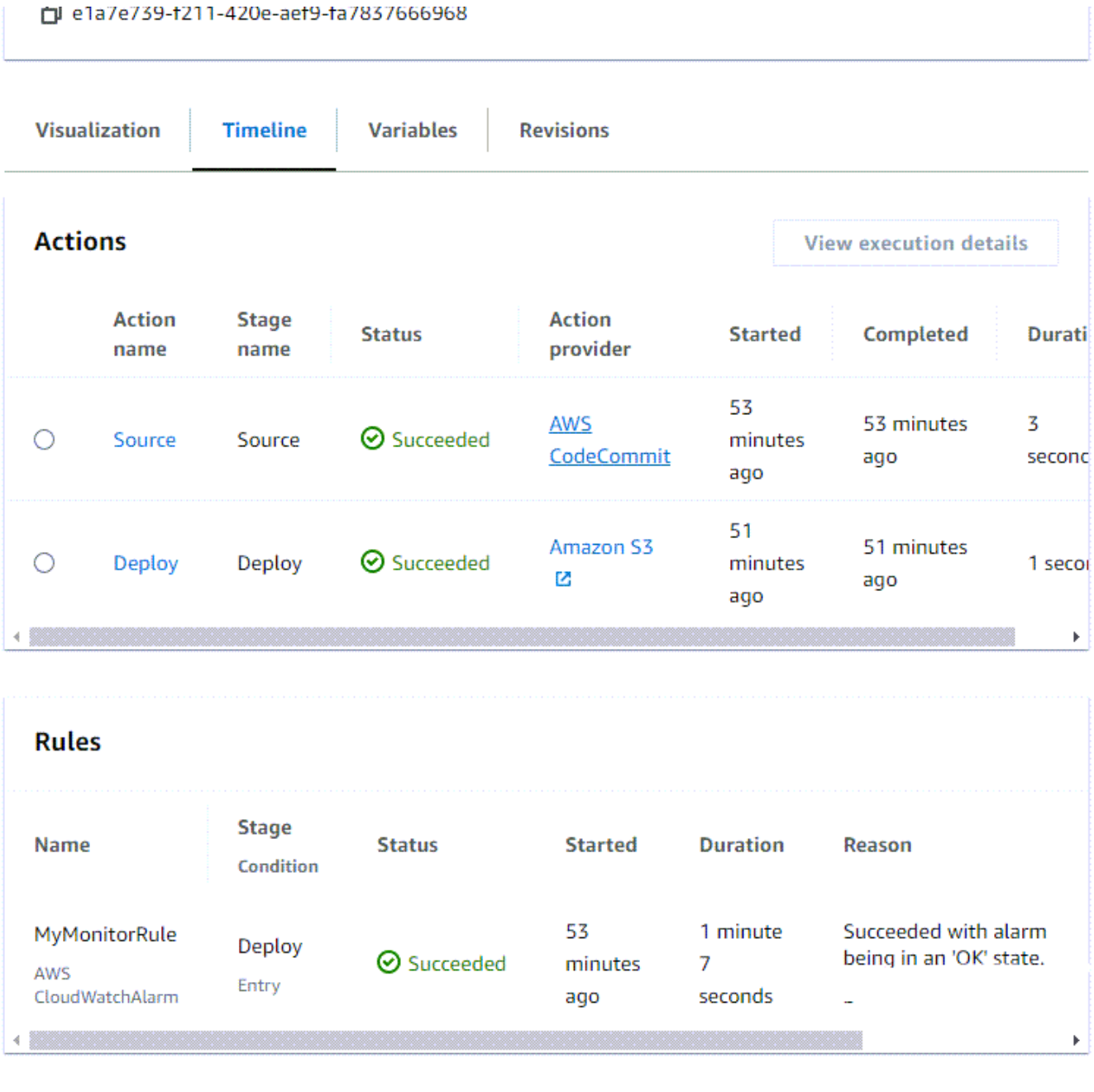
## Visualización de resultados de reglas para condiciones de etapa en el historial de ejecución (consola)

Puede usar la consola para ver los resultados de las reglas para una ejecución en la que la condición de una etapa ejecutara una regla y activara un resultado para dicha etapa.

Visualización de resultados de reglas para condiciones de etapa (consola)

1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http:// console.aws.amazon. com/codesuite/codepipeline/home.

   Se muestran los nombres y estados de todas las canalizaciones asociadas con su Cuenta de AWS .
2. En Nombre, elija el nombre de la canalización que desee visualizar.
3. Seleccione Historial y, a continuación, elija la ejecución. En la página del historial, seleccione la pestaña Línea temporal. En Reglas, consulte los resultados de la regla para la ejecución.




## Visualización de resultados de reglas para condiciones de etapa con **`list-rule-executions`** (CLI)

Puede usar la CLI para ver los resultados de las reglas para una ejecución en la que la condición de una etapa ejecutara una regla y activara un resultado para dicha etapa.

- Abre una terminal (Linux, macOS o Unix) o una línea de comandos (Windows) y usa el comando AWS CLI para ejecutar el list-rule-executions comando para una canalización llamada*`MyPipeline`*:

```
aws codepipeline list-rule-executions --pipeline-name MyFirstPipeline
```

Este comando devuelve una lista de todas las ejecuciones de reglas completadas asociadas con la canalización.

En el siguiente ejemplo, se muestran los datos devueltos para una canalización con una condición de fase en la que se asigna un nombre *MyMonitorRule* a la regla.

```
{
    "ruleExecutionDetails": [
        {
            "pipelineExecutionId": "e1a7e739-f211-420e-aef9-fa7837666968",
            "ruleExecutionId": "3aafc0c7-0e1c-44f1-b357-d1b16a28e483",
            "pipelineVersion": 9,
            "stageName": "Deploy",
            "ruleName": "MyMonitorRule",
            "startTime": "2024-07-29T15:55:01.271000+00:00",
            "lastUpdateTime": "2024-07-29T15:56:08.682000+00:00",
            "status": "Succeeded",
            "input": {
                "ruleTypeId": {
                    "category": "Rule",
                    "owner": "AWS",
                    "provider": "CloudWatchAlarm",
                    "version": "1"
                },
                "configuration": {
                    "AlarmName": "CWAlarm",
                    "WaitTime": "1"
                },
                "resolvedConfiguration": {
                    "AlarmName": "CWAlarm",
                    "WaitTime": "1"
                },
                "region": "us-east-1",
                "inputArtifacts": []
            },
            "output": {
                "executionResult": {
                    "externalExecutionSummary": "Succeeded with alarm 'CWAlarm'
 being i
```

```
n an 'OK' state."
                }
            }
        }
```

# Eliminar una canalización en CodePipeline

Siempre puede editar una canalización para cambiar su funcionalidad, pero también puede estimar conveniente eliminarla. Puede usar la consola de AWS CodePipeline o ejecutar el comando delete-pipeline en la AWS CLI para eliminar canalizaciones.

Temas

- Eliminar una canalización (consola)
- Eliminar una canalización (CLI)

## Eliminar una canalización (consola)

Cómo eliminar una canalización

1. Inicie sesión en la Consola de administración de AWS y abra la consola de CodePipeline en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres y estados de todas las canalizaciones asociadas con su cuenta de AWS.
2. En Name, elija el nombre de la canalización que desea eliminar.
3. En la página de detalles de la canalización, elija Edit.
4. En la página Edit, elija Delete.
5. Escriba **`delete`** en el campo para confirmar y, a continuación, elija Delete (Eliminar).

> **Important**
>
> Esta acción no se puede deshacer.

# Eliminar una canalización (CLI)

Para eliminar manualmente una canalización mediante la AWS CLI, utilice el comando delete-pipeline.

> **Important**
>
> La eliminación de una canalización es irreversible. No hay un cuadro de diálogo de confirmación. Una vez ejecutado el comando, la canalización se habrá eliminado, pero todos los recursos utilizados en ella seguirán existiendo. De este modo es más sencillo crear una nueva canalización que use esos recursos para automatizar la publicación de su software.

Cómo eliminar una canalización

1. Abra un terminal o el símbolo del sistema (Windows) y utilice la AWS CLI para ejecutar el comando delete-pipeline, especificando el nombre de la canalización que desea iniciar. Por ejemplo, para eliminar una canalización denominada *MyFirstPipeline*:

   ```
   aws codepipeline delete-pipeline --name MyFirstPipeline
   ```

   Este comando no devuelve nada.

2. Elimine los recursos que ya no necesite.

   > **Note**
   >
   > Al eliminar una canalización no se eliminan los recursos usados en ella, como la aplicación de CodeDeploy o Elastic beanstalk que usó para implementar el código o si creó una canalización a partir de la consola de CodePipeline, la CodePipeline del bucket de Amazon S3 creado para almacenar los artefactos en sus canalizaciones. Asegúrese de eliminar los recursos que ya no necesite para que no se le cobren en el futuro. Por ejemplo, si usa la consola para crear una canalización por primera vez, CodePipeline crea un bucket de Amazon S3 que almacena los artefactos de todas sus canalizaciones. Si ha eliminado todas sus canalizaciones, siga los pasos en Eliminar un bucket.

# Cree una canalización en la CodePipeline que se utilicen los recursos de otra AWS inscrita

Puede ser conveniente crear una canalización que use recursos creados o administrados por otra cuenta de AWS . Por ejemplo, podría ser conveniente usar una cuenta para una canalización y otra para los recursos de CodeDeploy .

> **Note**
>
> Al crear una canalización con acciones de varias cuentas, debe configurar sus acciones para que puedan obtener acceso a los artefactos en las limitaciones de las canalizaciones entre cuentas. Las siguientes limitaciones se aplican a las acciones entre cuentas:
>
> - En general, una acción solo puede consumir un artefacto si:
>   - La acción está en la misma cuenta que la cuenta de canalización, o
>   - El artefacto se creó en la cuenta de la canalización para una acción en otra cuenta, o
>   - El artefacto se produjo por una acción anterior en la misma cuenta que la acción
>
>   En otras palabras, no puede pasar un artefacto de una cuenta a otra si ninguna de las dos cuentas es la de canalización.
> - Cross-account las acciones no son compatibles con los siguientes tipos de acción:
>   - Acciones de compilación Jenkins

En este ejemplo, debes crear una clave AWS Key Management Service (AWS KMS) para usarla, agregar la clave a la canalización y configurar las políticas y funciones de la cuenta para permitir el acceso entre cuentas. Para una clave de AWS KMS, puedes usar el ID de la clave, el ARN de la clave o el ARN del alias.

> **Note**
>
> Los alias se reconocen únicamente en la cuenta que ha creado la clave de KMS. Para las acciones entre cuentas, solo puedes usar el ID de la clave o el ARN de la clave para identificar la clave. Cross-account las acciones implican usar el rol de la otra cuenta (AccountB), por lo que al especificar el ID de la clave se utilizará la clave de la otra cuenta (AccountB).

En este tutorial y sus ejemplos, *AccountA* se muestra la cuenta que se usó originalmente para crear la canalización. Tiene acceso al bucket de Amazon S3 que se utiliza para almacenar los artefactos de la canalización y al rol de servicio que utiliza. AWS CodePipeline*AccountB*es la cuenta utilizada originalmente para crear la CodeDeploy aplicación, el grupo de implementación y el rol de servicio utilizados por CodeDeploy.

*AccountA*Para editar una canalización para usar la CodeDeploy aplicación creada por*AccountB*, *AccountA* debes:

- Solicita el ARN o el ID de la cuenta de *AccountB* (en este tutorial, el *AccountB* ID es*012ID_ACCOUNT_B*).
- Cree o utilice una clave gestionada por el AWS KMS cliente en la región para la canalización y conceda permisos para usar esa clave para el rol de servicio (*CodePipeline_Service_Role*) y. *AccountB*
- Cree una política de bucket de Amazon S3 que conceda *AccountB* acceso al bucket de Amazon S3 (por ejemplo,*codepipeline-us-east-2-1234567890*).
- Cree una política que permita *AccountA* asumir un rol configurado por *AccountB* y adjunte esa política al rol de servicio (*CodePipeline_Service_Role*).
- Edite la canalización para usar la AWS KMS clave gestionada por el cliente en lugar de la clave predeterminada.

*AccountB*Para permitir el acceso a sus recursos a una canalización creada en*AccountA*, *AccountB* debe:

- Solicita el ARN o el ID de la cuenta de *AccountA* (en este tutorial, el *AccountA* ID es*012ID_ACCOUNT_A*).
- Cree una política aplicada al rol de instancia de Amazon EC2 configurado para el CodeDeploy que permita el acceso al bucket de Amazon S3 (). *codepipeline-us-east-2-1234567890*
- Cree una política aplicada al rol de instancia de Amazon EC2 configurado para el CodeDeploy que permita el acceso a la clave gestionada por el AWS KMS cliente que se utiliza para cifrar los artefactos de la canalización. *AccountA*
- Configure y asocie un rol de IAM (*CrossAccount_Role*) con una política de relaciones de confianza que permita al rol de CodePipeline servicio *AccountA* asumir el rol.
- Cree una política que permita el acceso a los recursos de implementación que requiere la canalización y adjúntela a*CrossAccount_Role*.

- Cree una política que permita el acceso al bucket de Amazon S3 (*codepipeline-us-east-2-1234567890*) y adjúntela a*CrossAccount_Role*.

Temas


- Requisito previo: crear un AWS KMS clave de cifrado
- Paso 1: Configurar roles y políticas de cuenta
- Paso 2: Editar la canalización


## Requisito previo: crear un AWS KMS clave de cifrado

Customer-managed las claves son específicas de una región, al igual que todas AWS KMS las claves. Debes crear tu AWS KMS clave gestionada por el cliente en la misma región en la que se creó la canalización (por ejemplo,`us-east-2`).

Para crear una clave gestionada por el cliente en AWS KMS

1. Inicia sesión en Consola de administración de AWS con *AccountA* y abre la AWS KMS consola.
2. A la izquierda, seleccione Claves administradas por el cliente.
3. Elija Crear clave. En la Configurar clave, deje el valor predeterminado Simétrico seleccionado y elija Siguiente.
4. En Alias, introduce el alias que quieras usar para esta clave (por ejemplo,*PipelineName-Key*). Opcionalmente, proporcione una descripción y las etiquetas de esta clave; después, elija Paso siguiente.
5. En Definir permisos de administración de claves, elija su usuario de y cualesquiera otros usuarios o grupos que desee que actúen como administradores de esta clave y, a continuación, elija Siguiente.
6. En Definir los permisos de uso de claves, en Esta cuenta, seleccione el nombre del rol de servicio para la canalización (por ejemplo, CodePipeline _Service_Role). En Otras AWS cuentas, elige Agregar otra AWS cuenta. Introduce el identificador de la cuenta *AccountB* para completar el ARN y, a continuación, selecciona Siguiente.
7. En Revisar y editar política de claves, revise la política y, después, elija Terminar.
8. En la lista de claves, elija el alias de la clave y copie su ARN (por ejemplo, ***arn:aws:kms:us-east-2:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE***). Lo necesitará cuando edite la canalización y configure las políticas.

# Paso 1: Configurar roles y políticas de cuenta

Después de crear la AWS KMS clave, debe crear y adjuntar políticas que permitan el acceso multicuenta. Esto requiere acciones tanto por parte de como*AccountA*. *AccountB*

Temas

- Configure las políticas y los roles en la cuenta que creará la canalización (Cuenta A)
- Configure las políticas y los roles en la cuenta propietaria del AWS recurso (cuenta B)

## Configure las políticas y los roles en la cuenta que creará la canalización (***Cuenta A***)

Para crear una canalización que utilice CodeDeploy los recursos asociados a otra AWS cuenta, *AccountA* debe configurar políticas tanto para el bucket de Amazon S3 que se utiliza para almacenar artefactos como para el rol de servicio para CodePipeline.

Para crear una política para el bucket de Amazon S3 que concede acceso a AccountB (consola)

1. Inicie sesión en la consola Consola de administración de AWS de Amazon S3 *AccountA* y ábrala en https://console.aws.amazon.com/s3/.
2. En la lista de buckets de Amazon S3, elija el bucket de Amazon S3 en el que se almacenan los artefactos para la canalización. Este bucket tiene un nombre`codepipeline-`*region*`-`*1234567EXAMPLE*, donde *region* corresponde a la AWS región en la que creaste la canalización y *1234567EXAMPLE* es un número aleatorio de diez dígitos que garantiza que el nombre del bucket sea único (por ejemplo,*codepipeline-us-east-2-1234567890*).
3. En la página de detalles del bucket de Amazon S3, elija Propiedades.
4. En el panel de propiedades, expanda Permisos y después elija Añadir política de bucket.

> **Note**
>
> Si ya hay una política asociada al bucket de Amazon S3, elija Editar política de bucket. A continuación, puede añadir las instrucciones del siguiente ejemplo a la política. Para añadir una nueva política, elige el enlace y sigue las instrucciones del generador de AWS políticas. Para obtener más información, consulte Información general de las políticas de IAM.

5. En la ventana Bucket Policy Editor, cree una política que permita *AccountB* acceder a los artefactos de canalización y que permita añadir artefactos de salida si una acción, como una acción de creación o fuente personalizada, los crea. *AccountB*
6. Elija Guardar y, a continuación, cierre el editor de políticas.
7. Elija Guardar para guardar los permisos para el bucket de Amazon S3.

Para crear una política para el rol de servicio de CodePipeline (consola)

1. Inicia sesión en Consola de administración de AWS con *AccountA* y abre la consola de IAM en https://console.aws.amazon.com/iam/.
2. Seleccione Roles en el panel de navegación.
3. En la lista de funciones, bajo Role Name (Nombre de función), elija el nombre de la función de servicio para CodePipeline.
4. En la pestaña Permisos, elija Añadir política en línea.
5. Elige la pestaña JSON e introduce la siguiente política para poder *AccountB* asumir el rol. En el siguiente ejemplo, *012ID_ACCOUNT_B* es el ARN de*AccountB*:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": [
            "arn:aws:iam::111122223333:role/*"
        ]
    }
}
```

6. Elija Revisar política.
7. En Name (Nombre), introduzca un nombre para esta política. Elija Crear política.

## Configure las políticas y los roles en la cuenta propietaria del AWS recurso (*cuenta B*)

Al crear una aplicación, una implementación y un grupo de implementaciones en CodeDeploy, también crea un rol de instancia de Amazon EC2. (Este rol se crea automáticamente cuando se usa el asistente para ejecutar la implementación, pero también puede crearlo manualmente). Para que una canalización creada en *AccountA* utilice CodeDeploy los recursos creados en*AccountB*, debe:

- Configurar una política para el rol de instancia que le permita el acceso al bucket de Amazon S3 donde se almacenan los artefactos de la canalización.
- Cree un segundo rol si *AccountB* está configurado para el acceso entre cuentas.

  Este segundo rol no solo debe tener acceso al bucket de Amazon S3*AccountA*, sino que también debe contener una política que permita el acceso a los CodeDeploy recursos y una política de relación de confianza que permita *AccountA* al rol de CodePipeline servicio asumir el rol.

> **Note**
>
> Estas políticas son específicas para configurar CodeDeploy los recursos que se utilizarán en una canalización creada con una AWS cuenta diferente. Otros AWS recursos requerirán políticas específicas para sus necesidades de recursos.

Para crear una política para el rol de instancia de Amazon EC2 configurado para CodeDeploy (consola)

1. Inicie sesión en Consola de administración de AWS con *AccountB* y abra la consola de IAM en. https://console.aws.amazon.com/iam/
2. Seleccione Roles en el panel de navegación.
3. En la lista de funciones, en Nombre de función, elija el nombre de la función de servicio utilizada como función de instancia de Amazon EC2 para la CodeDeploy aplicación. El nombre de este rol puede variar, y un grupo de implementaciones puede usar más de un rol de instancia. Para obtener más información, consulte Crear un perfil de instancia de IAM para sus instancias de Amazon EC2.
4. En la pestaña Permisos, elija Añadir política en línea.

5. Seleccione la pestaña JSON e introduzca la siguiente política para conceder el acceso al bucket de Amazon S3 que se utiliza *AccountA* para almacenar artefactos para las canalizaciones (en este ejemplo,*codepipeline-us-east-2-1234567890*):

   JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "s3:Get*"
       ],
       "Resource": [
         "arn:aws:s3:::amzn-s3-demo-bucket/*"
       ]
     },
     {
       "Effect": "Allow",
       "Action": [
         "s3:ListBucket"
       ],
       "Resource": [
         "arn:aws:s3:::amzn-s3-demo-bucket"
       ]
     }
   ]
 }
```

6. Elija Revisar política.
7. En Name (Nombre), introduzca un nombre para esta política. Elija Crear política.
8. Cree una segunda política que indique AWS KMS dónde ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** está el ARN de la clave gestionada por el cliente creada *AccountA* y configurada *AccountB* para permitir su uso:

   JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey",
                "kms:GenerateDataKey*",
                "kms:Encrypt",
                "kms:ReEncrypt*",
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:us-
east-1:111122223333:key/2222222-3333333-4444-556677EXAMPLE"
            ]
        }
    ]
}
```

> **Important**
>
> Debe usar el identificador de cuenta de *AccountA* esta política como parte del ARN del recurso para la AWS KMS clave, tal y como se muestra aquí, o la política no funcionará.

9. Elija Revisar política.
10. En Name (Nombre), introduzca un nombre para esta política. Elija Crear política.

Ahora cree un rol de IAM para usarlo en el acceso entre cuentas y configúrelo para que el rol de CodePipeline servicio *AccountA* pueda asumir el rol. Este rol debe contener políticas que permitan el acceso a CodeDeploy los recursos y al bucket de Amazon S3 en el que se almacenan los artefactos. *AccountA*

Para configurar el rol de acceso entre cuentas en IAM

1. Inicie sesión en Consola de administración de AWS con *AccountB* y abra la consola de IAM en https://console.aws.amazon.com/iam.
2. Seleccione Roles en el panel de navegación. Elija Create role (Crear rol).
3. En Seleccione el tipo de entidad de confianza, elija Otra cuenta de AWS ). En Especificar las cuentas que pueden usar este rol, en ID de cuenta, introduce el ID de la AWS cuenta que va

a crear la canalización en CodePipeline (*AccountA*) y, a continuación, selecciona Siguiente: Permisos.

> **Important**
>
> Este paso crea la política de relación de confianza entre *AccountB* y*AccountA*. Sin embargo, esto otorga acceso de nivel raíz a la cuenta y CodePipeline recomienda reducirlo al rol de CodePipeline servicio en*AccountA*. Siga el paso 16 para restringir los permisos.

4. En Adjuntar políticas de permisos AmazonS3ReadOnlyAccess, elija Siguiente: Etiquetas y, a continuación, elija Siguiente.

> **Note**
>
> Esta no es la política que usará. Debe elegir una política para completar el asistente.

5. Elija Siguiente: Revisar. Escriba un nombre para este rol en Nombre del rol (por ejemplo,*CrossAccount_Role*). Puede dar a este rol el nombre que desee, siempre y cuando siga las convenciones de nomenclatura de IAM. Considere dar al rol un nombre que indique claramente su finalidad. Seleccione Crear rol.
6. En la lista de funciones, elija la función que acaba de crear (por ejemplo,*CrossAccount_Role*) para abrir la página de resumen de esa función.
7. En la pestaña Permisos, elija Añadir política en línea.
8. Selecciona la pestaña JSON e introduce la siguiente política para permitir el acceso a CodeDeploy los recursos:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
```

```
        "codedeploy:GetApplicationRevision",
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource": "*"
    }
  ]
}
```

9. Elija Revisar política.
10. En Name (Nombre), introduzca un nombre para esta política. Elija Crear política.
11. En la pestaña Permisos, elija Añadir política en línea.
12. Seleccione la pestaña JSON e introduzca la siguiente política para permitir que este rol recupere los artefactos de entrada del bucket de Amazon S3 y coloque los artefactos de salida en el bucket de Amazon S3*AccountA*:

    JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "s3:GetObject*",
         "s3:PutObject",
         "s3:PutObjectAcl"
       ],
       "Resource": [
         "arn:aws:s3:::amzn-s3-demo-bucket/*"
       ]
     }
   ]
}
```

13. Elija Revisar política.
14. En Name (Nombre), introduzca un nombre para esta política. Elija Crear política.
15. En la pestaña Permisos, busque el nombre de la política AmazonS3ReadOnlyAccess en la lista de políticas y elija el icono de eliminación (X) situado junto a la política. Cuando se le pregunte, elija Separar.

16. Seleccione la pestaña Relación de confianza y, a continuación, Editar la política de confianza. Seleccione la opción Agregar una entidad principal en la columna de la izquierda. Para el tipo de principal, seleccione Funciones de IAM y, a continuación, introduzca el ARN del rol de CodePipeline servicio en. *AccountA* Elimine `arn:aws:iam::Account_A:root` de la lista de Entidades principales de AWS y, a continuación, seleccione Actualizar la política.

## Paso 2: Editar la canalización

No puedes usar la CodePipeline consola para crear o editar una canalización que utilice los recursos asociados a otra AWS cuenta. Sin embargo, puede usar la consola para crear la estructura general de la canalización y, a continuación, usarla AWS CLI para editar la canalización y agregar esos recursos. Como alternativa, puede usar la estructura de una canalización existente y agregarle los recursos manualmente.

Para agregar los recursos asociados a otro AWS cuenta (AWS CLI)

1. En un terminal (Linux, macOS o Unix) o el símbolo del sistema (Windows), ejecute el comando get-pipeline para la canalización a la que desea añadir recursos. Copie el resultado del comando a un archivo JSON. Por ejemplo, para una canalización denominada MyFirstPipeline, debería escribir un texto similar al siguiente:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   El resultado se envía al archivo *pipeline.json*.

2. Abra el archivo JSON en cualquier editor de texto sin formato. Una vez `"type": "S3"` en el almacén de artefactos, añada la clave de cifrado, el ID y la información de tipo de KMS, donde *codepipeline-us-east-2-1234567890* figure el nombre del bucket de Amazon S3 utilizado para almacenar los artefactos para la canalización y el ARN de la clave ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** gestionada por el cliente que acaba de crear:

   ```
   {
     "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. Agregue una acción de implementación en una etapa para usar los CodeDeploy recursos asociados*AccountB*, incluidos los `roleArn` valores del rol multicuenta que creó (). *CrossAccount_Role*

   El siguiente ejemplo muestra un JSON que agrega una acción de implementación denominada*ExternalDeploy*. Utiliza los CodeDeploy recursos creados *AccountB* en una etapa denominada*Staging*. En el siguiente ejemplo, el ARN de *AccountB* es*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**
>
> Este no es el JSON para toda la canalización, sino solo la estructura de la acción en una etapa.

4. Debe eliminar las líneas `metadata` del archivo para que el comando update-pipeline pueda utilizarlo. Quite la sección de la estructura de canalizaciones del archivo JSON (las líneas `"metadata": { }` y los campos `"created"`, `"pipelineARN"` y `"updated"` que contenga).

   Por ejemplo, quite las siguientes líneas de la estructura:

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

   Guarde el archivo.

5. Para aplicar los cambios, ejecute el comando update-pipeline, especificando el archivo JSON de la canalización, de forma similar a como se muestra a continuación:

   > **Important**
   >
   > Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

   Este comando devuelve la estructura completa de la canalización editada.

Para probar la canalización que usa los recursos asociados a otra AWS inscrita

1. En un terminal (Linux, macOS o Unix) o el símbolo del sistema (Windows), ejecute el comando start-pipeline-execution, especificando el nombre de la canalización, de un modo similar al siguiente:

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

Para obtener más información, consulte Iniciar la canalización manualmente.

2. Inicie sesión en Consola de administración de AWS con *AccountA* y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres de todas las canalizaciones asociadas a su AWS cuenta.

3. En Name, elija el nombre de la canalización que acaba de editar. Esto abre una vista detallada de la canalización, que incluye el estado de cada acción en cada etapa de la canalización.

4. Vea el progreso en la canalización. Espere a que aparezca un mensaje de éxito sobre la acción que utiliza el recurso asociado a otra AWS cuenta.

> **Note**
>
> Recibirás un error si intentas ver los detalles de la acción mientras has iniciado sesión con*AccountA*. Cierra sesión y, a continuación, inicia sesión con *AccountB* para ver los detalles de la implementación en CodeDeploy.

# Migrar los canales de sondeo para utilizar la detección de cambios basada en eventos

AWS CodePipeline admite una entrega continua completa de principio a fin, lo que incluye iniciar el proceso cada vez que se produce un cambio de código. Hay dos formas de iniciar la canalización tras un cambio de código: detección de cambios basada en eventos y sondeo. Recomendamos utilizar la detección de cambios basada en eventos para las canalizaciones.

Utilice los procedimientos que se incluyen aquí para migrar (actualizar) sus canalizaciones de sondeo al método de detección de cambios basada en eventos para su canalización.

El método de detección de cambios basado en eventos recomendado para las canalizaciones viene determinado por el origen de la canalización, por ejemplo. CodeCommit En ese caso, por ejemplo, el proceso de sondeo tendría que migrar a una detección de cambios basada en eventos con. EventBridge

# Cómo migrar canalizaciones de sondeo

Para migrar canalizaciones de sondeo, determine sus canalizaciones de sondeo y, a continuación, determine el método recomendado de detección de cambios basada en eventos:

- Siga los pasos que se indican en Cómo ver las canalizaciones de sondeo de su cuenta para determinar sus canalizaciones de sondeo.
- En la tabla, busque el tipo de origen de su canalización y, a continuación, elija el procedimiento con la implementación que desee usar para migrar su canalización de sondeo. Cada sección contiene varios métodos de migración, como el uso de la CLI o CloudFormation.

| Cómo migrar canalizaciones al método de detección de cambios recomendado | | |
|---|---|---|
| Origen de la canalización | Método recomendado de detección basada en eventos | Procedimientos de migración |
| AWS CodeCommit | EventBridge (recomendado). | Consulte Migre las canalizaciones de sondeo con una fuente CodeCommit. |
| Amazon S3 | EventBridge y un bucket habilitado para las notificaciones de eventos (recomendado). | Consulte Migración de canalizaciones de sondeo con un origen de S3 habilitado para los eventos. |
| Amazon S3 | EventBridge y un AWS CloudTrail sendero. | Consulte Migre las canalizaciones de sondeo con una fuente y CloudTrail un registro de S3. |
| GitHub (a través de GitHub la aplicación) | Conexiones (recomendado) | Consulte Migre las canalizaciones de sondeo de una GitHub acción fuente (mediante una aplicación OAuth) a las conexiones. |
| GitHub (a través de la aplicación OAuth) | Webhooks | Consulte Migra los canales de sondeo de una acción fuente GitHub (a través de la aplicación OAuth) a webhooks. |

> **Important**
>
> En el caso de las actualizaciones de configuración de las acciones de canalización aplicables, como las canalizaciones con una acción GitHub (mediante una aplicación de OAuth), debes establecer explícitamente el `PollForSourceChanges` parámetro en falso en la configuración de la acción de origen para evitar que una canalización sea sondeada. Por lo tanto, es posible configurar erróneamente una canalización mediante la detección de cambios basada en eventos y el sondeo, por ejemplo, configurando una EventBridge regla y omitiendo el parámetro. `PollForSourceChanges` Esto se traduce en ejecuciones de canalizaciones duplicadas y la canalización se computa para los límites del número total de canalizaciones que realicen sondeos, lo que de forma predeterminada es mucho menor que las canalizaciones basadas en eventos. Para obtener más información, consulte Cuotas en AWS CodePipeline.

## Cómo ver las canalizaciones de sondeo de su cuenta

Como primer paso, utilice uno de los siguientes scripts para determinar qué canalizaciones de su cuenta están configuradas para el sondeo. Estas son las canalizaciones para migrar a la detección de cambios basada en eventos.

### Ver las canalizaciones de sondeo de la cuenta (script)

Siga estos pasos para determinar con un script las canalizaciones de su cuenta que utilizan el sondeo.

1. Abra una ventana de terminal y, a continuación, realice una de las operaciones siguientes:

   - Ejecute el siguiente comando para crear un nuevo script denominado. PollingPipelinesExtractor.sh

     ```
     vi PollingPipelinesExtractor.sh
     ```

   - Para usar una secuencia de comandos de Python, ejecute el siguiente comando para crear una nueva secuencia de comandos de Python denominada PollingPipelinesExtractor.py.

     ```
     vi PollingPipelinesExtractor.py
     ```

2. Copia y pega el siguiente código en el PollingPipelinesExtractor script. Realice una de las siguientes acciones:

   - Copie y pegue el siguiente código en el PollingPipelinesExtractor.sh script.

```
#!/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."
```

- Copie y pegue el siguiente código en el PollingPipelinesExtractor.py script.

```
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. Deberá ejecutar el script para cada región en la que tenga canalizaciones. Para ejecutar el script, realice una de las siguientes operaciones:

   - Ejecute el siguiente comando para ejecutar el script denominado PollingPipelinesExtractor.sh. En este ejemplo, la región es us-west-2.

     ```
     ./PollingPipelinesExtractor.sh us-west-2
     ```

   - Para la secuencia de comandos de Python, ejecute el siguiente comando para ejecutar la secuencia de comandos de Python denominada PollingPipelinesExtractor.py. En este ejemplo, la región es us-west-2.

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

   En el siguiente ejemplo de salida del script, la región us-west-2 devolvió una lista de canalizaciones de sondeo y muestra la hora de la última ejecución de cada canalización.

   ```
   % ./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...%
```

Analice la salida del script y, para cada canalización de la lista, actualice el origen de sondeo al método recomendado de detección de cambios basada en eventos.

> **Note**
>
> Sus canalizaciones de sondeo vienen determinadas por la configuración de acción de la canalización para el parámetro `PollForSourceChanges`. Si la configuración de la fuente de canalización tiene el `PollForSourceChanges` parámetro omitido, el CodePipeline valor predeterminado es sondear tu repositorio para ver si hay cambios en el código fuente. Este comportamiento es el mismo que si se incluye `PollForSourceChanges` y se establece en true. Para obtener más información, consulte los parámetros de configuración de la acción de origen de su canalización, como los parámetros de configuración de la acción de origen de Amazon S3, en Referencia sobre la acción de origen de Amazon S3.

Tenga en cuenta que este script también genera un archivo.csv que contiene la lista de canalizaciones de sondeo de su cuenta y guarda el archivo.csv en la carpeta de trabajo actual.

## Migre las canalizaciones de sondeo con una fuente CodeCommit

Puede migrar su proceso de sondeo para usarlo EventBridge para detectar cambios en su repositorio de CodeCommit origen o en su bucket de fuentes de Amazon S3.

CodeCommit-- En el caso de una canalización con una CodeCommit fuente, modifique la canalización para que la detección de cambios esté completamente automatizada EventBridge. Elija uno de los siguientes métodos para implementar la migración:

- Consola: Migre las canalizaciones de sondeo (CodeCommit o la fuente de Amazon S3) (consola)
- CLI: Migrar los canales de sondeo (CodeCommit fuente) (CLI)

- CloudFormation: Migrar los canales de sondeo (CodeCommit fuente) (CloudFormation plantilla)

## Migre las canalizaciones de sondeo (CodeCommit o la fuente de Amazon S3) (consola)

Puede usar la CodePipeline consola para actualizar su canalización y usarla EventBridge para detectar cambios en su repositorio de CodeCommit origen o en su depósito de fuentes de Amazon S3.

> **Note**
>
> Cuando usas la consola para editar una canalización que tiene un repositorio de CodeCommit origen o un bucket de origen de Amazon S3, la regla y el rol de IAM se crean automáticamente. Si utilizas el AWS CLI para editar la canalización, debes crear tú mismo la EventBridge regla y el rol de IAM. Para obtener más información, consulte CodeCommit acciones de origen y EventBridge.

Utilice estos pasos para editar una canalización que utiliza las comprobaciones periódicas. Si desea crear una canalización, consulte Creación de una canalización, etapas y acciones.

Para editar la etapa de código fuente de la canalización

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres de todas las canalizaciones asociadas a su AWS cuenta.
2. En Nombre, elija el nombre de la canalización que desea editar. Esto abre una vista detallada de la canalización, que incluye el estado de cada una de las acciones en cada etapa de la canalización.
3. En la página de detalles de la canalización, elija Edit.
4. En la etapa Edit (Editar), elija el icono de edición en la acción de código fuente.
5. Amplíe las opciones de detección de cambios y seleccione Usar CloudWatch eventos para iniciar automáticamente mi canalización cuando se produzca un cambio (recomendado).

   Aparece un mensaje que muestra la EventBridge regla que se va a crear para esta canalización. Elija Actualizar.

Si está actualizando una canalización que tiene origen en Amazon S3, aparecerá el mensaje siguiente. Elija Actualizar.

6. Cuando haya terminado de editar la canalización, elija Save pipeline changes para volver a la página de resumen.

   Un mensaje muestra el nombre de la EventBridge regla que se va a crear para tu canalización. Elija Guardar y continuar.

7. Para probar tu acción, publica un cambio utilizando el AWS CLI para confirmar un cambio en la fuente especificada en la fase de origen de la canalización.

## Migrar los canales de sondeo (CodeCommit fuente) (CLI)

Siga estos pasos para editar una canalización que utiliza sondeos (comprobaciones periódicas) para usar una EventBridge regla para iniciar la canalización. Si desea crear una canalización, consulte Creación de una canalización, etapas y acciones.

Para crear una canalización basada en eventos CodeCommit, debes editar el `PollForSourceChanges` parámetro de tu canalización y, a continuación, crear los siguientes recursos:

- EventBridge evento
- Rol de IAM para permitir que este evento inicie la canalización

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

1. Ejecute el comando get-pipeline para copiar la estructura de canalización en un archivo JSON. Por ejemplo, para una canalización denominada `MyFirstPipeline`, escriba el siguiente comando:

```
aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
```

   Este comando no devuelve nada, pero el archivo creado debería aparecer en el directorio en el que se ejecutó el comando.

2. Abra el archivo JSON en cualquier editor de texto sin formato y edite la etapa de origen cambiando el parámetro `PollForSourceChanges` por `false`, tal y como se muestra en este ejemplo.

   ¿Por qué voy a hacer este cambio? Al cambiar este parámetro a `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

```
"configuration": {
    "PollForSourceChanges": "false",
    "BranchName": "main",
    "RepositoryName": "MyTestRepo"
},
```

3. Si está trabajando con la estructura de la canalización recuperada mediante el comando get-pipeline, elimine las líneas `metadata` del archivo JSON. De lo contrario, el comando update-pipeline no puede utilizarlo. Elimine las líneas `"metadata": { }` y los campos `"updated"`, `"created"` y `"pipelineARN"`.

   Por ejemplo, quite las siguientes líneas de la estructura:

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

   Guarde el archivo.

4. Para aplicar los cambios, ejecute el comando update-pipeline especificando el archivo JSON:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

Este comando devuelve la estructura completa de la canalización editada.

> **Note**
>
> El comando update-pipeline detiene la canalización. Si se está ejecutando una revisión en la canalización al ejecutar el comando update-pipeline, dicha ejecución se detiene. Debe iniciar manualmente la canalización para ejecutar dicha revisión en la canalización actualizada. Utilice el comando **`start-pipeline-execution`** para iniciar manualmente la canalización.

Para crear una EventBridge CodeCommit regla con el origen y CodePipeline el destino del evento

1. Agregue los permisos EventBridge para usarlos CodePipeline para invocar la regla. Para obtener más información, consulte Uso de políticas basadas en recursos para Amazon EventBridge

   a. Usa el siguiente ejemplo para crear la política de confianza que te permita EventBridge asumir el rol de servicio. Ponga un nombre a la política de confianza `trustpolicyforEB.json`.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Utilice el comando para crear el rol `Role-for-MyRule` y asocie la política de confianza.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Cree el JSON de la política de permisos, tal y como se muestra en este ejemplo para la canalización denominada `MyFirstPipeline`. Ponga un nombre a la política de permisos `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilice el siguiente comando para asociar la política de permisos `CodePipeline-Permissions-Policy-for-EB` al rol `Role-for-MyRule`.

¿Por qué voy a hacer este cambio? Al agregar esta política al rol, se crean permisos para EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Llame al comando put-rule e incluya los parámetros `--name`, `--event-pattern` y `--role-arn`.

   ¿Por qué voy a hacer este cambio? Este comando permite que CloudFormation cree el evento.

   El siguiente comando de ejemplo crea una regla llamada `MyCodeCommitRepoRule`.

```
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. Para agregarla CodePipeline como destino, ejecute el put-targets comando e incluya los siguientes parámetros:

   - El parámetro `--rule` se usa con el `rule_name` que creó con el comando put-rule.
   - El parámetro `--targets` se usa con el `Id` del destino de la lista de destinos y el ARN de la canalización de destino.

   El siguiente comando de muestra especifica que, para la regla denominada `MyCodeCommitRepoRule`, el destino `Id` se compone del número uno, lo que indica que, en lo que puede ser una lista de destinos de la regla, se trata del destino 1. El comando de muestra también especifica un ARN de ejemplo para la canalización. La canalización se inicia cuando se produce algún cambio en el repositorio.

```
aws events put-targets --rule MyCodeCommitRepoRule --targets
 Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

4. (Opcional) Para configurar un transformador de entrada con anulaciones de origen para un ID de imagen específico, utilice el siguiente fragmento de código JSON en el comando de la CLI. En el siguiente ejemplo, se configura una anulación donde:

   - `actionName` (`Source` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.

- `revisionType` (`COMMIT_ID` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
- En este ejemplo`revisionValue`, **<** *`revisionValue`* **>** se deriva de la variable de evento de origen.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "pipeline-ARN",
            "InputTransformer": {
                "sourceRevisions": {
                    "actionName": "Source",
                    "revisionType": "COMMIT_ID",
                    "revisionValue": "<revisionValue>"
                },
                "variables": [
                    {
                        "name": "Branch_Name",
                        "value": "value"
                    }
                ]
            }
        }
    ]
}
```

## Migrar los canales de sondeo (CodeCommit fuente) (CloudFormation plantilla)

Para crear una canalización basada en eventos AWS CodeCommit, debes editar el `PollForSourceChanges` parámetro de tu canalización y, a continuación, añadir los siguientes recursos a tu plantilla:

- Una regla EventBridge
- Una función de IAM para su regla EventBridge

Si la utilizas CloudFormation para crear y gestionar tus canalizaciones, la plantilla incluye contenido como el siguiente.

> **Note**
>
> La propiedad `Configuration` en la etapa de código fuente denominada `PollForSourceChanges`. Si dicha propiedad no está incluida en la plantilla, `PollForSourceChanges` se establece en `true` de forma predeterminada.

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
        }]
    },
```

Para actualizar tu canalización CloudFormation crear una plantilla y crear una EventBridge regla

1. En la sección inferior de la plantilla`Resources`, usa el `AWS::IAM::Role` CloudFormation recurso para configurar la función de IAM que permitirá que tu evento inicie su proceso. Esta entrada crea un rol que utiliza dos políticas:

   - La primera política permite asumir el rol.
   - La segunda política concede permisos para iniciar la canalización.

   ¿Por qué voy a hacer este cambio? Añadir el `AWS::IAM::Role` recurso permite CloudFormation crear permisos para EventBridge. Este recurso se agrega a su CloudFormation pila.

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. En la plantilla, en`Resources`, usa el `AWS::Events::Rule` CloudFormation recurso para agregar una EventBridge regla. Este patrón de eventos crea un evento que monitoriza la introducción de cambios en su repositorio. Cuando EventBridge detecta un cambio en el estado del repositorio, la regla se invoca `StartPipelineExecution` en tu canalización de destino.

   ¿Por qué voy a hacer este cambio? Añadir el `AWS::Events::Rule` recurso permite CloudFormation crear el evento. Este recurso se añade a tu CloudFormation pila.

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. (Opcional) Para configurar un transformador de entrada con anulaciones de origen para un ID de imagen específico, utilice el siguiente fragmento de código YAML. En el siguiente ejemplo, se configura una anulación donde:

   - `actionName` (`Source` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - `revisionType` (`COMMIT_ID` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - En este ejemplo`revisionValue`, **<** *`revisionValue`* **>** se deriva de la variable de evento de origen.
   - Se especifican las variables de salida para `BranchName` y `Value`.

```
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
```

```
  InputTransformer:
    sourceRevisions:
      actionName: Source
      revisionType: COMMIT_ID
      revisionValue: <revisionValue>
    variables:
    - name: BranchName
      value: value
```

4. Guarde la plantilla actualizada en su equipo local y, a continuación, abra la CloudFormation consola.
5. Seleccione la pila y luego elija Create Change Set for Current Stack (Crear conjuntos de cambios para la pila actual).
6. Cargue la plantilla y, a continuación, consulte los cambios indicados en CloudFormation. Estos son los cambios que se realizan en la pila. Debería ver los nuevos recursos en la lista.
7. Elija Ejecutar.

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> En muchos casos, el parámetro `PollForSourceChanges` es true de forma predeterminada al crear una canalización. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

- En la plantilla, cambie `PollForSourceChanges` por `false`. Si no ha incluido `PollForSourceChanges` en la definición de la canalización, añádalo y establézcalo en `false`.

  ¿Por qué voy a hacer este cambio? Al cambiar este parámetro a `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

  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

Cuando creas estos recursos con CloudFormation, tu canalización se activa cuando se crean o actualizan los archivos de tu repositorio. Este es el fragmento final de la plantilla:

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
                        }
                    ]
                },

...
```

## Migración de canalizaciones de sondeo con un origen de S3 habilitado para los eventos

En el caso de una canalización con una fuente de Amazon S3, modifique la canalización para que la detección de cambios se EventBridge automatice mediante y con un bucket de origen que esté habilitado para las notificaciones de eventos. Este es el método recomendado si utiliza la CLI o CloudFormation para migrar su canalización.

> **Note**
>
> Esto incluye el uso de un bucket que esté habilitado para las notificaciones de eventos, en el que no es necesario crear un CloudTrail registro independiente. Si utilizas la consola, tendrás configuradas una regla CloudTrail y un registro del evento. Para los siguientes pasos, consulte Migre las canalizaciones de sondeo con una fuente y CloudTrail un registro de S3.

- CLI: Migre las canalizaciones de sondeo con una fuente y un CloudTrail registro (CLI) de S3
- CloudFormation: Migre las canalizaciones de sondeo con una fuente y un CloudTrail registro de S3 (CloudFormation plantilla)

### Migrar los canales de sondeo con origen de S3 habilitado para los eventos (CLI)

Sigue estos pasos para editar una canalización que esté usando sondeos (comprobaciones periódicas) para usar un evento en EventBridge su lugar. Si desea crear una canalización, consulte Creación de una canalización, etapas y acciones.

Para crear una canalización basada en eventos con Amazon S3, debe editar el parámetro `PollForSourceChanges` de la canalización y, a continuación, crear los siguientes recursos manualmente:

- EventBridge regla de evento
- Función de IAM para permitir que el EventBridge evento inicie su proceso

Para crear una EventBridge regla con Amazon S3 como origen y CodePipeline destino del evento y aplicar la política de permisos

1. Otorgue los permisos EventBridge para usarlos CodePipeline para invocar la regla. Para obtener más información, consulte Uso de políticas basadas en recursos para Amazon. EventBridge

   a. Utilice el siguiente ejemplo para crear la política de confianza que permite que EventBridge asuma el rol de servicio. Denomínelo `trustpolicyforEB.json`.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   b. Utilice el comando para crear el rol `Role-for-MyRule` y asocie la política de confianza.

      ¿Por qué voy a hacer este cambio? Al añadir esta política de confianza al rol, se crean permisos para EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Cree el JSON de la política de permisos tal y como se muestra aquí para la canalización denominada `MyFirstPipeline`. Ponga un nombre a la política de permisos `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilice el siguiente comando para asociar la nueva política de permisos `CodePipeline-Permissions-Policy-for-EB` al rol `Role-for-MyRule` que ha creado.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Llame al comando put-rule e incluya los parámetros `--name`, `--event-pattern` y `--role-arn`.

   El siguiente comando de ejemplo crea una regla denominada `EnabledS3SourceRule`.

```
aws events put-rule --name "EnabledS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"Object Created\"],\"detail\":{\"bucket\":{\"name\":
[\"amzn-s3-demo-source-bucket\"]}}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-
for-MyRule"
```

3. Para añadirlo CodePipeline como objetivo, ejecuta el put-targets comando e incluye los parámetros `--rule` y`--targets`.

El siguiente comando especifica que, para la regla denominada `EnabledS3SourceRule`, el destino `Id` se compone del número uno, lo que indica que, en lo que puede ser una lista de destinos de la regla, se trata del destino 1. El comando también especifica un ARN de ejemplo para la canalización. La canalización se inicia cuando se produce algún cambio en el repositorio.

```
aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-
AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

1. Ejecute el comando get-pipeline para copiar la estructura de canalización en un archivo JSON. Por ejemplo, para una canalización denominada `MyFirstPipeline`, escriba el siguiente comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando no devuelve nada, pero el archivo creado debería aparecer en el directorio en el que se ejecutó el comando.

2. Abra el archivo JSON en cualquier editor de texto sin formato y edite la etapa de código fuente cambiando el parámetro `PollForSourceChanges` del bucket denominado `amzn-s3-demo-source-bucket` a `false`, tal y como se muestra en este ejemplo.

   ¿Por qué voy a hacer este cambio? Al establecer este parámetro en `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

```
"configuration": {
    "S3Bucket": "amzn-s3-demo-source-bucket",
    "PollForSourceChanges": "false",
    "S3ObjectKey": "index.zip"
},
```

3. Si está trabajando con la estructura de la canalización recuperada mediante el comando get-pipeline, debe eliminar las líneas `metadata` del archivo JSON. De lo contrario, el comando update-pipeline no puede utilizarlo. Elimine las líneas `"metadata": { }` y los campos `"updated"`, `"created"` y `"pipelineARN"`.

   Por ejemplo, quite las siguientes líneas de la estructura:

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

   Guarde el archivo.

4. Para aplicar los cambios, ejecute el comando update-pipeline especificando el archivo JSON:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

Este comando devuelve la estructura completa de la canalización editada.

> **Note**
>
> El comando update-pipeline detiene la canalización. Si se está ejecutando una revisión en la canalización al ejecutar el comando update-pipeline, dicha ejecución se detiene. Debe iniciar manualmente la canalización para ejecutar dicha revisión en la canalización

actualizada. Utilice el comando start-pipeline-execution para iniciar manualmente la canalización.

## Migre las canalizaciones de sondeo con una fuente S3 habilitada para los eventos (CloudFormation plantilla)

Este procedimiento es para una canalización en la que el bucket de origen tiene los eventos habilitados.

Utilice estos pasos para editar una canalización con origen en Amazon S3 que usa sondeos para que use la detección de cambios basada en eventos.

Para crear una canalización basada en eventos con Amazon S3, deberá editar el parámetro `PollForSourceChanges` de la canalización y, a continuación, añadir los siguientes recursos a su plantilla:

- EventBridge regla y rol de IAM para permitir que este evento inicie tu canalización.

Si la usas CloudFormation para crear y administrar tus canalizaciones, tu plantilla incluye contenido como el siguiente.

> **Note**
>
> La propiedad `Configuration` en la etapa de código fuente denominada `PollForSourceChanges`. Si la plantilla no incluye esa propiedad, `PollForSourceChanges` se establece en `true` de forma predeterminada.

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
                        }
                    ]
                },


...
```

Para crear una EventBridge regla con Amazon S3 como origen y CodePipeline destino del evento y aplicar la política de permisos

1. En la sección de abajo de la plantilla`Resources`, utilice el `AWS::IAM::Role` CloudFormation recurso para configurar el rol de IAM que permite que su evento inicie su canalización. Esta entrada crea un rol que utiliza dos políticas:

   - La primera política permite asumir el rol.
   - La segunda política concede permisos para iniciar la canalización.

   ¿Por qué voy a hacer este cambio? Añadir `AWS::IAM::Role` un recurso permite CloudFormation crear permisos para EventBridge. Este recurso se agrega a su CloudFormation pila.

   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. Usa el `AWS::Events::Rule` CloudFormation recurso para agregar una EventBridge regla. Este patrón de eventos crea un evento que monitoriza la creación o eliminación de objetos en el bucket de origen de Amazon S3. Además, incluya un objetivo de la canalización. Cuando se crea un objeto, esta regla invoca `StartPipelineExecution` en la canalización de destino.

   ¿Por qué voy a hacer este cambio? Agregar el `AWS::Events::Rule` recurso permite CloudFormation crear el evento. Este recurso se añade a tu CloudFormation pila.

   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. Guarde la plantilla actualizada en el equipo local y abra la consola de CloudFormation .
4. Seleccione la pila y luego elija Create Change Set for Current Stack (Crear conjuntos de cambios para la pila actual).

5. Cargue su plantilla actualizada y, a continuación, consulte los cambios indicados en CloudFormation. Estos son los cambios que se realizarán en la pila. Debería ver los nuevos recursos en la lista.
6. Elija Ejecutar.

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

- En la plantilla, cambie `PollForSourceChanges` por `false`. Si no ha incluido `PollForSourceChanges` en la definición de la canalización, añádalo y establézcalo en `false`.

  ¿Por qué voy a hacer este cambio? Al cambiar `PollForSourceChanges` a `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

  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

Cuando los usas CloudFormation para crear estos recursos, tu canalización se activa cuando se crean o actualizan los archivos de tu repositorio.

> **Note**
>
> No se detenga aquí. Aunque su canalización esté creada, debe crear una segunda CloudFormation plantilla para su canalización de Amazon S3. Si no crea la segunda plantilla, la canalización no tendrá ninguna funcionalidad de detección de cambios.

YAML

```
Parameters:
  SourceObjectKey:
    Description: 'S3 source artifact'
    Type: String
    Default: SampleApp_Linux.zip
  ApplicationName:
    Description: 'CodeDeploy application name'
    Type: String
    Default: DemoApplication
  BetaFleet:
    Description: 'Fleet configured in CodeDeploy'
    Type: String
    Default: DemoFleet

Resources:
  SourceBucket:
    Type: AWS::S3::Bucket
    Properties:
      NotificationConfiguration:
        EventBridgeConfiguration:
          EventBridgeEnabled: true
      VersioningConfiguration:
        Status: Enabled
  CodePipelineArtifactStoreBucket:
    Type: AWS::S3::Bucket
  CodePipelineArtifactStoreBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref CodePipelineArtifactStoreBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          -
```

```
            Sid: DenyUnEncryptedObjectUploads
            Effect: Deny
            Principal: '*'
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*' ] ]
            Condition:
              StringNotEquals:
                s3:x-amz-server-side-encryption: aws:kms
          -
            Sid: DenyInsecureConnections
            Effect: Deny
            Principal: '*'
            Action: s3:*
            Resource: !Sub ${CodePipelineArtifactStoreBucket.Arn}/*
            Condition:
              Bool:
                aws:SecureTransport: false
  CodePipelineServiceRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - codepipeline.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: AWS-CodePipeline-Service-3
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action:
                  - codecommit:CancelUploadArchive
                  - codecommit:GetBranch
                  - codecommit:GetCommit
                  - codecommit:GetUploadArchiveStatus
```

```
      - codecommit:UploadArchive
    Resource: 'resource_ARN'
  -
    Effect: Allow
    Action:
      - codedeploy:CreateDeployment
      - codedeploy:GetApplicationRevision
      - codedeploy:GetDeployment
      - codedeploy:GetDeploymentConfig
      - codedeploy:RegisterApplicationRevision
    Resource: 'resource_ARN'
  -
    Effect: Allow
    Action:
      - codebuild:BatchGetBuilds
      - codebuild:StartBuild
    Resource: 'resource_ARN'
  -
    Effect: Allow
    Action:
      - devicefarm:ListProjects
      - devicefarm:ListDevicePools
      - devicefarm:GetRun
      - devicefarm:GetUpload
      - devicefarm:CreateUpload
      - devicefarm:ScheduleRun
    Resource: 'resource_ARN'
  -
    Effect: Allow
    Action:
      - lambda:InvokeFunction
      - lambda:ListFunctions
    Resource: 'resource_ARN'
  -
    Effect: Allow
    Action:
      - iam:PassRole
    Resource: 'resource_ARN'
  -
    Effect: Allow
    Action:
      - elasticbeanstalk:*
      - ec2:*
      - elasticloadbalancing:*
```

```
                - autoscaling:*
                - cloudwatch:*
                - s3:*
                - sns:*
                - cloudformation:*
                - rds:*
                - sqs:*
                - ecs:*
              Resource: 'resource_ARN'
AppPipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
    Name: s3-events-pipeline
    RoleArn:
      !GetAtt CodePipelineServiceRole.Arn
    Stages:
      -
        Name: Source
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
              - Name: SourceOutput
            Configuration:
              S3Bucket: !Ref SourceBucket
              S3ObjectKey: !Ref SourceObjectKey
              PollForSourceChanges: false
            RunOrder: 1
      -
        Name: Beta
        Actions:
          -
            Name: BetaAction
            InputArtifacts:
              - Name: SourceOutput
            ActionTypeId:
              Category: Deploy
              Owner: AWS
              Version: 1
```

```
              Provider: CodeDeploy
            Configuration:
              ApplicationName: !Ref ApplicationName
              DeploymentGroupName: !Ref BetaFleet
            RunOrder: 1
    ArtifactStore:
      Type: S3
      Location: !Ref CodePipelineArtifactStoreBucket
 EventRole:
   Type: AWS::IAM::Role
   Properties:
     AssumeRolePolicyDocument:
       Version: 2012-10-17
       Statement:
         -
           Effect: Allow
           Principal:
             Service:
               - events.amazonaws.com
           Action: sts:AssumeRole
     Path: /
     Policies:
       -
         PolicyName: eb-pipeline-execution
         PolicyDocument:
           Version: 2012-10-17
           Statement:
             -
               Effect: Allow
               Action: codepipeline:StartPipelineExecution
               Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
 EventRule:
   Type: AWS::Events::Rule
   Properties:
     EventBusName: default
     EventPattern:
       source:
         - aws.s3
       detail-type:
         - Object Created
       detail:
         bucket:
           name:
```

```
              - !Ref SourceBucket
      Name: EnabledS3SourceRule
      State: ENABLED
      Targets:
        -
          Arn:
            !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
          RoleArn: !GetAtt EventRole.Arn
          Id: codepipeline-AppPipeline
```

JSON

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "appconfig:StartDeployment",
                "appconfig:StopDeployment",
                "appconfig:GetDeployment"
            ],
            "Resource": [
                "arn:aws:appconfig:*:111122223333:application/
[[Application]]",
                "arn:aws:appconfig:*:111122223333:application/[[Application]]/
*",
                "arn:aws:appconfig:*:111122223333:deploymentstrategy/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

# Migre las canalizaciones de sondeo con una fuente y CloudTrail un registro de S3

En el caso de una canalización con una fuente de Amazon S3, modifique la canalización para que la detección de cambios esté completamente automatizada. EventBridge Elija uno de los siguientes métodos para implementar la migración:

- Consola: Migre las canalizaciones de sondeo (CodeCommit o la fuente de Amazon S3) (consola)
- CLI: Migre las canalizaciones de sondeo con una fuente y un CloudTrail registro (CLI) de S3
- CloudFormation: Migre las canalizaciones de sondeo con una fuente y un CloudTrail registro de S3 (CloudFormation plantilla)

## Migre las canalizaciones de sondeo con una fuente y un CloudTrail registro (CLI) de S3

Siga estos pasos para editar una canalización que utilice sondeos (comprobaciones periódicas) para utilizar un evento en EventBridge su lugar. Si desea crear una canalización, consulte Creación de una canalización, etapas y acciones.

Para crear una canalización basada en eventos con Amazon S3, debe editar el parámetro `PollForSourceChanges` de la canalización y, a continuación, crear los siguientes recursos manualmente:

- AWS CloudTrail política de seguimiento, depósito y depósito que Amazon S3 puede utilizar para registrar los eventos.
- EventBridge evento
- Función de IAM para permitir que el EventBridge evento inicie su tramitación

Para crear un AWS CloudTrail rastrear y habilitar el registro

Para usar el AWS CLI para crear una ruta, ejecute el create-trail comando y especifique:

- El nombre del registro de seguimiento.
- El bucket al que ya haya aplicado la política de bucket para AWS CloudTrail.

Para obtener más información, consulte Crear una ruta con la interfaz de línea de AWS comandos.

1. Llame al comando create-trail e incluya los parámetros `--name` y `--s3-bucket-name`.

   ¿Por qué voy a hacer este cambio? Esto crea el registro de seguimiento de CloudTrail necesario para su bucket de origen de S3.

   El comando siguiente utiliza `--name` y `--s3-bucket-name` para crear un registro de seguimiento llamado `my-trail` y un bucket llamado `amzn-s3-demo-source-bucket`.

   ```
   aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-source-bucket
   ```

2. Use el comando start-logging e incluya el parámetro `--name`.

   ¿Por qué voy a hacer este cambio? Este comando inicia el CloudTrail registro del bucket de origen y envía los eventos a EventBridge.

   Ejemplo:

   El siguiente comando utiliza `--name` para iniciar el registro en un registro de seguimiento llamado `my-trail`.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Llame al comando put-event-selectors e incluya los parámetros `--trail-name` y `--event-selectors`. Usa los selectores de eventos para especificar que quieres que tu ruta registre eventos de datos para tu bucket de origen y envíe los eventos a la EventBridge regla.

   ¿Por qué voy a hacer este cambio? Este comando filtra eventos.

   Ejemplo:

   El siguiente comando utiliza `--trail-name` y `--event-selectors` para especificar eventos de datos para un bucket de origen y un prefijo llamados `amzn-s3-demo-source-bucket/myFolder`.

   ```
   aws cloudtrail put-event-selectors --trail-name my-trail --event-selectors
    '[{ "ReadWriteType": "WriteOnly", "IncludeManagementEvents":false,
    "DataResources": [{ "Type": "AWS::S3::Object", "Values": ["arn:aws:s3:::amzn-s3-demo-source-bucket/myFolder/file.zip"] }] }]'
   ```

Para crear una EventBridge regla con Amazon S3 como origen y destino del evento CodePipeline , y aplicar la política de permisos

1. Otorgue los permisos EventBridge para usarlos CodePipeline para invocar la regla. Para obtener más información, consulte Uso de políticas basadas en recursos para Amazon. EventBridge

   a. Usa el siguiente ejemplo para crear la política de confianza que te permita EventBridge asumir la función de servicio. Denomínelo `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Utilice el comando para crear el rol `Role-for-MyRule` y asocie la política de confianza.

      ¿Por qué voy a hacer este cambio? Al agregar esta política de confianza al rol, se crean permisos para EventBridge.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Cree el JSON de la política de permisos tal y como se muestra aquí para la canalización denominada `MyFirstPipeline`. Ponga un nombre a la política de permisos `permissionspolicyforEB.json`.

      JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilice el siguiente comando para asociar la nueva política de permisos `CodePipeline-Permissions-Policy-for-EB` al rol `Role-for-MyRule` que ha creado.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Llame al comando put-rule e incluya los parámetros `--name`, `--event-pattern` y `--role-arn`.

   El siguiente comando de ejemplo crea una regla denominada `MyS3SourceRule`.

```
aws events put-rule --name "MyS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"AWS API Call via CloudTrail\"],\"detail\":
{\"eventSource\":[\"s3.amazonaws.com\"],\"eventName\":[\"CopyObject\",\"PutObject
\",\"CompleteMultipartUpload\"],\"requestParameters\":{\"bucketName\":[\"amzn-s3-
demo-source-bucket\"],\"key\":[\"my-key\"]}}}
 --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. Para CodePipeline añadirla como destino, ejecute el put-targets comando e incluya los `--targets` parámetros `--rule` y.

   El siguiente comando especifica que, para la regla denominada `MyS3SourceRule`, el destino `Id` se compone del número uno, lo que indica que, en lo que puede ser una lista de destinos de la regla, se trata del destino 1. El comando también especifica un ARN de ejemplo para la canalización. La canalización se inicia cuando se produce algún cambio en el repositorio.

```
aws events put-targets --rule MyS3SourceRule --targets
 Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

4. (Opcional) Para configurar un transformador de entrada con anulaciones de origen para un ID de imagen específico, utilice el siguiente fragmento de código JSON en el comando de la CLI. En el siguiente ejemplo, se configura una anulación donde:

   - `actionName` (`Source` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - `revisionType` (`S3_OBJECT_VERSION_ID` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - En este ejemplo`revisionValue`, **<** *`revisionValue`* **>** se deriva de la variable de evento de origen.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "ARN",
            "InputTransformer": {
                "InputPathsMap": {
                    "revisionValue": "$.detail.object.version-id"
                },
                "InputTemplate": {
                    "sourceRevisions": {
                        "actionName": "Source",
                        "revisionType": "S3_OBJECT_VERSION_ID",
                        "revisionValue": "<revisionValue>"
                    }
                }
            }
        }
    ]
}
```

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

1. Ejecute el comando get-pipeline para copiar la estructura de canalización en un archivo JSON. Por ejemplo, para una canalización denominada `MyFirstPipeline`, escriba el siguiente comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando no devuelve nada, pero el archivo creado debería aparecer en el directorio en el que se ejecutó el comando.

2. Abra el archivo JSON en cualquier editor de texto sin formato y edite la etapa de código fuente cambiando el parámetro `PollForSourceChanges` del bucket denominado `amzn-s3-demo-source-bucket` a `false`, tal y como se muestra en este ejemplo.

   ¿Por qué voy a hacer este cambio? Al establecer este parámetro en `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Si está trabajando con la estructura de la canalización recuperada mediante el comando get-pipeline, debe eliminar las líneas `metadata` del archivo JSON. De lo contrario, el comando update-pipeline no puede utilizarlo. Elimine las líneas `"metadata": { }` y los campos `"updated"`, `"created"` y `"pipelineARN"`.

Por ejemplo, quite las siguientes líneas de la estructura:

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

Guarde el archivo.

4. Para aplicar los cambios, ejecute el comando update-pipeline especificando el archivo JSON:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

Este comando devuelve la estructura completa de la canalización editada.

> **Note**
>
> El comando update-pipeline detiene la canalización. Si se está ejecutando una revisión en la canalización al ejecutar el comando update-pipeline, dicha ejecución se detiene. Debe iniciar manualmente la canalización para ejecutar dicha revisión en la canalización actualizada. Utilice el comando start-pipeline-execution para iniciar manualmente la canalización.

## Migre las canalizaciones de sondeo con una fuente y un CloudTrail registro de S3 (CloudFormation plantilla)

Utilice estos pasos para editar una canalización con origen en Amazon S3 que usa sondeos para que use la detección de cambios basada en eventos.

Para crear una canalización basada en eventos con Amazon S3, deberá editar el parámetro `PollForSourceChanges` de la canalización y, a continuación, añadir los siguientes recursos a su plantilla:

- EventBridge requiere que se registren todos los eventos de Amazon S3. Debe crear un registro de seguimiento de AWS CloudTrail , un bucket y una política de bucket que Amazon S3 pueda usar para registrar los eventos que se produzcan. Para obtener más información, consulte Registro de eventos de datos y Registro de eventos de gestión para registros de seguimiento.
- EventBridge regla y rol de IAM para permitir que este evento inicie su canalización.

Si la usas CloudFormation para crear y administrar tus canalizaciones, tu plantilla incluye contenido como el siguiente.

> **Note**
>
> La propiedad `Configuration` en la etapa de código fuente denominada `PollForSourceChanges`. Si la plantilla no incluye esa propiedad, `PollForSourceChanges` se establece en `true` de forma predeterminada.

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
                            }
                        ]
                    },


...
```

Para crear una EventBridge regla con Amazon S3 como origen y CodePipeline destino del evento y aplicar la política de permisos

1. En la sección de abajo de la plantilla`Resources`, utilice el `AWS::IAM::Role` CloudFormation recurso para configurar el rol de IAM que permite que su evento inicie su canalización. Esta entrada crea un rol que utiliza dos políticas:

   - La primera política permite asumir el rol.
   - La segunda política concede permisos para iniciar la canalización.

   ¿Por qué voy a hacer este cambio? Añadir `AWS::IAM::Role` un recurso permite CloudFormation crear permisos para EventBridge. Este recurso se agrega a su CloudFormation pila.

   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. Usa el `AWS::Events::Rule` CloudFormation recurso para agregar una EventBridge regla. Este patrón de eventos crea un evento que monitoriza `CopyObject`, `PutObject` y `CompleteMultipartUpload` en el bucket de origen de Amazon S3. Además, incluya un objetivo de la canalización. Cuando se produce `CopyObject`, `PutObject` o `CompleteMultipartUpload`, esta regla invoca `StartPipelineExecution` en la canalización de destino.

   ¿Por qué voy a hacer este cambio? Agregar el `AWS::Events::Rule` recurso permite CloudFormation crear el evento. Este recurso se añade a tu CloudFormation pila.

   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. Añada este fragmento a su primera plantilla para permitir la funcionalidad de pila cruzada:

   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. (Opcional) Para configurar un transformador de entrada con anulaciones de origen para un ID de imagen específico, utilice el siguiente fragmento de código YAML. En el siguiente ejemplo, se configura una anulación donde:

   - `actionName` (`Source` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - `revisionType` (`S3_OBJECT_VERSION_ID` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - En este ejemplo`revisionValue`, **<** *`revisionValue`* **>** se deriva de la variable de evento de origen.

```
---
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
    InputPathsMap:
      revisionValue: "$.detail.object.version-id"
    InputTemplate:
      sourceRevisions:
        actionName: Source
        revisionType: S3_OBJECT_VERSION_ID
        revisionValue: '<revisionValue>'
```

5. Guarde la plantilla actualizada en su equipo local y abra la CloudFormation consola.
6. Seleccione la pila y luego elija Create Change Set for Current Stack (Crear conjuntos de cambios para la pila actual).
7. Cargue su plantilla actualizada y, a continuación, consulte los cambios indicados en CloudFormation. Estos son los cambios que se realizarán en la pila. Debería ver los nuevos recursos en la lista.
8. Elija Ejecutar.

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

- En la plantilla, cambie `PollForSourceChanges` por `false`. Si no ha incluido `PollForSourceChanges` en la definición de la canalización, añádalo y establézcalo en `false`.

¿Por qué voy a hacer este cambio? Al cambiar `PollForSourceChanges` a `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

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
  }
```

Para crear una segunda plantilla para los CloudTrail recursos de su canalización de Amazon S3

- En una plantilla independiente, en`Resources`, utilice los `AWS::S3::Bucket` `AWS::CloudTrail::Trail` CloudFormation recursos y para proporcionar una definición y un registro sencillos del bucket CloudTrail. `AWS::S3::BucketPolicy`

  ¿Por qué voy a hacer este cambio? Dado el límite actual de cinco rutas por cuenta, la CloudTrail ruta se debe crear y gestionar por separado. (Consulta la sección Límites en AWS CloudTrail.) Sin embargo, puede incluir muchos buckets de Amazon S3 en un único registro de seguimiento, por lo que puede crear el registro de seguimiento una vez y, a continuación, añadir buckets de Amazon S3 para otras canalizaciones según sea necesario. Pegue lo siguiente en el segundo archivo de plantilla de ejemplo.

  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

Cuando los usas CloudFormation para crear estos recursos, tu canalización se activa cuando se crean o actualizan los archivos de tu repositorio.

> **Note**
>
> No se detenga aquí. Aunque su canalización esté creada, debe crear una segunda CloudFormation plantilla para su canalización de Amazon S3. Si no crea la segunda plantilla, la canalización no tendrá ninguna funcionalidad de detección de cambios.

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"
            }
        }
    }
}


...
```

## Migre las canalizaciones de sondeo de una GitHub acción fuente (mediante una aplicación OAuth) a las conexiones

Puedes migrar una acción fuente GitHub (mediante una aplicación OAuth) para usar las conexiones de tu repositorio externo. Este es el método de detección de cambios recomendado para las canalizaciones con una acción fuente GitHub (a través de la aplicación OAuth).

En el caso de una canalización con una acción fuente GitHub (a través de una aplicación de OAuth), recomendamos modificar la canalización para usar una acción GitHub (a través de una GitHub aplicación) de modo que la detección de cambios se automatice completamente. AWS CodeConnections Para obtener más información sobre cómo trabajar con conexiones, consulte GitHub conexiones.

### Crea una conexión a GitHub (consola)

Puede usar la consola para crear una conexión con GitHub.

Paso 1: Sustituye tu acción GitHub (a través de la aplicación OAuth)

Usa la página de edición del pipeline para reemplazar tu acción GitHub (a través de la aplicación OAuth) por una acción GitHub (a través GitHub de la aplicación).

Para reemplazar tu acción GitHub (a través de la aplicación OAuth)

1. Inicia sesión en la CodePipeline consola.

2. Seleccione su canalización y, a continuación, elija Editar. Elija Editar etapa en la etapa de fuente. Aparece un mensaje en el que se recomienda actualizar la acción.
3. En Action provider, selecciona GitHub (a través de GitHub la aplicación).
4. Realice una de las siguientes acciones:

   - En Conexión, si aún no has creado una conexión con tu proveedor, selecciona Conectar a GitHub. Continúe con el paso 2: crear una conexión a GitHub.
   - En Conexión, si ya ha creado una conexión con su proveedor, seleccione la conexión. Continúe con el Paso 3: Guardar la acción de origen para la conexión.

### Paso 2: Crear una conexión a GitHub

Tras elegir crear la conexión, se muestra la GitHub página Conectarse a.

Para crear una conexión a GitHub

1. En la configuración de GitHub conexión, el nombre de la conexión aparece en Nombre de la conexión.

   En GitHub Aplicaciones, elige la instalación de una aplicación o elige Instalar una nueva aplicación para crear una.

   > **Note**
   >
   > Se instala una aplicación para todas las conexiones a un proveedor en particular. Si ya has instalado la GitHub aplicación, elígela y omite este paso.

2. Si GitHub aparece la página de autorización, inicie sesión con sus credenciales y, a continuación, elija continuar.
3. En la página de instalación de la aplicación, aparece un mensaje que indica que la AWS CodeStar aplicación está intentando conectarse a tu GitHub cuenta.

   > **Note**
   >
   > Solo instalas la aplicación una vez para cada GitHub cuenta. Si instaló la aplicación previamente, puede elegir Configurar para dirigirse a una página de modificación para la instalación de la aplicación o puede utilizar el botón Atrás para volver a la consola.

4. En la página Instalar AWS CodeStar, seleccione Instalar.
5. En la GitHub página Conectarse a, se muestra el identificador de conexión de la nueva instalación. Elija Conectar.

### Paso 3: Guarde la acción GitHub de origen

Complete las actualizaciones en la página Editar acción para guardar la nueva acción fuente.

Para guardar la acción GitHub de origen

1. En Repositorio, introduzca el nombre del repositorio de terceros. En Ramificación, introduzca la ramificación en la que desea que la canalización detecte los cambios de origen.

   > **Note**
   >
   > En Repositorio, escriba `owner-name/repository-name` como se muestra en este ejemplo:
   >
   > ```
   > my-account/my-repository
   > ```

2. En Formato del artefacto de salida, debe elegir el formato de los artefactos.

   - Para almacenar los artefactos de salida de la GitHub acción mediante el método predeterminado, elija CodePipeline predeterminado. La acción accede a los archivos desde el GitHub repositorio y almacena los artefactos en un archivo ZIP en el almacén de artefactos de la canalización.
   - Para almacenar un archivo JSON que contiene una referencia URL al repositorio de manera que las acciones posteriores puedan ejecutar comandos Git directamente, elija Clonación completa. Esta opción solo la pueden usar las acciones CodeBuild posteriores.

     Si eliges esta opción, tendrás que actualizar los permisos de tu función de servicio de CodeBuild proyectos, tal y como se muestra enAñade CodeBuild GitClone permisos para las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com. Para ver un tutorial que muestra cómo utilizar la opción Clonación completa, consulte Tutorial: Utilice un clon completo con una fuente de GitHub canalización.

3. En Artefactos de salida, puede conservar el nombre del artefacto de salida para esta acción, por ejemplo `SourceArtifact`. Seleccione Listo para cerrar la página Editar acción.

4. Seleccione Listo para cerrar la página de edición de etapa. Seleccione Guardar para cerrar la página de edición de la canalización.

## Cree una conexión a GitHub (CLI)

Puede usar el AWS Command Line Interface (AWS CLI) para crear una conexión con GitHub.

Para ello, utilice el comando create-connection.

> **Important**
>
> Una conexión creada a través del AWS CLI o AWS CloudFormation está en `PENDING` estado de forma predeterminada. Después de crear una conexión con la CLI o CloudFormation utilice la consola para editar la conexión y establecer su estado`AVAILABLE`.

Para crear una conexión a GitHub

1. Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows). Utilice el AWS CLI para ejecutar el create-connection comando, especificando el `--provider-type` signo y `--connection-name` para la conexión. En este ejemplo, el nombre del proveedor de terceros es `GitHub` y el nombre especificado para la conexión es `MyConnection`.

   ```
   aws codeconnections create-connection --provider-type GitHub --connection-name
    MyConnection
   ```

   Si se ejecuta correctamente, este comando devuelve la información del ARN de la conexión, que será similar a lo siguiente.

   ```
   {
       "ConnectionArn": "arn:aws:codeconnections:us-west-2:account_id:connection/
   aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
   }
   ```

2. Utilice la consola para completar la conexión.

# Migra los canales de sondeo de una acción fuente GitHub (a través de la aplicación OAuth) a webhooks

Puedes migrar tu canalización para usar webhooks y detectar cambios en tu repositorio de origen. GitHub Esta migración a webhooks es únicamente para la acción GitHub (a través de la aplicación OAuth).

- Consola: Migre los canales de sondeo a webhooks (mediante la aplicación OAuthGitHub ) (acciones fuente) (consola)
- CLI: Migre las canalizaciones de sondeo a webhooks GitHub (a través de la aplicación OAuth) (acciones fuente) (CLI)
- CloudFormation: Actualiza las canalizaciones para las acciones fuente de los eventos push GitHub (a través de la aplicación OAuth) (CloudFormation plantilla)

> **Important**
>
> Al crear CodePipeline webhooks, no utilices tus propias credenciales ni reutilices el mismo token secreto en varios webhooks. Para garantizar una seguridad óptima, genere un token secreto único para cada webhook que vaya a crear. El token secreto es una cadena arbitraria que tú proporcionas y que se GitHub utiliza para calcular y firmar las cargas útiles de los webhooks a las que se envían CodePipeline, a fin de proteger la integridad y la autenticidad de las cargas útiles de los webhooks. Usar sus propias credenciales o reutilizar el mismo token en varios webhooks puede provocar vulnerabilidades de seguridad.

## Migre los canales de sondeo a webhooks (mediante la aplicación OAuthGitHub ) (acciones fuente) (consola)

En el caso de la acción fuente GitHub (mediante la aplicación OAuth), puedes usar la CodePipeline consola para actualizar tu canalización y usar webhooks para detectar cambios en tu repositorio fuente. GitHub

Sigue estos pasos para editar una canalización que utilice sondeos (comprobaciones periódicas) para usarla en su lugar. EventBridge Si desea crear una canalización, consulte Creación de una canalización, etapas y acciones.

Al utilizar la consola, el parámetro `PollForSourceChanges` de la canalización se cambia automáticamente. El GitHub webhook se ha creado y registrado a tu nombre.

Para editar la etapa de código fuente de la canalización

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres de todas las canalizaciones asociadas a su AWS cuenta.
2. En Nombre, elija el nombre de la canalización que desea editar. Esto abre una vista detallada de la canalización, que incluye el estado de cada una de las acciones en cada etapa de la canalización.
3. En la página de detalles de la canalización, elija Edit.
4. En la etapa Edit (Editar), elija el icono de edición en la acción de código fuente.
5. Amplíe las opciones de detección de cambios y seleccione Usar Amazon CloudWatch Events para iniciar automáticamente mi canalización cuando se produzca un cambio (recomendado).

   Aparece un mensaje en el que se indica que se CodePipeline crea un webhook GitHub para detectar cambios en la fuente: AWS CodePipeline crearemos un webhook para ti. Puede renunciar en las siguientes opciones. Elija Actualizar. Además del webhook, CodePipeline crea lo siguiente:

   - Un secreto, generado aleatoriamente y utilizado para autorizar la conexión a GitHub.
   - La URL de webhook se genera utilizando el punto de enlace público correspondiente a la región.

   CodePipeline registra el webhook con GitHub. De este modo se suscribe la URL para recibir eventos del repositorio.
6. Cuando haya terminado de editar la canalización, elija Save pipeline changes para volver a la página de resumen.

   Aparece un mensaje que muestra el nombre del webhook que se creará para la canalización. Elija Guardar y continuar.
7. Para probar tu acción, publica un cambio utilizando el AWS CLI para confirmar un cambio en la fuente especificada en la fase de origen de la canalización.

## Migre las canalizaciones de sondeo a webhooks GitHub (a través de la aplicación OAuth) (acciones fuente) (CLI)

Siga estos pasos para editar una canalización que utiliza comprobaciones periódicas para utilizar un webhook en su lugar. Si desea crear una canalización, consulte Creación de una canalización, etapas y acciones.

Para crear una canalización basada en eventos, debe editar el parámetro `PollForSourceChanges` de la canalización y, a continuación, crear los siguientes recursos manualmente:

- GitHub parámetros de autorización y webhook

Para crear y registrar su webhook

> **Note**
>
> Cuando utilices la CLI o CloudFormation para crear una canalización y añadir un webhook, debes deshabilitar las comprobaciones periódicas. Para deshabilitar las comprobaciones periódicas, debe añadir de forma explícita el parámetro `PollForSourceChanges` y establecerlo en false, tal y como se detalla en el último procedimiento que viene a continuación. De lo contrario, el valor predeterminado para una CLI o una CloudFormation canalización es que el `PollForSourceChanges` valor predeterminado sea true y no aparezca en el resultado de la estructura de canalización. Para obtener más información sobre PollForSourceChanges los valores predeterminados, consulte. Configuración válida para el parámetro `PollForSourceChanges`

1. En un editor de texto, cree y guarde un archivo JSON para el webhook que desea crear. Utilice este archivo de ejemplo para un webhook denominado `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. Llame al comando put-webhook e incluya los parámetros `--cli-input` y `--region`.

   El comando de muestra siguiente crea un webhook con el archivo JSON `webhook_json`.

```
aws codepipeline put-webhook --cli-input-json file://webhook_json.json --region
 "eu-central-1"
```

3. En la salida que se muestra en este ejemplo, se devuelven la URL y el ARN de un webhook denominado `my-webhook`.

```
{
    "webhook": {
        "url": "https://webhooks.domain.com/
trigger111111111EXAMPLE11111111111111111",
        "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"
    }]
}
```

En este ejemplo, se etiqueta el webhook con la clave `Project` y el valor `ProjectA`. Para obtener más información sobre cómo etiquetar los recursos en CodePipeline, consulte. Etiquetado de recursos

4. Use el comando register-webhook-with-third-party e incluya el parámetro `--webhook-name`.

   El comando de muestra siguiente registra un webhook denominado `my-webhook`.

   ```
   aws codepipeline register-webhook-with-third-party --webhook-name my-webhook
   ```

Para editar el parámetro de tu canalización PollForSourceChanges

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

1. Ejecute el comando get-pipeline para copiar la estructura de canalización en un archivo JSON. Por ejemplo, para una canalización denominada `MyFirstPipeline`, debería escribir el siguiente comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando no devuelve nada, pero el archivo creado debería aparecer en el directorio en el que se ejecutó el comando.

2. Abra el archivo JSON en cualquier editor de texto sin formato y edite la etapa de código fuente cambiando o añadiendo el parámetro `PollForSourceChanges`. En este ejemplo, para un repositorio llamado `UserGitHubRepo`, el parámetro está establecido en `false`.

   ¿Por qué voy a hacer este cambio? Al cambiar este parámetro, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

```
"configuration": {
    "Owner": "name",
    "Repo": "UserGitHubRepo",
    "PollForSourceChanges": "false",
    "Branch": "main",
    "OAuthToken": "****"
},
```

3. Si está trabajando con la estructura de canalizaciones recuperada mediante el comando get-pipeline, debe editar la estructura en el archivo JSON quitando las líneas `metadata` del archivo. De lo contrario, el comando update-pipeline no puede utilizarlo. Quite la sección `"metadata"` de la estructura de canalizaciones del archivo JSON, incluido `{ }` y los campos `"created"`, `"pipelineARN"` y `"updated"`.

   Por ejemplo, quite las siguientes líneas de la estructura:

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

   Guarde el archivo.

4. Para aplicar los cambios, ejecute el comando update-pipeline, especificando el archivo JSON de la canalización, de forma similar a como se muestra a continuación:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

   Este comando devuelve la estructura completa de la canalización editada.

> **Note**
>
> El comando update-pipeline detiene la canalización. Si se está ejecutando una revisión en la canalización al ejecutar el comando update-pipeline, dicha ejecución se detiene. Debe iniciar manualmente la canalización para ejecutar dicha revisión en la canalización actualizada. Utilice el comando start-pipeline-execution para iniciar manualmente la canalización.

## Actualiza las canalizaciones para las acciones fuente de los eventos push GitHub (a través de la aplicación OAuth) (CloudFormation plantilla)

Sigue estos pasos para actualizar tu proceso (con una GitHub fuente), pasando de las comprobaciones periódicas (sondeos) a la detección de cambios basada en eventos mediante webhooks.

Para crear una canalización basada en eventos AWS CodeCommit, debes editar el `PollForSourceChanges` parámetro de tu canalización y, a continuación, añadir un recurso de GitHub webhooks a tu plantilla.

Si la usas CloudFormation para crear y administrar tus canalizaciones, tu plantilla tiene un contenido como el siguiente.

> **Note**
>
> Tenga en cuenta la propiedad de configuración `PollForSourceChanges` de la etapa de código fuente. Si la plantilla no incluye esa propiedad, `PollForSourceChanges` se establece en `true` de forma predeterminada.

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
                            }
                        ]
                    },


...
```

Para añadir parámetros y crear un webhook en la plantilla

Te recomendamos encarecidamente que la utilices AWS Secrets Manager para almacenar tus credenciales. Si utiliza Secrets Manager, debe haber configurado y almacenado sus parámetros secretos en Secrets Manager. En este ejemplo, se utilizan referencias dinámicas a Secrets Manager para las GitHub credenciales de tu webhook. Para obtener más información, consulte Uso de referencias dinámicas para especificar valores de plantillas.

> **Important**
>
> Al pasar parámetros del secreto, no introduzca el valor directamente en la plantilla. El valor se representa como texto no cifrado y, por lo tanto, se puede leer. Por motivos de seguridad, no utilices texto sin formato en la CloudFormation plantilla para almacenar tus credenciales.

Cuando utilices la CLI o CloudFormation para crear una canalización y añadir un webhook, debes inhabilitar las comprobaciones periódicas.

> **Note**
>
> Para deshabilitar las comprobaciones periódicas, debe añadir de forma explícita el parámetro `PollForSourceChanges` y establecerlo en false, tal y como se detalla en el último procedimiento que viene a continuación. De lo contrario, el valor predeterminado para una CLI o una CloudFormation canalización es que el `PollForSourceChanges` valor predeterminado sea true y no aparezca en el resultado de la estructura de canalización. Para obtener más información sobre PollForSourceChanges los valores predeterminados, consulte. Configuración válida para el parámetro `PollForSourceChanges`

1. En la plantilla, bajo `Resources`, añada los parámetros:

   YAML

   ```
   Parameters:
           GitHubOwner:
             Type: String

   ...
   ```

   JSON

   ```
   {
    "Parameters": {
     "BranchName": {
      "Description": "GitHub branch name",
      "Type": "String",
      "Default": "main"
   ```

```
  },
  "GitHubOwner": {
   "Type": "String"
  },

...
```

2. Usa el `AWS::CodePipeline::Webhook` CloudFormation recurso para agregar un webhook.

> **Note**
>
> El `TargetAction` especificado debe coincidir con la propiedad `Name` de su acción de código fuente definida en la canalización.

Si `RegisterWithThirdParty` está configurado en`true`, asegúrese de que el usuario asociado al `OAuthToken` pueda establecer los ámbitos requeridos. GitHub El token y el webhook requieren los siguientes alcances: GitHub

- `repo`: se utiliza para controlar totalmente la lectura y la extracción de artefactos de los repositorios públicos y privados en una canalización.
- `admin:repo_hook`: se utiliza para el control total de los enlaces del repositorio.

De lo contrario, GitHub devuelve un 404. Para obtener más información acerca del error 404 devuelto, consulte 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. Guarda la plantilla actualizada en tu ordenador local y, a continuación, abre la CloudFormation consola.

4. Seleccione la pila y luego elija Create Change Set for Current Stack (Crear conjuntos de cambios para la pila actual).
5. Cargue la plantilla y, a continuación, consulte los cambios indicados en CloudFormation. Estos son los cambios que se realizan en la pila. Debería ver los nuevos recursos en la lista.
6. Elija Ejecutar.

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

- En la plantilla, cambie `PollForSourceChanges` por `false`. Si no ha incluido `PollForSourceChanges` en la definición de la canalización, añádalo y establézcalo en false.

  ¿Por qué voy a hacer este cambio? Al cambiar este parámetro a `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

  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

Al crear estos recursos con CloudFormation, el webhook definido se crea en el GitHub repositorio especificado. Al confirmar, se activará la canalización.

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

...
```

# Crear el rol CodePipeline de servicio

Cuando cree una canalización, deberá crear un rol de servicio o utilizar uno ya existente.

Puede usar la CodePipeline consola o la AWS CLI para crear un rol CodePipeline de servicio. Es necesario un rol de servicio para crear una canalización y esta siempre estará asociada a dicho rol.

Antes de crear una canalización con la AWS CLI, debes crear un rol CodePipeline de servicio para tu canalización. Para ver un ejemplo CloudFormation de plantilla con el rol de servicio y la política especificados, consulta los tutoriales deTutorial: Cree una canalización que utilice variables de AWS CloudFormation acciones de despliegue.

La función de servicio no es una función AWS gestionada, sino que se crea inicialmente para la creación de canalizaciones y, a medida que se vayan añadiendo nuevos permisos a la política de funciones de servicio, es posible que tengas que actualizar la función de servicio de tu canalización. Una vez creada la canalización con un rol de servicio, no es posible aplicar otro rol de servicio distinto de dicha canalización. Asocie la política recomendada al rol de servicio.

Para obtener más información sobre el rol de servicio, consulte Administre la función CodePipeline de servicio.

## Crea el rol CodePipeline de servicio (consola)

Cuando usas la consola para crear una canalización, creas la función de CodePipeline servicio con el asistente de creación de canalizaciones.

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Elija Crear canalización y complete la página Paso 1: Elegir la configuración de canalización en el asistente de creación de canalizaciones.

   > **Note**
   >
   > Después de crear una canalización, no podrá cambiar el nombre. Para obtener información acerca de otras limitaciones, consulte Cuotas en AWS CodePipeline.

2. En Función de servicio, seleccione Nueva función de servicio CodePipeline para poder crear una nueva función de servicio en IAM.
3. Finalice la creación de la canalización. El rol de servicio para la canalización aparece en la lista de roles de IAM y puede ver el ARN del rol de servicio asociado a una canalización si ejecuta el comando `get-pipeline` con la CLI de AWS .

# Cree el rol CodePipeline de servicio (CLI)

Antes de crear una canalización con la AWS CLI o CloudFormation, debes crear una función de CodePipeline servicio para tu canalización y adjuntar la política de funciones de servicio y la política de confianza. Para usar la CLI para crear su rol de servicio, siga los pasos que se indican a continuación para crear primero una política de confianza JSON y la política de rol JSON como archivos independientes en el directorio donde ejecutará los comandos de la CLI.

> **Note**
>
> Recomendamos permitir solo a los administradores crear cualquier rol de servicio. Una persona con permisos para crear un rol y adjuntar cualquier política puede escalar sus propios permisos. En su lugar, cree una política que les permita crear solo los roles que necesitan o haga que un administrador cree el rol de servicio en su nombre.

1. En una ventana de terminal, introduzca el siguiente comando para crear un archivo con el nombre `TrustPolicy.json`, en el que pegará el JSON de política de roles. En este ejemplo se utiliza VIM.

   ```
   vim TrustPolicy.json
   ```

2. Pegue el siguiente JSON en el archivo.

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "codepipeline.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

Para guardar el archivo y salir de él, introduzca el siguiente comando VIM:

```
:wq
```

3. En una ventana de terminal, introduzca el siguiente comando para crear un archivo con el nombre `RolePolicy.json`, en el que pegará el JSON de política de roles. En este ejemplo se utiliza VIM.

   ```
   vim RolePolicy.json
   ```

4. Pegue la política JSON en el archivo. Utilice la política de rol de servicio con los permisos mínimos que se indica en CodePipeline política de rol de servicio. Además, agregue los permisos adecuados al rol de servicio en función de las acciones que vaya a utilizar. Para obtener una lista de acciones y un enlace a los permisos de rol de servicio necesarios para cada acción, consulte Agrega permisos al rol de CodePipeline servicio.

   Asegúrese de restringir los permisos en la medida de lo posible reduciendo el alcance al nivel de recurso en el campo `Resource`.

   Para guardar el archivo y salir de él, introduzca el siguiente comando VIM:

   ```
   :wq
   ```

5. Introduzca el siguiente comando para crear el rol y adjuntar la política del rol de confianza. Normalmente, el formato de los nombres de política es el mismo que el formato de los nombres de rol. En este ejemplo, se utilizan el nombre del rol `MyRole` y la política `TrustPolicy` que se crearon como un archivo independiente.

   ```
   aws iam create-role --role-name MyRole --assume-role-policy-document file://
   TrustPolicy.json
   ```

6. Introduzca el siguiente comando para crear la política de rol y asociarla al rol. Normalmente, el formato de los nombres de política es el mismo que el formato de los nombres de rol. En este ejemplo, se utilizan el nombre del rol `MyRole` y la política `MyRole` que se crearon como un archivo independiente.

   ```
   aws iam put-role-policy --role-name MyRole --policy-name RolePolicy --policy-
   document file://RolePolicy.json
   ```

7. Para ver el nombre del rol creado y la política de confianza, introduzca el siguiente comando para el rol denominado `MyRole`:

```
aws iam get-role --role-name MyRole
```

8. Usa el ARN del rol de servicio al crear tu canalización con la AWS CLI o CloudFormation.

## Etiquetado de recursos

Una  etiqueta  es una etiqueta de atributo personalizada que usted o usted AWS asigna a un AWS recurso. Cada AWS etiqueta tiene dos partes:

- Una clave de etiqueta (por ejemplo, `CostCenter`, `Environment`, `Project` o `Secret`). Las claves de etiqueta distinguen entre mayúsculas y minúsculas.
- Un campo opcional que se denomina valor de etiqueta (por ejemplo, `111122223333`, `Production` o el nombre de un equipo). Omitir el valor de etiqueta es lo mismo que utilizar una cadena vacía. Al igual que las claves de etiquetas, los valores de las etiquetas distinguen entre mayúsculas y minúsculas.

En conjunto, se conocen como pares clave-valor.

Las etiquetas ayudan a identificar y organizar AWS los recursos. Muchas Servicios de AWS admiten el etiquetado, por lo que puedes asignar la misma etiqueta a los recursos de diferentes servicios para indicar que los recursos están relacionados. Por ejemplo, puede asignar a una canalización la misma etiqueta que tiene un bucket de origen de Amazon S3.

Para obtener sugerencias acerca del uso de etiquetas, consulte la publicación sobre Estrategias de etiquetado de AWS en el blog de respuestas de AWS .

Puede etiquetar los siguientes tipos de recursos de CodePipeline:

- Etiquetar una canalización CodePipeline
- Etiquetar una acción personalizada en CodePipeline

Puedes usar las AWS CLI CodePipeline API o los AWS SDK para:

- Añadir etiquetas a una canalización, una acción personalizada o un webhook al crearlos.

- Añadir, administrar y quitar etiquetas de una canalización, una acción personalizada o un webhook.

También puede utilizar la consola para añadir, administrar y quitar etiquetas de una canalización.

Además de utilizar etiquetas para identificar, organizar y realizar el seguimiento de un recurso, puede utilizarlas en políticas de IAM para ayudar a controlar quién puede ver el recurso e interactuar con él. Para ver ejemplos de políticas de acceso basadas en etiquetas, consulte Uso de etiquetas para controlar el acceso a los recursos CodePipeline.

# Etiquetar una canalización CodePipeline

Las etiquetas son pares clave-valor asociados AWS a los recursos. Puede aplicar etiquetas a las canalizaciones de CodePipeline. Para obtener información sobre el etiquetado de CodePipeline recursos, los casos de uso, las restricciones de clave y valor de las etiquetas y los tipos de recursos compatibles, consulte. Etiquetado de recursos

Puede utilizar la CLI para especificar etiquetas al crear una canalización. Puede utilizar la consola o la CLI para añadir o eliminar etiquetas, y para actualizar los valores de las etiquetas de una canalización. Puede añadir hasta 50 etiquetas a cada canalización.

Temas

- Etiquetado de canalizaciones (consola)
- Etiquetado de canalizaciones (CLI)

## Etiquetado de canalizaciones (consola)

Puede utilizar la consola o la CLI para etiquetar recursos. Las canalizaciones son el único recurso de CodePipeline que se puede administrar con la consola o la CLI.

Temas

- Incorporación de etiquetas en una canalización (consola)
- Consulta de las etiquetas de una canalización (consola)
- Edición de las etiquetas de una canalización (consola)
- Eliminación de las etiquetas de una canalización (consola)

## Incorporación de etiquetas en una canalización (consola)

Puede utilizar la consola para añadir etiquetas a una canalización existente.

1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http://console.aws.amazon. com/codesuite/codepipeline/home.
2. En la página Pipelines (Canalizaciones), elija la canalización a la que desea añadir etiquetas.
3. En el panel de navegación, seleccione Configuración.
4. En Pipeline tags (Etiquetas de canalización), elija Edit (Editar).
5. En los campos Key (Clave) y Value (Valor), escriba un par de claves para cada conjunto de etiquetas que desea añadir. (El campo Valor es opcional). Por ejemplo, en Clave, escriba **`Project`**. En Valor, escriba **`ProjectA`**.
6. (Opcional) Haga clic en Añadir etiqueta para añadir más filas e introducir más etiquetas.
7. Seleccione Enviar. Las etiquetas se encuentran en la sección de configuración de la canalización.

## Consulta de las etiquetas de una canalización (consola)

Puede utilizar la consola para obtener una lista de las etiquetas de las canalizaciones existentes.

1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http://console.aws.amazon. com/codesuite/codepipeline/home.
2. En la página Pipelines (Canalizaciones), elija la canalización cuyas etiquetas desea ver.
3. En el panel de navegación, seleccione Configuración.
4. En Pipeline tags (Etiquetas de canalización), puede ver las etiquetas de la canalización en las columnas Key (Clave) y Value (Valor).

## Edición de las etiquetas de una canalización (consola)

Puede utilizar la consola para editar las etiquetas que se han añadido a las canalizaciones.

1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http://console.aws.amazon. com/codesuite/codepipeline/home.
2. En la página Pipelines (Canalizaciones), elija la canalización cuyas etiquetas desea actualizar.
3. En el panel de navegación, seleccione Configuración.

4. En Pipeline tags (Etiquetas de canalización), elija Edit (Editar).
5. En los campos Clave y Valor, actualice los valores que sean necesarios. Por ejemplo, para la clave **`Project`**, en Valor, cambie **`ProjectA`** a **`ProjectB`**.
6. Seleccione Enviar.

### Eliminación de las etiquetas de una canalización (consola)

Puede utilizar la consola para eliminar etiquetas de canalizaciones. Cuando se quitan etiquetas del recurso asociado, las etiquetas se eliminan.

1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http://console.aws.amazon. com/codesuite/codepipeline/home.
2. En la página Pipelines (Canalizaciones), elija la canalización cuyas etiquetas desea eliminar.
3. En el panel de navegación, seleccione Configuración.
4. En Pipeline tags (Etiquetas de canalización), elija Edit (Editar).
5. Junto a la clave y el valor de cada etiqueta que desea eliminar, elija Remove tag (Quitar etiqueta).
6. Seleccione Enviar.

## Etiquetado de canalizaciones (CLI)

Puede utilizar la CLI para etiquetar recursos. Debe utilizar la consola para administrar las etiquetas de las canalizaciones.

Temas

- Incorporación de etiquetas en una canalización (CLI)
- Consulta de las etiquetas de una canalización (CLI)
- Edición de las etiquetas de una canalización (CLI)
- Eliminación de las etiquetas de una canalización (CLI)

### Incorporación de etiquetas en una canalización (CLI)

Puedes usar la consola o la AWS CLI para etiquetar canalizaciones.

Para añadir una etiqueta a una canalización al crearla, consulte Creación de una canalización, etapas y acciones.

En estos pasos, se presupone que ya ha instalado una versión reciente de la AWS CLI o que la ha actualizado a la versión actual. Para obtener más información, consulte Instalación de la AWS Command Line Interface.

En el terminal o la línea de comandos, ejecute el comando tag-resource, especificando el nombre de recurso de Amazon (ARN) de la canalización a la que desea añadir etiquetas, y la clave y el valor de la etiqueta que desea añadir. Puede añadir más de una etiqueta a una canalización. Por ejemplo, para etiquetar una canalización *MyPipeline* con dos etiquetas, una clave de etiqueta *DeploymentEnvironment* con el valor de etiqueta de *Test* y una clave de etiqueta *IscontainerBased* con el valor de etiqueta de*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
```

Si se ejecuta correctamente, este comando no devuelve nada.

## Consulta de las etiquetas de una canalización (CLI)

Sigue estos pasos para usar el AWS CLI para ver las AWS etiquetas de una canalización. Si no se han añadido etiquetas, la lista obtenida está vacía.

En el terminal o la línea de comandos, ejecute el comando list-tags-for-resource. Por ejemplo, para ver una lista de claves y valores de etiquetas de una canalización denominada *MyPipeline* con el valor `arn:aws:codepipeline:`*us-west-2*`:`*account-id*`:`*MyPipeline* ARN:

```
aws codepipeline list-tags-for-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline
```

Si se ejecuta correctamente, este comando proporciona información similar a la siguiente:

```
{
    "tags": {
        "Project": "ProjectA",
        "IscontainerBased": "true"
    }
}
```

### Edición de las etiquetas de una canalización (CLI)

Sigue estos pasos para usar el AWS CLI para editar una etiqueta para una canalización. Puede cambiar el valor de una clave existente o añadir otra clave. También puede eliminar etiquetas de una canalización, tal y como se muestra en la sección siguiente.

En el terminal o la línea de comandos, ejecute el comando tag-resource, especificando el ARN de la canalización cuya etiqueta desea actualizar y especifique la clave y el valor de la etiqueta:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tags key=Project,value=ProjectA
```

Si se ejecuta correctamente, este comando no devuelve nada.

### Eliminación de las etiquetas de una canalización (CLI)

Sigue estos pasos para usar el AWS CLI para eliminar una etiqueta de una canalización. Cuando se quitan etiquetas del recurso asociado, las etiquetas se eliminan.

> **Note**
>
> Si elimina una canalización, todas las asociaciones de etiquetas se quitan de la canalización eliminada. No es necesario quitar las etiquetas antes de eliminar una canalización.

En el terminal o la línea de comandos, ejecute el comando untag-resource, especificando el ARN de la canalización cuya etiqueta desea quitar y la clave de la etiqueta que desea quitar. Por ejemplo, para eliminar varias etiquetas de una canalización *MyPipeline* con el nombre de las claves de etiqueta *Project* y*IscontainerBased*:

```
aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tag-keys Project IscontainerBased
```

Si se ejecuta correctamente, este comando no devuelve nada. Para ver las etiquetas asociadas a la canalización, ejecute el comando list-tags-for-resource.

## Creación de una regla de notificación

Puede utilizar reglas de notificación para notificar a los usuarios los cambios importantes, como cuando una canalización comienza la ejecución. Las reglas de notificación especifican tanto los

eventos como el tema de Amazon SNS que se utiliza para enviar notificaciones. Para obtener más información, consulte ¿Qué son las notificaciones?

Puede usar la consola o la AWS CLI para crear reglas de notificación para AWS CodePipeline.

Para crear una regla de notificación (consola)

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. Elija Canalizaciones, y, a continuación, elija una canalización en la que desee agregar notificaciones.
3. En la página de canalización, elija Notify (Notificar) y, a continuación, elija Create notification rule (Crear regla de notificación). También puede ir a la página Settings (Configuración) de la canalización y elegir Create notification rule (Crear regla de notificación).
4. En Nombre de la notificación, introduzca un nombre para la regla.
5. En Tipo de detalle, selecciona Básico si quieres que solo se EventBridge incluya en la notificación la información proporcionada a Amazon. Selecciona Completa si deseas incluir la información proporcionada a Amazon EventBridge y la información que pueda proporcionar el administrador de notificaciones CodePipeline o el administrador de notificaciones.

   Para obtener más información, consulte Descripción del contenido y la seguridad de las notificaciones.
6. En Eventos que activan notificaciones, seleccione los eventos para los que desea enviar notificaciones. Para obtener más información, consulte Eventos para reglas de notificación en canalizaciones.
7. En Destinos, realice una de las siguientes operaciones:

   - Si ya ha configurado un recurso para utilizarlo con notificaciones, en Elegir tipo de destino, elija Amazon Q Developer en aplicaciones de chat (Slack) o Tema de SNS. En Elegir destino, elija el nombre del cliente (para un cliente de Slack configurado en Amazon Q Developer en aplicaciones de chat) o el nombre de recurso de Amazon (ARN) del tema de Amazon SNS (para los temas de Amazon SNS ya configurados con la política necesaria para las notificaciones).
   - Si no ha configurado un recurso para utilizarlo con notificaciones, elija Crear destino y, a continuación, elija Tema de SNS. Indique el nombre del tema después de codestar-notifications- y, a continuación, elija Crear.

> **Note**
>
> - Si crea el tema de Amazon SNS durante la creación de la regla de notificación, se aplica la política que permite a la característica de notificaciones publicar eventos en el tema. El uso de un tema creado para las reglas de notificación lo ayuda a garantizar que solo suscriba a los usuarios que desea recibir notificaciones sobre este recurso.
> - No se puede crear un cliente de Amazon Q Developer en aplicaciones de chat durante la creación de una regla de notificación. Si elige Amazon Q Developer en aplicaciones de chat (Slack), aparecerá un botón que lo llevará a configurar un cliente en Amazon Q Developer en aplicaciones de chat. Elija esa opción para abrir la consola de Amazon Q Developer en aplicaciones de chat. Para obtener más información, consulte Configuración de la integración entre las notificaciones y Amazon Q Developer en aplicaciones de chat.
> - Si desea utilizar un tema de Amazon SNS ya existente como destino, debe agregar la política necesaria para las notificaciones de AWS CodeStar además de otras políticas que puedan existir para ese tema. Para obtener más información, consulte Configuración de temas de Amazon SNS para notificaciones y Descripción del contenido y la seguridad de las notificaciones.

8. Para terminar de crear la regla, elija Enviar.
9. Debe suscribir a los usuarios al tema de Amazon SNS de la regla antes de que puedan recibir notificaciones. Para obtener más información, vea Suscribir usuarios a temas de Amazon SNS que son destinos. También puede configurar la integración entre las notificaciones y Amazon Q Developer en aplicaciones de chat para enviar notificaciones a salas de chat de Amazon Chime o canales de Slack. Para obtener más información, consulte Configuración de la integración entre las notificaciones y Amazon Q Developer en aplicaciones de chat.

Para crear una regla de notificación (AWS CLI)

1. En un terminal o símbolo del sistema, ejecute el comando create-notification rule para generar el esqueleto JSON:

```
aws codestar-notifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

Puede asignar al archivo el nombre que desee. En este ejemplo, el archivo se denomina *rule.json*.

2. Abra el archivo JSON en un editor de texto sin formato y edítelo para incluir el recurso, los tipos de eventos y el destino que desea para la regla. En el siguiente ejemplo, se muestra una regla de notificación con **MyNotificationRule** el nombre de una canalización nombrada *MyDemoPipeline* en AWS una cuenta con el ID*123456789012*. Las notificaciones se envían con todos los detalles a un tema de Amazon SNS denominado *codestar-notifications-MyNotificationTopic* Cuando se inician las ejecuciones de la canalización:

```
{
    "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"
}
```

Guarde el archivo.

3. Mediante el archivo que acaba de modificar, en el terminal o línea de comandos, vuelva a ejecutar el comando create-notification-rule para crear la regla de notificación:

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. Si se ejecuta correctamente, el comando devuelve el ARN de la regla de notificación, similar a lo siguiente:

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
```

```
}
```

# Acciones de origen y métodos de detección de cambios

Cuando añade una acción de origen a su canalización, las acciones funcionan con los recursos adicionales que se describen en la tabla.

> **Note**
>
> Las acciones de origen CodeCommit y de S3 requieren un recurso de detección de cambios configurado (una EventBridge regla) o utilizar la opción de sondear el repositorio en busca de cambios en la fuente. En el caso de las canalizaciones con una acción fuente de Bitbucket o GitHub Enterprise Server, no es necesario configurar un webhook ni utilizar el sondeo de forma predeterminada. GitHub La acción de conexiones administra la detección de cambios por usted.

| Origen | ¿Utiliza recursos adicionales? | Steps | |
|---|---|---|---|
| Amazon S3 con CloudTrail recursos | Esta acción de origen utiliza una regla de eventos y CloudTrail recursos adicionales. Cuando utiliza la CLI o CloudFormation para crear esta acción, también crea y administra estos recursos. | Consulte Creación de una canalización, etapas y acciones y Conectarse a las acciones de origen de Amazon S3 que utilizan EventBridge y AWS CloudTrail. | |
| Amazon S3 sin CloudTrail recursos | Esta acción de origen utiliza un bucket habilitado para eventos con una regla de eventos sin necesidad de CloudTrail recursos adicionales. Cuando utiliza la CLI o CloudFormation | Consulte Creación de una canalización, etapas y acciones y Conexión a acciones de origen de Amazon S3 con un origen habilitado para eventos. | |

| Origen | ¿Utiliza recursos adicionales? | Steps | |
|---|---|---|---|
| | para crear esta acción, también crea y administra estos recursos. | | |
| Bitbucket Cloud | Esta acción de origen utiliza un recurso de conexión. | Consulte Conexiones de Bitbucket Cloud | |
| AWS CodeCommit | Amazon EventBridge (recomendado). Este es el valor predeterminado para las canalizaciones cuyo código fuente está en CodeCommit creadas o editadas en la consola. | Consulte Creación de una canalización, etapas y acciones y CodeCommit acciones de origen y EventBridge. | |
| Amazon ECR | Amazon EventBridge. Este valor lo crea el asistente para las canalizaciones cuyo origen de Amazon ECR se ha creado o editado en la consola. | Consulte Creación de una canalización, etapas y acciones y Acciones y recursos de origen de Amazon ECR EventBridge. | |
| GitHub o nube GitHub empresarial | Esta acción de origen utiliza un recurso de conexión. | Consulte GitHub conexiones | |
| GitHub Servidor empresarial | Esta acción de origen utiliza un recurso de conexión y un recurso de host. | Consulte GitHub Conexiones de Enterprise Server | |

| Origen | ¿Utiliza recursos adicionales? | Steps | |
|---|---|---|---|
| GitLab.com | Esta acción de origen utiliza un recurso de conexión. | Consulte GitLab.com conexiones | |
| GitLab autogestionado | Esta acción de origen utiliza un recurso de conexión y un recurso de host. | Consulte Conexiones para GitLab autogestión | |

Si tiene una canalización que utiliza sondeos, puede actualizarla para utilizar el método de detección recomendado. Para obtener más información, consulte Actualizar canalizaciones de sondeo para utilizar el método de detección de cambios recomendado.

Si desea desactivar la detección de cambios para una acción de origen que utiliza conexiones, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

# Conexión a proveedores de origen propios mediante acciones de origen

Puedes usar la AWS CodePipeline consola o la AWS CLI para conectarte a proveedores de acciones de origen, como CodeCommit S3.

> **Note**
>
> Si utiliza la consola para crear o editar una canalización, los recursos de detección de cambios se crean automáticamente. Si usa la AWS CLI para crear la canalización, debe crear los recursos adicionales usted mismo. Para obtener más información, consulte CodeCommit acciones de origen y EventBridge.

Temas


- Acciones y recursos de origen de Amazon ECR EventBridge
- Conexión a acciones de origen de Amazon S3 con un origen habilitado para eventos
- Conectarse a las acciones de origen de Amazon S3 que utilizan EventBridge y AWS CloudTrail
- CodeCommit acciones de origen y EventBridge


## Acciones y recursos de origen de Amazon ECR EventBridge

Para añadir una acción fuente de Amazon ECR CodePipeline, puede elegir entre:

- Utilice la CodePipeline consola Create Pipeline Wizard (Creación de una canalización personalizada (consola)) o la página de edición de acciones para elegir la opción de proveedor de Amazon ECR. La consola crea una EventBridge regla que inicia la canalización cuando cambia la fuente.
- Utilice la CLI para agregar la configuración de la acción `ECR` y crear recursos adicionales de la siguiente manera:
  - Utilice el ejemplo de configuración de acciones `ECR` en Referencia de acciones de origen de Amazon ECR para crear su acción, como se muestra en Crear una canalización (CLI).
  - El método de detección de cambios consiste de forma predeterminada en iniciar la canalización sondeando el origen. Debe deshabilitar las comprobaciones periódicas y crear la regla

de detección de cambios manualmente. Utilice uno de los siguientes métodos: Cree una EventBridge regla para una fuente de Amazon ECR (consola), Cree una EventBridge regla para una fuente (CLI) de Amazon ECR o Cree una EventBridge regla para una fuente de Amazon ECR (CloudFormation plantilla).

Temas


- Cree una EventBridge regla para una fuente de Amazon ECR (consola)
- Cree una EventBridge regla para una fuente (CLI) de Amazon ECR
- Cree una EventBridge regla para una fuente de Amazon ECR (CloudFormation plantilla)


## Cree una EventBridge regla para una fuente de Amazon ECR (consola)

Para crear una EventBridge regla para usarla en CodePipeline las operaciones (fuente de Amazon ECR)

1. Abra la EventBridge consola de Amazon en https://console.aws.amazon.com/events/.
2. En el panel de navegación, seleccione Events (Eventos).
3. Elija Crear regla y, a continuación, en Origen de evento, en Nombre del servicio, elija Elastic Container Registry (ECR).
4. En Origen de evento, elija Patrón de eventos.

   Elija Editar y, a continuación, pegue el siguiente patrón de eventos de ejemplo en la ventana Origen de evento para un repositorio `eb-test` con una etiqueta de imagen de `cli-testing`:

```
{
    "detail-type": [
        "ECR Image Action"
    ],
    "source": [
        "aws.ecr"
    ],
    "detail": {
        "action-type": [
            "PUSH"
        ],
        "image-tag": [
            "latest"
        ],
```

```
        "repository-name": [
            "eb-test"
        ],
        "result": [
            "SUCCESS"
        ]
    }
}
```

> **Note**
>
> Para ver el patrón de eventos completo compatible con los eventos de Amazon ECR, consulte Amazon ECR Events EventBridge o Amazon Elastic Container Registry Events.

5. Seleccione Save.

   En el panel Event Pattern Preview, visualice la regla.

6. En Targets, elija. CodePipeline
7. Introduzca el ARN de la canalización que iniciará esta regla.

> **Note**
>
> Puede encontrar el ARN de la canalización en la salida de metadatos después de ejecutar el comando get-pipeline. El ARN de canalización se crea con el siguiente formato:
> arn:aws:codepipeline::: *region account pipeline-name*
> ARN de canalización de muestra:
> `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

8. Crea o especifica un rol de servicio de IAM que conceda EventBridge permisos para invocar el objetivo asociado a tu EventBridge regla (en este caso, el objetivo es). CodePipeline

   - Elige Crear un rol nuevo para este recurso específico a fin de crear un rol de servicio que otorgue EventBridge permisos para iniciar las ejecuciones de tu canalización.
   - Elige Usar el rol existente para ingresar un rol de servicio que otorgue EventBridge permisos para iniciar las ejecuciones de tu canalización.

9. (Opcional) Para especificar las anulaciones de la fuente con un ID de imagen específico, usa el transformador de entrada para pasar los datos como parámetros de JSON. También puede usar el transformador de entrada para pasar variables de canalización.

   - Amplíe Configuración adicional.

     En Configurar entrada de destino, elija Configurar transformador de entrada.

     En la ventana de cuadro de diálogo, elija Escribir el mío. En el cuadro Ruta de entrada, escriba los siguientes pares clave-valor.

     ```
     {"revisionValue": "$.detail.image-digest"}
     ```

   - En el cuadro Plantilla, escriba los siguientes pares clave-valor.

     ```
     {
         "sourceRevisions": [
             {
                 "actionName": "Source",
                 "revisionType": "IMAGE_DIGEST",
                 "revisionValue": "<revisionValue>"
             }
         ],
          "variables": [
             {
                 "name": "Variable_Name",
                 "value": "Variable_Value"
             }
         ]
     }
     ```

   - Elija Confirmar.

10. Revise la configuración de la regla para asegurarse de que se ajusta a sus necesidades.
11. Seleccione Configurar los detalles.
12. En la página Configure rule details (Configurar detalles de regla), escriba un nombre y una descripción para la regla y, a continuación, elija State (Estado) para habilitarla.
13. Si está satisfecho con la regla, elija Create rule (Crear regla).

# Cree una EventBridge regla para una fuente (CLI) de Amazon ECR

Ejecute el comando put-rule especificando lo siguiente:

- Un nombre que identifique de forma inequívoca la regla que está creando. Este nombre debe ser único en todas las canalizaciones que cree y que estén CodePipeline asociadas a su AWS cuenta.
- El patrón de eventos para el origen y los campos de detalles utilizados por la regla. Para obtener más información, consulta Amazon EventBridge y Event Patterns.

Para crear una EventBridge regla con Amazon ECR como origen y destino CodePipeline del evento

1. Añada los permisos EventBridge para utilizarlos CodePipeline para invocar la regla. Para obtener más información, consulte Uso de políticas basadas en recursos para Amazon. EventBridge

   a. Utilice el siguiente ejemplo para crear la política de confianza que permite que EventBridge asuma el rol de servicio. Ponga un nombre a la política de confianza `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Utilice el comando para crear el rol `Role-for-MyRule` y asocie la política de confianza.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

c. Cree el JSON de la política de permisos, tal y como se muestra en este ejemplo para la canalización denominada `MyFirstPipeline`. Ponga un nombre a la política de permisos `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilice el siguiente comando para asociar la política de permisos `CodePipeline-Permissions-Policy-for-EB` al rol `Role-for-MyRule`.

¿Por qué voy a hacer este cambio? Al añadir esta política al rol, se crean permisos para. EventBridge

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Llame al comando put-rule e incluya los parámetros `--name`, `--event-pattern` y `--role-arn`.

¿Por qué voy a hacer este cambio? Debe crear un evento con una regla que especifique cómo se debe hacer una inserción de imagen, y un objetivo que indique el nombre de la canalización que va a iniciar el evento.

El siguiente comando de ejemplo crea una regla llamada `MyECRRepoRule`.

```
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**
>
> Para ver el patrón de eventos completo compatible con los eventos de Amazon ECR, consulte Amazon ECR Events EventBridge o Amazon Elastic Container Registry Events.

3. Para añadirlo CodePipeline como destino, put-targets ejecute el comando e incluya los siguientes parámetros:

   - El parámetro `--rule` se usa con el `rule_name` que creó con el comando put-rule.
   - El parámetro `--targets` se usa con el `Id` del destino de la lista de destinos y el ARN de la canalización de destino.

   El siguiente comando de muestra especifica que, para la regla denominada `MyECRRepoRule`, el destino `Id` se compone del número uno, lo que indica que, en lo que puede ser una lista de destinos de la regla, se trata del destino 1. El comando de ejemplo también especifica un `Arn` de ejemplo para la canalización y el `RoleArn` de ejemplo para la regla. La canalización se inicia cuando se produce algún cambio en el repositorio.

   ```
   aws events put-targets --rule MyECRRepoRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-
   west-2:80398EXAMPLE:TestPipeline,RoleArn=arn:aws:iam::80398EXAMPLE:role/Role-for-
   MyRule
   ```

4. (Opcional) Para configurar un transformador de entrada con anulaciones de origen para un ID de imagen específico, utilice el siguiente fragmento de código JSON en el comando de la CLI. En el siguiente ejemplo, se configura una anulación donde:

   - `actionName` (`Source` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - `revisionType` (`IMAGE_DIGEST` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.

- En este ejemplo`revisionValue`, < *revisionValue* > se deriva de la variable de evento de origen.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "ARN",
            "InputTransformer": {
                "InputPathsMap": {
                    "revisionValue": "$.detail.image-digest"
                },
                "InputTemplate": {
                    "sourceRevisions": [
                        {
                            "actionName": "Source",
                            "revisionType": "IMAGE_DIGEST",
                            "revisionValue": "<revisionValue>"
                        }
                    ]
                }
            }
        }
    ]
}
```

## Cree una EventBridge regla para una fuente de Amazon ECR (CloudFormation plantilla)

CloudFormation Para crear una regla, utilice el fragmento de plantilla tal y como se muestra aquí.

Para actualizar tu canalización CloudFormation crear una plantilla y crear una EventBridge regla

1. En la sección inferior de la plantilla`Resources`, usa el `AWS::IAM::Role` CloudFormation recurso para configurar la función de IAM que permitirá que tu evento inicie su proceso. Esta entrada crea un rol que utiliza dos políticas:

   - La primera política permite asumir el rol.

- La segunda política concede permisos para iniciar la canalización.

¿Por qué voy a hacer este cambio? Debes crear un rol que pueda asumir EventBridge para iniciar una ejecución en nuestro proceso.

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. En la sección de la plantilla`Resources`, utilice el `AWS::Events::Rule` CloudFormation recurso para añadir una EventBridge regla para la fuente de Amazon ECR. Este patrón de eventos crea un evento que monitoriza las confirmaciones en su repositorio. Cuando EventBridge detecta un cambio en el estado del repositorio, la regla se invoca `StartPipelineExecution` en la canalización de destino.

   ¿Por qué voy a hacer este cambio? Debe crear un evento con una regla que especifique cómo se debe hacer una inserción de imagen, y un objetivo que indique el nombre de la canalización que va a iniciar el evento.

Este fragmento utiliza una imagen denominada `eb-test` con una etiqueta `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**
>
> Para ver el patrón de eventos completo compatible con los eventos de Amazon ECR, consulte Amazon ECR Events EventBridge o Amazon Elastic Container Registry Events.

3. (Opcional) Para configurar un transformador de entrada con anulaciones de origen para un ID de imagen específico, utilice el siguiente fragmento de código YAML. En el siguiente ejemplo, se configura una anulación donde:

   - `actionName` (`Source` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.

- `revisionType` (`IMAGE_DIGEST` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
- En este ejemplo`revisionValue`, **<** *`revisionValue`* **>** se deriva de la variable de evento de origen.

```
---
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: ARN
  InputTransformer:
    InputPathsMap:
      revisionValue: "$.detail.image-digest"
    InputTemplate:
      sourceRevisions:
        actionName: Source
        revisionType: IMAGE_DIGEST
        revisionValue: '<revisionValue>'
```

4. Guarde la plantilla actualizada en el equipo local y luego abra la consola de CloudFormation .
5. Seleccione la pila y luego elija Create Change Set for Current Stack (Crear conjuntos de cambios para la pila actual).
6. Cargue la plantilla y, a continuación, consulte los cambios indicados en CloudFormation. Estos son los cambios que se realizan en la pila. Debería ver los nuevos recursos en la lista.
7. Elija Ejecutar.

## Conexión a acciones de origen de Amazon S3 con un origen habilitado para eventos

Las instrucciones de esta sección proporcionan los pasos para crear la acción fuente de S3, sin necesidad de crear o administrar AWS CloudTrail recursos.

> **Important**
>
> El procedimiento para crear esta acción sin AWS CloudTrail recursos no está disponible en la consola. Para usar la CLI, consulte aquí los procedimientos o visite Migración de canalizaciones de sondeo con un origen de S3 habilitado para los eventos.

En el caso de una canalización con una fuente de Amazon S3, modifique la canalización para que la detección de cambios se automatice mediante EventBridge un bucket de origen que esté habilitado para las notificaciones de eventos. Este es el método recomendado si utiliza la CLI o CloudFormation para migrar su canalización.

> **Note**
>
> Esto incluye el uso de un bucket que esté habilitado para las notificaciones de eventos, en el que no es necesario crear una CloudTrail ruta independiente. Si utilizas la consola, tendrás configuradas una regla CloudTrail y un registro del evento. Para los siguientes pasos, consulte Migre las canalizaciones de sondeo con una fuente y CloudTrail un registro de S3.

- CLI: Migre las canalizaciones de sondeo con una fuente y un CloudTrail registro (CLI) de S3
- CloudFormation: Migre las canalizaciones de sondeo con una fuente y un CloudTrail registro de S3 (CloudFormation plantilla)

## Creación de canalizaciones con un origen de S3 habilitado para los eventos (CLI)

Siga estos pasos para crear una canalización con una fuente de S3 que utilice una entrada de eventos EventBridge para detectar cambios. Si desea conocer los pasos completos para crear una canalización con la CLI, consulte Creación de una canalización, etapas y acciones.

Para crear una canalización basada en eventos con Amazon S3, debe editar el parámetro `PollForSourceChanges` de la canalización y, a continuación, crear los siguientes recursos manualmente:

- EventBridge regla de evento
- Función de IAM para permitir que el EventBridge evento inicie su proceso

Para crear una EventBridge regla con Amazon S3 como origen y CodePipeline destino del evento y aplicar la política de permisos

1. Otorgue los permisos EventBridge para usarlos CodePipeline para invocar la regla. Para obtener más información, consulte Uso de políticas basadas en recursos para Amazon. EventBridge

   a. Utilice el siguiente ejemplo para crear la política de confianza que permite que EventBridge asuma el rol de servicio. Denomínelo `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Utilice el comando para crear el rol `Role-for-MyRule` y asocie la política de confianza.

      ¿Por qué voy a hacer este cambio? Al añadir esta política de confianza al rol, se crean permisos para EventBridge.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Cree el JSON de la política de permisos tal y como se muestra aquí para la canalización denominada `MyFirstPipeline`. Ponga un nombre a la política de permisos `permissionspolicyforEB.json`.

      JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilice el siguiente comando para asociar la nueva política de permisos `CodePipeline-Permissions-Policy-for-EB` al rol `Role-for-MyRule` que ha creado.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Llame al comando put-rule e incluya los parámetros `--name`, `--event-pattern` y `--role-arn`.

   El siguiente comando de ejemplo crea una regla denominada `EnabledS3SourceRule`.

```
aws events put-rule --name "EnabledS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"Object Created\"],\"detail\":{\"bucket\":{\"name\":
[\"amzn-s3-demo-source-bucket\"]}}}" --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-
for-MyRule"
```

3. Para añadirlo CodePipeline como objetivo, ejecuta el put-targets comando e incluye los parámetros `--rule` y`--targets`.

   El siguiente comando especifica que, para la regla denominada `EnabledS3SourceRule`, el destino `Id` se compone del número uno, lo que indica que, en lo que puede ser una lista de destinos de la regla, se trata del destino 1. El comando también especifica un ARN de ejemplo para la canalización. La canalización se inicia cuando se produce algún cambio en el repositorio.

```
aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-
AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

1. Ejecute el comando get-pipeline para copiar la estructura de canalización en un archivo JSON. Por ejemplo, para una canalización denominada `MyFirstPipeline`, escriba el siguiente comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando no devuelve nada, pero el archivo creado debería aparecer en el directorio en el que se ejecutó el comando.

2. Abra el archivo JSON en cualquier editor de texto sin formato y edite la etapa de código fuente cambiando el parámetro `PollForSourceChanges` del bucket denominado `amzn-s3-demo-source-bucket` a `false`, tal y como se muestra en este ejemplo.

   ¿Por qué voy a hacer este cambio? Al establecer este parámetro en `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Si está trabajando con la estructura de la canalización recuperada mediante el comando get-pipeline, debe eliminar las líneas `metadata` del archivo JSON. De lo contrario, el comando update-pipeline no puede utilizarlo. Elimine las líneas `"metadata": { }` y los campos `"updated"`, `"created"` y `"pipelineARN"`.

Por ejemplo, quite las siguientes líneas de la estructura:

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

Guarde el archivo.

4. Para aplicar los cambios, ejecute el comando update-pipeline especificando el archivo JSON:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

Este comando devuelve la estructura completa de la canalización editada.

> **Note**
>
> El comando update-pipeline detiene la canalización. Si se está ejecutando una revisión en la canalización al ejecutar el comando update-pipeline, dicha ejecución se detiene. Debe iniciar manualmente la canalización para ejecutar dicha revisión en la canalización actualizada. Utilice el comando start-pipeline-execution para iniciar manualmente la canalización.

## Crea canalizaciones con una fuente de S3 habilitada para eventos (CloudFormation plantilla)

Este procedimiento es para una canalización en la que el bucket de origen tiene los eventos habilitados.

Utilice estos pasos para crear una canalización con un origen de Amazon S3 para la detección de cambios basada en eventos.

Para crear una canalización basada en eventos con Amazon S3, deberá editar el parámetro `PollForSourceChanges` de la canalización y, a continuación, añadir los siguientes recursos a su plantilla:

- EventBridge regla y rol de IAM para permitir que este evento inicie tu canalización.

Si la usas CloudFormation para crear y administrar tus canalizaciones, tu plantilla incluye contenido como el siguiente.

> **Note**
>
> La propiedad `Configuration` en la etapa de código fuente denominada `PollForSourceChanges`. Si la plantilla no incluye esa propiedad, `PollForSourceChanges` se establece en `true` de forma predeterminada.

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
                            }
```

```
                    ]
                },


...
```

Para crear una EventBridge regla con Amazon S3 como origen y CodePipeline destino del evento y aplicar la política de permisos

1. En la sección de abajo de la plantilla`Resources`, utilice el `AWS::IAM::Role` CloudFormation recurso para configurar el rol de IAM que permite que su evento inicie su canalización. Esta entrada crea un rol que utiliza dos políticas:

   - La primera política permite asumir el rol.
   - La segunda política concede permisos para iniciar la canalización.

   ¿Por qué voy a hacer este cambio? Añadir `AWS::IAM::Role` un recurso permite CloudFormation crear permisos para EventBridge. Este recurso se agrega a su CloudFormation pila.

   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. Usa el `AWS::Events::Rule` CloudFormation recurso para agregar una EventBridge regla. Este patrón de eventos crea un evento que monitoriza la creación o eliminación de objetos en el bucket de origen de Amazon S3. Además, incluya un objetivo de la canalización. Cuando se crea un objeto, esta regla invoca `StartPipelineExecution` en la canalización de destino.

   ¿Por qué voy a hacer este cambio? Agregar el `AWS::Events::Rule` recurso permite CloudFormation crear el evento. Este recurso se añade a tu CloudFormation pila.

   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. Guarde la plantilla actualizada en el equipo local y abra la consola de CloudFormation .
4. Seleccione la pila y luego elija Create Change Set for Current Stack (Crear conjuntos de cambios para la pila actual).
5. Cargue su plantilla actualizada y, a continuación, consulte los cambios indicados en CloudFormation. Estos son los cambios que se realizarán en la pila. Debería ver los nuevos recursos en la lista.
6. Elija Ejecutar.

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al

añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

- En la plantilla, cambie `PollForSourceChanges` por `false`. Si no ha incluido `PollForSourceChanges` en la definición de la canalización, añádalo y establézcalo en `false`.

  ¿Por qué voy a hacer este cambio? Al cambiar `PollForSourceChanges` a `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

  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

Cuando los usas CloudFormation para crear estos recursos, tu canalización se activa cuando se crean o actualizan los archivos de tu repositorio.

> **Note**
>
> No se detenga aquí. Aunque su canalización esté creada, debe crear una segunda CloudFormation plantilla para su canalización de Amazon S3. Si no crea la segunda plantilla, la canalización no tendrá ninguna funcionalidad de detección de cambios.

YAML

```
Parameters:
  SourceObjectKey:
    Description: 'S3 source artifact'
    Type: String
    Default: SampleApp_Linux.zip
  ApplicationName:
    Description: 'CodeDeploy application name'
    Type: String
```

```
    Default: DemoApplication
  BetaFleet:
    Description: 'Fleet configured in CodeDeploy'
    Type: String
    Default: DemoFleet

Resources:
  SourceBucket:
    Type: AWS::S3::Bucket
    Properties:
      NotificationConfiguration:
        EventBridgeConfiguration:
          EventBridgeEnabled: true
      VersioningConfiguration:
        Status: Enabled
  CodePipelineArtifactStoreBucket:
    Type: AWS::S3::Bucket
  CodePipelineArtifactStoreBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref CodePipelineArtifactStoreBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Sid: DenyUnEncryptedObjectUploads
            Effect: Deny
            Principal: '*'
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt CodePipelineArtifactStoreBucket.Arn, '/
*' ] ]
            Condition:
              StringNotEquals:
                s3:x-amz-server-side-encryption: aws:kms
          -
            Sid: DenyInsecureConnections
            Effect: Deny
            Principal: '*'
            Action: s3:*
            Resource: !Sub ${CodePipelineArtifactStoreBucket.Arn}/*
            Condition:
              Bool:
                aws:SecureTransport: false
  CodePipelineServiceRole:
```

```
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - codepipeline.amazonaws.com
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: AWS-CodePipeline-Service-3
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action:
                  - codecommit:CancelUploadArchive
                  - codecommit:GetBranch
                  - codecommit:GetCommit
                  - codecommit:GetUploadArchiveStatus
                  - codecommit:UploadArchive
                Resource: 'resource_ARN'
              -
                Effect: Allow
                Action:
                  - codedeploy:CreateDeployment
                  - codedeploy:GetApplicationRevision
                  - codedeploy:GetDeployment
                  - codedeploy:GetDeploymentConfig
                  - codedeploy:RegisterApplicationRevision
                Resource: 'resource_ARN'
              -
                Effect: Allow
                Action:
                  - codebuild:BatchGetBuilds
                  - codebuild:StartBuild
                Resource: 'resource_ARN'
              -
                Effect: Allow
```

```
                Action:
                  - devicefarm:ListProjects
                  - devicefarm:ListDevicePools
                  - devicefarm:GetRun
                  - devicefarm:GetUpload
                  - devicefarm:CreateUpload
                  - devicefarm:ScheduleRun
                Resource: 'resource_ARN'
              -
                Effect: Allow
                Action:
                  - lambda:InvokeFunction
                  - lambda:ListFunctions
                Resource: 'resource_ARN'
              -
                Effect: Allow
                Action:
                  - iam:PassRole
                Resource: 'resource_ARN'
              -
                Effect: Allow
                Action:
                  - elasticbeanstalk:*
                  - ec2:*
                  - elasticloadbalancing:*
                  - autoscaling:*
                  - cloudwatch:*
                  - s3:*
                  - sns:*
                  - cloudformation:*
                  - rds:*
                  - sqs:*
                  - ecs:*
                Resource: 'resource_ARN'
  AppPipeline:
    Type: AWS::CodePipeline::Pipeline
    Properties:
      Name: s3-events-pipeline
      RoleArn:
        !GetAtt CodePipelineServiceRole.Arn
      Stages:
        -
          Name: Source
          Actions:
```

```
            -
              Name: SourceAction
              ActionTypeId:
                Category: Source
                Owner: AWS
                Version: 1
                Provider: S3
              OutputArtifacts:
                - Name: SourceOutput
              Configuration:
                S3Bucket: !Ref SourceBucket
                S3ObjectKey: !Ref SourceObjectKey
                PollForSourceChanges: false
              RunOrder: 1
        -
          Name: Beta
          Actions:
            -
              Name: BetaAction
              InputArtifacts:
                - Name: SourceOutput
              ActionTypeId:
                Category: Deploy
                Owner: AWS
                Version: 1
                Provider: CodeDeploy
              Configuration:
                ApplicationName: !Ref ApplicationName
                DeploymentGroupName: !Ref BetaFleet
              RunOrder: 1
      ArtifactStore:
        Type: S3
        Location: !Ref CodePipelineArtifactStoreBucket
  EventRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
```

```
            Action: sts:AssumeRole
      Path: /
      Policies:
        -
          PolicyName: eb-pipeline-execution
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              -
                Effect: Allow
                Action: codepipeline:StartPipelineExecution
                Resource: !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region',
':', !Ref 'AWS::AccountId', ':', !Ref AppPipeline ] ]
 EventRule:
   Type: AWS::Events::Rule
   Properties:
     EventBusName: default
     EventPattern:
       source:
         - aws.s3
       detail-type:
         - Object Created
       detail:
         bucket:
           name:
             - !Ref SourceBucket
     Name: EnabledS3SourceRule
     State: ENABLED
     Targets:
       -
         Arn:
           !Join [ '', [ 'arn:aws:codepipeline:', !Ref 'AWS::Region', ':', !Ref
'AWS::AccountId', ':', !Ref AppPipeline ] ]
         RoleArn: !GetAtt EventRole.Arn
         Id: codepipeline-AppPipeline
```

JSON

JSON

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

```
    "Statement": [
        {
            "Action": [
                "appconfig:StartDeployment",
                "appconfig:StopDeployment",
                "appconfig:GetDeployment"
            ],
            "Resource": [
                "arn:aws:appconfig:*:111122223333:application/
[[Application]]",
                "arn:aws:appconfig:*:111122223333:application/[[Application]]/
*",
                "arn:aws:appconfig:*:111122223333:deploymentstrategy/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

# Conectarse a las acciones de origen de Amazon S3 que utilizan EventBridge y AWS CloudTrail

Las instrucciones de esta sección proporcionan los pasos para crear la acción fuente de S3 que utilice AWS CloudTrail los recursos que debe crear y administrar. Para usar la acción fuente de S3 de forma EventBridge que no requiera AWS CloudTrail recursos adicionales, siga las instrucciones de la CLI que se encuentran enMigración de canalizaciones de sondeo con un origen de S3 habilitado para los eventos.

> **Important**
>
> Este procedimiento proporciona los pasos para crear la acción de origen de S3 que utiliza recursos de AWS CloudTrail que el usuario debe crear y administrar. El procedimiento para crear esta acción sin AWS CloudTrail recursos no está disponible en la consola. Para utilizar la CLI, consulte Migración de canalizaciones de sondeo con un origen de S3 habilitado para los eventos.

Para añadir una acción de origen de Amazon S3 CodePipeline, puede elegir entre:

- Utilice la CodePipeline consola Create Pipeline Wizard (Creación de una canalización personalizada (consola)) o la página de edición de acciones para elegir la opción de proveedor de S3. La consola crea una EventBridge regla y un CloudTrail registro que inician tu canalización cuando cambia la fuente.
- Utilice la configuración AWS CLI para añadir la S3 acción y crear recursos adicionales de la siguiente manera:
  - Utilice el ejemplo de configuración de acciones S3 en Referencia sobre la acción de origen de Amazon S3 para crear su acción, como se muestra en Crear una canalización (CLI).
  - El método de detección de cambios consiste de forma predeterminada en iniciar la canalización sondeando el origen. Debe deshabilitar las comprobaciones periódicas y crear la regla de detección de cambios manualmente. Utilice uno de los siguientes métodos: Cree una EventBridge regla para una fuente de Amazon S3 (consola), Cree una EventBridge regla para una fuente de Amazon S3 (CLI) o Cree una EventBridge regla para una fuente de Amazon S3 (CloudFormation plantilla).

AWS CloudTrail es un servicio que registra y filtra los eventos en su bucket de origen de Amazon S3. La ruta envía los cambios de la fuente filtrada a la EventBridge regla. La EventBridge regla detecta el cambio de fuente y, a continuación, inicia la canalización.

Requisitos:

- Si no va a crear una ruta, utilice una AWS CloudTrail ruta existente para registrar los eventos en su bucket fuente de Amazon S3 y enviar los eventos filtrados a la EventBridge regla.
- Cree o utilice un bucket de S3 existente en el que AWS CloudTrail pueda almacenar sus archivos de registro. AWS CloudTrail debe tener los permisos necesarios para entregar los archivos de registro a un bucket de Amazon S3. El bucket no se puede configurar como un bucket de pago por solicitante. Cuando crea un bucket de Amazon S3 como parte de la creación o actualización de un registro en la consola, AWS CloudTrail adjunta automáticamente los permisos necesarios al bucket. Para obtener más información, consulte la política de buckets de Amazon S3 para CloudTrail.

## Cree una EventBridge regla para una fuente de Amazon S3 (consola)

Antes de configurar una regla EventBridge, debe crear una AWS CloudTrail ruta. Para obtener más información, consulte Creación de un registro de seguimiento en la consola.

> **Important**
>
> Si utilizas la consola para crear o editar tu canalización, la EventBridge regla y la AWS CloudTrail ruta se crearán automáticamente.

Creación de un registro de seguimiento

1. Abre la AWS CloudTrail consola.
2. En el panel de navegación, seleccione Trails.
3. Elija Create Trail (Crear registro de seguimiento). En Trail name (Nombre de registro de seguimiento), escriba un nombre para el registro de seguimiento.
4. En Storage location (Ubicación de almacenamiento), cree o especifique el bucket que se utilizará para almacenar los archivos de log. De forma predeterminada, los buckets y los objetos de Amazon S3 son privados. Solo el propietario del recurso (la AWS cuenta que creó el bucket) puede acceder al bucket y a sus objetos. El depósito debe tener una política de recursos que AWS CloudTrail permita el acceso a los objetos del depósito.
5. En Bucket y carpeta de registro de seguimiento, especifique un bucket de Amazon S3 y el prefijo del objeto (nombre de la carpeta) para registrar los eventos de datos de todos los objetos de la carpeta. Puede agregar hasta 250 objetos de Amazon S3 a cada registro de seguimiento. Complete la información de la clave de cifrado requerida y seleccione Siguiente.
6. En Tipo de evento, elija Eventos de administración.
7. En Eventos de administración, elija Escribir. El registro de seguimiento consigna la actividad de la API para cada objeto de Amazon S3 (por ejemplo, `GetObject` y `PutObject`) con el bucket y el prefijo especificados.
8. Elija Write (Escribir).
9. Si está satisfecho con el registro de seguimiento, elija Crear registro de seguimiento.

Para crear una EventBridge regla que apunte a tu canalización con una fuente de Amazon S3

1. Abra la EventBridge consola de Amazon en https://console.aws.amazon.com/events/.
2. En el panel de navegación, seleccione Reglas. Deje el bus predeterminado seleccionado o elija un bus de eventos. Elija Crear regla.
3. En Nombre, introduzca un nombre para la regla.
4. En Tipo de regla, elija Regla con un patrón de evento. Elija Siguiente.

5. En Fuente del evento, elige AWS eventos o eventos de EventBridge socios.
6. En Ejemplo de tipo de evento, seleccione Eventos de AWS .
7. En Ejemplos de eventos, escriba S3 como palabra clave para filtrar. Elige AWS API call via CloudTrail.
8. En Método de creación, elija Patrón de cliente (JSON editor).

   Pegue el patrón de eventos que se muestra a continuación. Asegúrese de agregar el nombre del bucket y la clave de objeto S3 (o el nombre de clave) que identifica de forma única al objeto en el bucket como `requestParameters`. En este ejemplo, se crea una regla para un bucket denominado `amzn-s3-demo-source-bucket` y una clave de objeto de `my-files.zip`. Cuando se utiliza la ventana Edit (Editar) para especificar los recursos, la regla se actualiza para utilizar un patrón de eventos personalizado.

   A continuación se muestra un patrón de eventos de muestra para copiar y pegar:

```
{
    "source": [
        "aws.s3"
    ],
    "detail-type": [
        "AWS API Call via CloudTrail"
    ],
    "detail": {
        "eventSource": [
            "s3.amazonaws.com"
        ],
        "eventName": [
            "CopyObject",
            "CompleteMultipartUpload",
            "PutObject"
        ],
        "requestParameters": {
            "bucketName": [
                "amzn-s3-demo-source-bucket"
            ],
            "key": [
                "my-files.zip"
            ]
        }
    }
```

```
}
```

9. Elija Siguiente.
10. En Tipos de destino, elija Servicio de AWS .
11. En  Seleccione un objetivo, elija  CodePipeline. En ARN de canalización, Introduzca el ARN de la canalización que se iniciará cuando la active esta regla.

> **Note**
>
> Para obtener el ARN de la canalización, ejecute el comando get-pipeline. El ARN de la canalización aparece en la salida. Se crea con el siguiente formato:
> arn:aws:codepipeline::: *region account pipeline-name*
> ARN de canalización de muestra:
> arn:aws:codepipeline: us-east - 2:80398 EXAMPLE:MyFirstPipeline

12. Para crear o especificar un rol de servicio de IAM que conceda EventBridge permisos para invocar el objetivo asociado a tu EventBridge regla (en este caso, el objetivo es CodePipeline):
    - Elige  Crear un rol nuevo para este recurso específico  a fin de crear un rol de servicio que otorgue EventBridge permisos para iniciar las ejecuciones de tu canalización.
    - Elige  Usar el rol existente  para ingresar un rol de servicio que otorgue EventBridge permisos para iniciar las ejecuciones de tu canalización.
13. (Opcional) Para especificar las anulaciones de la fuente con un ID de imagen específico, usa el transformador de entrada para pasar los datos como parámetros de JSON. También puede usar el transformador de entrada para pasar variables de canalización.
    - Amplíe Configuración adicional.

      En Configurar entrada de destino, elija Configurar transformador de entrada.

      En la ventana de cuadro de diálogo, elija Escribir el mío. En el cuadro Ruta de entrada, escriba los siguientes pares clave-valor.

      ```
      {"revisionValue": "$.detail.object.version-id"}
      ```

    - En el cuadro Plantilla, escriba los siguientes pares clave-valor.

      ```
      
      ```

```
                                {
    "sourceRevisions": [
        {
            "actionName": "Source",
            "revisionType": "S3_OBJECT_VERSION_ID",
            "revisionValue": "<revisionValue>"
        }
    ],
     "variables": [
        {
            "name": "Variable_Name",
            "value": "Variable_Value"
        }
    ]
}
```

- Seleccione Confirmar.

14. Elija Siguiente.
15. En la página Etiquetas, elija Siguiente:
16. En la página Revisar y crear, revise la configuración de la regla. Si está satisfecho con la regla, elija Create rule (Crear regla).

## Cree una EventBridge regla para una fuente de Amazon S3 (CLI)

Para crear un AWS CloudTrail rastrear y habilitar el registro

Para usar el AWS CLI para crear una ruta, ejecute el create-trail comando y especifique:

- El nombre del registro de seguimiento.
- El bucket al que ya haya aplicado la política de bucket para AWS CloudTrail.

Para obtener más información, consulte Crear una ruta con la interfaz de línea de AWS comandos.

1. Llame al comando create-trail e incluya los parámetros `--name` y `--s3-bucket-name`.

   ¿Por qué voy a hacer este cambio? Esto crea el registro de seguimiento de CloudTrail necesario para su bucket de origen de S3.

El comando siguiente utiliza `--name` y `--s3-bucket-name` para crear un registro de seguimiento llamado `my-trail` y un bucket llamado `amzn-s3-demo-source-bucket`.

```
aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-source-
bucket
```

2. Use el comando start-logging e incluya el parámetro `--name`.

   ¿Por qué voy a hacer este cambio? Este comando inicia el CloudTrail registro del bucket de origen y envía los eventos a EventBridge.

   Ejemplo:

   El siguiente comando utiliza `--name` para iniciar el registro en un registro de seguimiento llamado `my-trail`.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Llame al comando put-event-selectors e incluya los parámetros `--trail-name` y `--event-selectors`. Usa los selectores de eventos para especificar que quieres que tu ruta registre los eventos de datos del bucket de origen y envíe los eventos a la EventBridge regla.

   ¿Por qué voy a hacer este cambio? Este comando filtra eventos.

   Ejemplo:

   El siguiente comando utiliza `--trail-name` y `--event-selectors` para especificar eventos de datos para un bucket de origen y un prefijo llamados `amzn-s3-demo-source-bucket/myFolder`.

   ```
   aws cloudtrail put-event-selectors --trail-name my-trail --event-selectors
    '[{ "ReadWriteType": "WriteOnly", "IncludeManagementEvents":false,
    "DataResources": [{ "Type": "AWS::S3::Object", "Values": ["arn:aws:s3:::amzn-s3-
   demo-source-bucket/myFolder/file.zip"] }] }]'
   ```

Para crear una EventBridge regla con Amazon S3 como origen y destino del evento CodePipeline , y aplicar la política de permisos

1. Otorgue los permisos EventBridge para usarlos CodePipeline para invocar la regla. Para obtener más información, consulte Uso de políticas basadas en recursos para Amazon. EventBridge

   a. Usa el siguiente ejemplo para crear la política de confianza que te permita EventBridge asumir la función de servicio. Denomínelo `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Utilice el comando para crear el rol `Role-for-MyRule` y asocie la política de confianza.

      ¿Por qué voy a hacer este cambio? Al agregar esta política de confianza al rol, se crean permisos para EventBridge.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Cree el JSON de la política de permisos tal y como se muestra aquí para la canalización denominada `MyFirstPipeline`. Ponga un nombre a la política de permisos `permissionspolicyforEB.json`.

      JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilice el siguiente comando para asociar la nueva política de permisos `CodePipeline-Permissions-Policy-for-EB` al rol `Role-for-MyRule` que ha creado.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Llame al comando put-rule e incluya los parámetros `--name`, `--event-pattern` y `--role-arn`.

   El siguiente comando de ejemplo crea una regla denominada `MyS3SourceRule`.

```
aws events put-rule --name "MyS3SourceRule" --event-pattern "{\"source\":
[\"aws.s3\"],\"detail-type\":[\"AWS API Call via CloudTrail\"],\"detail\":
{\"eventSource\":[\"s3.amazonaws.com\"],\"eventName\":[\"CopyObject\",\"PutObject
\",\"CompleteMultipartUpload\"],\"requestParameters\":{\"bucketName\":[\"amzn-s3-
demo-source-bucket\"],\"key\":[\"my-key\"]}}}
 --role-arn "arn:aws:iam::ACCOUNT_ID:role/Role-for-MyRule"
```

3. Para CodePipeline añadirla como destino, ejecute el put-targets comando e incluya los `--targets` parámetros `--rule` y.

   El siguiente comando especifica que, para la regla denominada `MyS3SourceRule`, el destino `Id` se compone del número uno, lo que indica que, en lo que puede ser una lista de destinos de la regla, se trata del destino 1. El comando también especifica un ARN de ejemplo para la canalización. La canalización se inicia cuando se produce algún cambio en el repositorio.

```
aws events put-targets --rule MyS3SourceRule --targets
 Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

4. (Opcional) Para configurar un transformador de entrada con anulaciones de origen para un ID de imagen específico, utilice el siguiente fragmento de código JSON en el comando de la CLI. En el siguiente ejemplo, se configura una anulación donde:

   - `actionName` (`Source` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - `revisionType` (`S3_OBJECT_VERSION_ID` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - En este ejemplo`revisionValue`, **<** *`revisionValue`* **>** se deriva de la variable de evento de origen.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "ARN",
            "InputTransformer": {
                "InputPathsMap": {
                    "revisionValue": "$.detail.object.version-id"
                },
                "InputTemplate": {
                    "sourceRevisions": {
                        "actionName": "Source",
                        "revisionType": "S3_OBJECT_VERSION_ID",
                        "revisionValue": "<revisionValue>"
                    }
                }
            }
        }
    ]
}
```

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

1. Ejecute el comando get-pipeline para copiar la estructura de canalización en un archivo JSON. Por ejemplo, para una canalización denominada `MyFirstPipeline`, escriba el siguiente comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando no devuelve nada, pero el archivo creado debería aparecer en el directorio en el que se ejecutó el comando.

2. Abra el archivo JSON en cualquier editor de texto sin formato y edite la etapa de código fuente cambiando el parámetro `PollForSourceChanges` del bucket denominado `amzn-s3-demo-source-bucket` a `false`, tal y como se muestra en este ejemplo.

   ¿Por qué voy a hacer este cambio? Al establecer este parámetro en `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Si está trabajando con la estructura de la canalización recuperada mediante el comando get-pipeline, debe eliminar las líneas `metadata` del archivo JSON. De lo contrario, el comando update-pipeline no puede utilizarlo. Elimine las líneas `"metadata": { }` y los campos `"updated"`, `"created"` y `"pipelineARN"`.

Por ejemplo, quite las siguientes líneas de la estructura:

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

Guarde el archivo.

4. Para aplicar los cambios, ejecute el comando update-pipeline especificando el archivo JSON:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

Este comando devuelve la estructura completa de la canalización editada.

> **Note**
>
> El comando update-pipeline detiene la canalización. Si se está ejecutando una revisión en la canalización al ejecutar el comando update-pipeline, dicha ejecución se detiene. Debe iniciar manualmente la canalización para ejecutar dicha revisión en la canalización actualizada. Utilice el comando start-pipeline-execution para iniciar manualmente la canalización.

## Cree una EventBridge regla para una fuente de Amazon S3 (CloudFormation plantilla)

Para usarla CloudFormation para crear una regla, actualice la plantilla como se muestra aquí.

Para crear una EventBridge regla con Amazon S3 como origen y CodePipeline destino del evento y aplicar la política de permisos

1. En la sección de abajo de la plantilla`Resources`, utilice el `AWS::IAM::Role` CloudFormation recurso para configurar el rol de IAM que permite que su evento inicie su canalización. Esta entrada crea un rol que utiliza dos políticas:

   - La primera política permite asumir el rol.
   - La segunda política concede permisos para iniciar la canalización.

   ¿Por qué voy a hacer este cambio? Añadir `AWS::IAM::Role` un recurso permite CloudFormation crear permisos para EventBridge. Este recurso se agrega a su CloudFormation pila.

   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. Usa el `AWS::Events::Rule` CloudFormation recurso para agregar una EventBridge regla. Este patrón de eventos crea un evento que monitoriza `CopyObject`, `PutObject` y `CompleteMultipartUpload` en el bucket de origen de Amazon S3. Además, incluya un objetivo de la canalización. Cuando se produce `CopyObject`, `PutObject` o `CompleteMultipartUpload`, esta regla invoca `StartPipelineExecution` en la canalización de destino.

   ¿Por qué voy a hacer este cambio? Agregar el `AWS::Events::Rule` recurso permite CloudFormation crear el evento. Este recurso se añade a tu CloudFormation pila.

   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. Añada este fragmento a su primera plantilla para permitir la funcionalidad de pila cruzada:

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. (Opcional) Para configurar un transformador de entrada con anulaciones de origen para un ID de imagen específico, utilice el siguiente fragmento de código YAML. En el siguiente ejemplo, se configura una anulación donde:

   - `actionName` (`Source` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - `revisionType` (`S3_OBJECT_VERSION_ID` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - En este ejemplo`revisionValue`, **<** *`revisionValue`* **>** se deriva de la variable de evento de origen.

```
---
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
```

```
    InputPathsMap:
      revisionValue: "$.detail.object.version-id"
    InputTemplate:
      sourceRevisions:
        actionName: Source
        revisionType: S3_OBJECT_VERSION_ID
        revisionValue: '<revisionValue>'
```

5. Guarde la plantilla actualizada en su equipo local y abra la CloudFormation consola.
6. Seleccione la pila y luego elija Create Change Set for Current Stack (Crear conjuntos de cambios para la pila actual).
7. Cargue su plantilla actualizada y, a continuación, consulte los cambios indicados en CloudFormation. Estos son los cambios que se realizarán en la pila. Debería ver los nuevos recursos en la lista.
8. Elija Ejecutar.

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

- En la plantilla, cambie `PollForSourceChanges` por `false`. Si no ha incluido `PollForSourceChanges` en la definición de la canalización, añádalo y establézcalo en `false`.

  ¿Por qué voy a hacer este cambio? Al cambiar `PollForSourceChanges` a `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

  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
 }
```

Para crear una segunda plantilla para los CloudTrail recursos de su canalización de Amazon S3

- En una plantilla independiente, en`Resources`, utilice los `AWS::S3::Bucket` `AWS::CloudTrail::Trail` CloudFormation recursos y para proporcionar una definición y un registro sencillos del bucket CloudTrail. `AWS::S3::BucketPolicy`

  ¿Por qué voy a hacer este cambio? Dado el límite actual de cinco rutas por cuenta, la CloudTrail ruta se debe crear y gestionar por separado. (Consulta la sección Límites en AWS CloudTrail.) Sin embargo, puede incluir muchos buckets de Amazon S3 en un único registro de seguimiento, por lo que puede crear el registro de seguimiento una vez y, a continuación, añadir buckets de Amazon S3 para otras canalizaciones según sea necesario. Pegue lo siguiente en el segundo archivo de plantilla de ejemplo.

  YAML

```
################################################################################
# Prerequisites:
#   - S3 SourceBucket and SourceObjectKey must exist
################################################################################

Parameters:
  SourceObjectKey:
    Description: 'S3 source artifact'
    Type: String
    Default: SampleApp_Linux.zip

Resources:
  AWSCloudTrailBucketPolicy:
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket: !Ref AWSCloudTrailBucket
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          -
            Sid: AWSCloudTrailAclCheck
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:GetBucketAcl
            Resource: !GetAtt AWSCloudTrailBucket.Arn
```

```
          -
            Sid: AWSCloudTrailWrite
            Effect: Allow
            Principal:
              Service:
                - cloudtrail.amazonaws.com
            Action: s3:PutObject
            Resource: !Join [ '', [ !GetAtt AWSCloudTrailBucket.Arn, '/
AWSLogs/', !Ref 'AWS::AccountId', '/*' ] ]
            Condition:
              StringEquals:
                s3:x-amz-acl: bucket-owner-full-control
  AWSCloudTrailBucket:
    Type: AWS::S3::Bucket
    DeletionPolicy: Retain
  AwsCloudTrail:
    DependsOn:
      - AWSCloudTrailBucketPolicy
    Type: AWS::CloudTrail::Trail
    Properties:
      S3BucketName: !Ref AWSCloudTrailBucket
      EventSelectors:
        -
          DataResources:
            -
              Type: AWS::S3::Object
              Values:
                - !Join [ '', [ !ImportValue SourceBucketARN, '/', !Ref
 SourceObjectKey ] ]
          ReadWriteType: WriteOnly
          IncludeManagementEvents: false
      IncludeGlobalServiceEvents: true
      IsLogging: true
      IsMultiRegionTrail: true


...
```

JSON

```
{
  "Parameters": {
    "SourceObjectKey": {
```

```
      "Description": "S3 source artifact",
      "Type": "String",
      "Default": "SampleApp_Linux.zip"
    }
  },
  "Resources": {
    "AWSCloudTrailBucket": {
      "Type": "AWS::S3::Bucket",
        "DeletionPolicy": "Retain"
    },
    "AWSCloudTrailBucketPolicy": {
      "Type": "AWS::S3::BucketPolicy",
      "Properties": {
        "Bucket": {
          "Ref": "AWSCloudTrailBucket"
        },
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "AWSCloudTrailAclCheck",
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "cloudtrail.amazonaws.com"
                ]
              },
              "Action": "s3:GetBucketAcl",
              "Resource": {
                "Fn::GetAtt": [
                  "AWSCloudTrailBucket",
                  "Arn"
                ]
              }
            },
            {
              "Sid": "AWSCloudTrailWrite",
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "cloudtrail.amazonaws.com"
                ]
              },
              "Action": "s3:PutObject",
```

```
            "Resource": {
              "Fn::Join": [
                "",
                [
                  {
                    "Fn::GetAtt": [
                      "AWSCloudTrailBucket",
                      "Arn"
                    ]
                  },
                  "/AWSLogs/",
                  {
                    "Ref": "AWS::AccountId"
                  },
                  "/*"
                ]
              ]
            },
            "Condition": {
              "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control"
              }
            }
          }
        ]
      }
    }
  },
  "AwsCloudTrail": {
    "DependsOn": [
      "AWSCloudTrailBucketPolicy"
    ],
    "Type": "AWS::CloudTrail::Trail",
    "Properties": {
      "S3BucketName": {
        "Ref": "AWSCloudTrailBucket"
      },
      "EventSelectors": [
        {
          "DataResources": [
            {
              "Type": "AWS::S3::Object",
              "Values": [
                {
```

```
                    "Fn::Join": [
                      "",
                      [
                        {
                          "Fn::ImportValue": "SourceBucketARN"
                        },
                        "/",
                        {
                          "Ref": "SourceObjectKey"
                        }
                      ]
                    ]
                  }
                ]
              }
            ],
            "ReadWriteType": "WriteOnly",
            "IncludeManagementEvents": false
          }
        ],
        "IncludeGlobalServiceEvents": true,
        "IsLogging": true,
        "IsMultiRegionTrail": true
      }
    }
  }
}

...
```

## CodeCommit acciones de origen y EventBridge

Para añadir una acción CodeCommit fuente CodePipeline, puedes elegir entre las siguientes opciones:

- Usa el asistente de creación de canalizaciones (Creación de una canalización personalizada (consola)) de la CodePipeline consola o la página de edición de acciones para elegir la opción CodeCommit de proveedor. La consola crea una EventBridge regla que inicia la canalización cuando cambia la fuente.
- AWS CLI Utilícela para añadir la configuración de acción a la `CodeCommit` acción y crear recursos adicionales de la siguiente manera:

- Utilice el ejemplo de configuración de acciones `CodeCommit` en CodeCommit referencia de acción de origen para crear su acción, como se muestra en Crear una canalización (CLI).
- El método de detección de cambios consiste de forma predeterminada en iniciar la canalización sondeando el origen. Debe deshabilitar las comprobaciones periódicas y crear la regla de detección de cambios manualmente. Utilice uno de los siguientes métodos: Cree una EventBridge regla para una CodeCommit fuente (consola), Crea una EventBridge regla para una CodeCommit fuente (CLI) o Crea una EventBridge regla para una CodeCommit fuente (CloudFormation plantilla).

Temas


- Cree una EventBridge regla para una CodeCommit fuente (consola)
- Crea una EventBridge regla para una CodeCommit fuente (CLI)
- Crea una EventBridge regla para una CodeCommit fuente (CloudFormation plantilla)


## Cree una EventBridge regla para una CodeCommit fuente (consola)

> **Important**
>
> Si usas la consola para crear o editar tu canalización, la EventBridge regla se creará automáticamente.

Para crear una EventBridge regla para usarla en CodePipeline las operaciones

1. Abre la EventBridge consola de Amazon en https://console.aws.amazon.com/events/.
2. En el panel de navegación, seleccione Reglas. Deje el bus predeterminado seleccionado o elija un bus de eventos. Elija Crear regla.
3. En Nombre, introduzca un nombre para la regla.
4. En Tipo de regla, elija Regla con un patrón de evento. Elija Siguiente.
5. En Fuente del evento, elige AWS eventos o eventos de EventBridge socios.
6. En Ejemplo de tipo de evento, seleccione Eventos de AWS .
7. En Eventos de muestra, escribe CodeCommit la palabra clave por la que quieres filtrar. Seleccione CodeCommit Repository State Change.

8. En Método de creación, elija Patrón de cliente (JSON editor).

   Pegue el patrón de eventos que se muestra a continuación. El siguiente es un patrón de eventos de muestra de CodeCommit en la ventana Event (Evento) para el repositorio `MyTestRepo` con una ramificación denominada `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. En Objetivos, elija CodePipeline.
10. Introduzca el ARN de la canalización que iniciará esta regla.

> **Note**
>
> Puede encontrar el ARN de la canalización en la salida de metadatos después de ejecutar el comando get-pipeline. El ARN de canalización se crea con el siguiente formato:
> arn:aws:codepipeline::: *region account pipeline-name*
> ARN de canalización de muestra:
> `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

11. Para crear o especificar un rol de servicio de IAM que conceda EventBridge permisos para invocar el objetivo asociado a tu EventBridge regla (en este caso, el objetivo es): CodePipeline

- Elige Crear un rol nuevo para este recurso específico a fin de crear un rol de servicio que otorgue EventBridge permisos para iniciar las ejecuciones de tu canalización.
- Elige Usar el rol existente para ingresar un rol de servicio que otorgue EventBridge permisos para iniciar las ejecuciones de tu canalización.

12. (Opcional) Para especificar las anulaciones de la fuente con un ID de imagen específico, usa el transformador de entrada para pasar los datos como parámetros de JSON. También puede usar el transformador de entrada para pasar variables de canalización.

    - Amplíe Configuración adicional.

      En Configurar entrada de destino, elija Configurar transformador de entrada.

      En la ventana de cuadro de diálogo, elija Escribir el mío. En el cuadro Ruta de entrada, escriba los siguientes pares clave-valor.

      ```
      {"revisionValue": "$.detail.image-digest"}
      ```

    - En el cuadro Plantilla, escriba los siguientes pares clave-valor.

      ```
      {
          "sourceRevisions": [
              {
                  "actionName": "Source",
                  "revisionType": "COMMIT_ID",
                  "revisionValue": "<revisionValue>"
              }
          ],
          "variables": [
              {
                  "name": "Branch_Name",
                  "value": "value"
              }
          ]
      }
      ```

    - Seleccione Confirmar.

13. Elija Siguiente.
14. En la página Etiquetas, elija Siguiente:

15. En la página Revisar y crear, revise la configuración de la regla. Si está satisfecho con la regla, elija Create rule (Crear regla).

## Crea una EventBridge regla para una CodeCommit fuente (CLI)

Ejecute el comando put-rule especificando lo siguiente:

- Un nombre que identifique de forma inequívoca la regla que está creando. Este nombre debe ser único en todas las canalizaciones que crees y que estén CodePipeline asociadas a tu AWS cuenta.
- El patrón de eventos para el origen y los campos de detalles utilizados por la regla. Para obtener más información, consulta Amazon EventBridge y Event Patterns.

Para crear una EventBridge CodeCommit regla con el origen del evento y CodePipeline el destino

1. Agregue los permisos EventBridge para usarlos CodePipeline para invocar la regla. Para obtener más información, consulte Uso de políticas basadas en recursos para Amazon. EventBridge

   a. Usa el siguiente ejemplo para crear la política de confianza que te permita EventBridge asumir el rol de servicio. Ponga un nombre a la política de confianza `trustpolicyforEB.json`.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   b. Utilice el comando para crear el rol `Role-for-MyRule` y asocie la política de confianza.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Cree el JSON de la política de permisos, tal y como se muestra en este ejemplo para la canalización denominada `MyFirstPipeline`. Ponga un nombre a la política de permisos `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilice el siguiente comando para asociar la política de permisos `CodePipeline-Permissions-Policy-for-EB` al rol `Role-for-MyRule`.

¿Por qué voy a hacer este cambio? Al agregar esta política al rol, se crean permisos para EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Llame al comando put-rule e incluya los parámetros `--name`, `--event-pattern` y `--role-arn`.

¿Por qué voy a hacer este cambio? Este comando permite que CloudFormation cree el evento.

El siguiente comando de ejemplo crea una regla llamada `MyCodeCommitRepoRule`.

```
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. Para agregarla CodePipeline como destino, ejecute el put-targets comando e incluya los siguientes parámetros:

   - El parámetro `--rule` se usa con el `rule_name` que creó con el comando put-rule.
   - El parámetro `--targets` se usa con el `Id` del destino de la lista de destinos y el ARN de la canalización de destino.

   El siguiente comando de muestra especifica que, para la regla denominada `MyCodeCommitRepoRule`, el destino `Id` se compone del número uno, lo que indica que, en lo que puede ser una lista de destinos de la regla, se trata del destino 1. El comando de muestra también especifica un ARN de ejemplo para la canalización. La canalización se inicia cuando se produce algún cambio en el repositorio.

```
aws events put-targets --rule MyCodeCommitRepoRule --targets
 Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

4. (Opcional) Para configurar un transformador de entrada con anulaciones de origen para un ID de imagen específico, utilice el siguiente fragmento de código JSON en el comando de la CLI. En el siguiente ejemplo, se configura una anulación donde:

   - `actionName` (`Source` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - `revisionType` (`COMMIT_ID` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - En este ejemplo`revisionValue`, **<** *`revisionValue`* **>** se deriva de la variable de evento de origen.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
```

```
            "Id": "MyTargetId",
            "Arn": "pipeline-ARN",
            "InputTransformer": {
                "sourceRevisions": {
                    "actionName": "Source",
                    "revisionType": "COMMIT_ID",
                    "revisionValue": "<revisionValue>"
                },
                "variables": [
                    {
                        "name": "Branch_Name",
                        "value": "value"
                    }
                ]
            }
        }
    ]
}
```

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> Al crear una canalización con este método, el parámetro `PollForSourceChanges` se establece en true de forma predeterminada si no se establece explícitamente en false. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

1. Ejecute el comando get-pipeline para copiar la estructura de canalización en un archivo JSON. Por ejemplo, para una canalización denominada `MyFirstPipeline`, escriba el siguiente comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando no devuelve nada, pero el archivo creado debería aparecer en el directorio en el que se ejecutó el comando.

2. Abra el archivo JSON en cualquier editor de texto sin formato y edite la etapa de origen cambiando el parámetro `PollForSourceChanges` por `false`, tal y como se muestra en este ejemplo.

   ¿Por qué voy a hacer este cambio? Al cambiar este parámetro a `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

   ```
   "configuration": {
       "PollForSourceChanges": "false",
       "BranchName": "main",
       "RepositoryName": "MyTestRepo"
   },
   ```

3. Si está trabajando con la estructura de la canalización recuperada mediante el comando get-pipeline, elimine las líneas `metadata` del archivo JSON. De lo contrario, el comando update-pipeline no puede utilizarlo. Elimine las líneas `"metadata": { }` y los campos `"updated"`, `"created"` y `"pipelineARN"`.

   Por ejemplo, quite las siguientes líneas de la estructura:

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

   Guarde el archivo.

4. Para aplicar los cambios, ejecute el comando update-pipeline especificando el archivo JSON:

   > **Important**
   >
   > Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

   Este comando devuelve la estructura completa de la canalización editada.

> **Note**
>
> El comando update-pipeline detiene la canalización. Si se está ejecutando una revisión en la canalización al ejecutar el comando update-pipeline, dicha ejecución se detiene. Debe iniciar manualmente la canalización para ejecutar dicha revisión en la canalización actualizada. Utilice el comando **`start-pipeline-execution`** para iniciar manualmente la canalización.

## Crea una EventBridge regla para una CodeCommit fuente (CloudFormation plantilla)

Para usarla CloudFormation para crear una regla, actualice la plantilla como se muestra aquí.

Para actualizar tu canalización CloudFormation crear una plantilla y crear una EventBridge regla

1. En la sección inferior de la plantilla`Resources`, usa el `AWS::IAM::Role` CloudFormation recurso para configurar la función de IAM que permitirá que tu evento inicie su proceso. Esta entrada crea un rol que utiliza dos políticas:

   - La primera política permite asumir el rol.
   - La segunda política concede permisos para iniciar la canalización.

   ¿Por qué voy a hacer este cambio? Añadir el `AWS::IAM::Role` recurso permite CloudFormation crear permisos para EventBridge. Este recurso se agrega a su CloudFormation pila.

   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. En la plantilla, en`Resources`, usa el `AWS::Events::Rule` CloudFormation recurso para agregar una EventBridge regla. Este patrón de eventos crea un evento que monitoriza la introducción de cambios en su repositorio. Cuando EventBridge detecta un cambio en el estado del repositorio, la regla se invoca `StartPipelineExecution` en tu canalización de destino.

   ¿Por qué voy a hacer este cambio? Añadir el `AWS::Events::Rule` recurso permite CloudFormation crear el evento. Este recurso se añade a tu CloudFormation pila.

   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. (Opcional) Para configurar un transformador de entrada con anulaciones de origen para un ID de imagen específico, utilice el siguiente fragmento de código YAML. En el siguiente ejemplo, se configura una anulación donde:

   - `actionName` (`Source` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - `revisionType` (`COMMIT_ID` en este ejemplo) es el valor dinámico, que se define en la creación de la canalización, no se deriva del evento de origen.
   - En este ejemplo`revisionValue`, **<** *`revisionValue`* **>** se deriva de la variable de evento de origen.
   - Se especifican las variables de salida para `BranchName` y `Value`.

```
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
    sourceRevisions:
      actionName: Source
      revisionType: COMMIT_ID
      revisionValue: <revisionValue>
    variables:
    - name: BranchName
      value: value
```

4. Guarde la plantilla actualizada en su equipo local y, a continuación, abra la CloudFormation consola.

5. Seleccione la pila y luego elija Create Change Set for Current Stack (Crear conjuntos de cambios para la pila actual).
6. Cargue la plantilla y, a continuación, consulte los cambios indicados en CloudFormation. Estos son los cambios que se realizan en la pila. Debería ver los nuevos recursos en la lista.
7. Elija Ejecutar.

Para editar el PollForSourceChanges parámetro de tu canalización

> **Important**
>
> En muchos casos, el parámetro `PollForSourceChanges` es true de forma predeterminada al crear una canalización. Al añadir la detección de cambios basada en eventos, debe añadir el parámetro a la salida y establecerlo en false para deshabilitar el sondeo. De lo contrario, la canalización comienza dos veces para un único cambio en el código fuente. Para obtener más información, consulte Configuración válida para el parámetro `PollForSourceChanges`.

- En la plantilla, cambie `PollForSourceChanges` por `false`. Si no ha incluido `PollForSourceChanges` en la definición de la canalización, añádalo y establézcalo en `false`.

  ¿Por qué voy a hacer este cambio? Al cambiar este parámetro a `false`, se desactivan las comprobaciones periódicas, por lo que únicamente puede utilizar la detección de cambios basada en eventos.

  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
    }
  ]
},
```

# Agrega proveedores de fuentes de terceros a las canalizaciones mediante AWS CodeConnections

Puedes usar la AWS CodePipeline consola o la AWS CLI para conectar tu canalización a repositorios de terceros.

> **Note**
>
> Si utiliza la consola para crear o editar una canalización, los recursos de detección de cambios se crean automáticamente. Si usa la AWS CLI para crear la canalización, debe crear los recursos adicionales usted mismo. Para obtener más información, consulte CodeCommit acciones de origen y EventBridge.

Temas


- DevOps Conexiones de Azure
- Conexiones de Bitbucket Cloud
- GitHub conexiones
- GitHub Conexiones de Enterprise Server
- GitLab.com conexiones
- Conexiones para GitLab autogestión
- Uso de una conexión compartida con otra cuenta


## DevOps Conexiones de Azure

Las conexiones le permiten autorizar y establecer configuraciones que asocian a su proveedor externo con sus AWS recursos. Para asociar su repositorio de terceros como origen de su canalización, debe usar una conexión.

> **Note**
>
> En lugar de crear o usar una conexión existente en su cuenta, puede usar una conexión compartida con otra Cuenta de AWS. Consulte Uso de una conexión compartida con otra cuenta.

> **Note**
>
> Esta función no está disponible en las regiones Asia Pacífico (Hong Kong), Asia Pacífico (Hyderabad), Asia Pacífico (Yakarta), Asia Pacífico (Melbourne), Asia Pacífico (Osaka), África (Ciudad del Cabo), Oriente Medio (Baréin), Oriente Medio (Emiratos Árabes Unidos), Europa (España), Europa (Zúrich), Israel (Tel Aviv) o AWS GovCloud (). US-West Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline. Para ver consideraciones sobre esta acción en la región de Europa (Milán), consulte la nota que aparece en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

Para añadir una acción de DevOps origen de Azure CodePipeline, puede elegir entre las siguientes opciones:

- Utilice el asistente de creación de canalización de la CodePipeline consola o la página de edición de acciones para elegir la opción DevOps de proveedor de Azure. Consulte Creación de una conexión a Bitbucket Cloud (consola) para añadir la acción. La consola le ayuda a crear un recurso de conexiones.
- Use la CLI para agregar la configuración de acción para la `CreateSourceConnection` acción con el proveedor `Azure DevOps` de la siguiente manera:
  - Para crear sus recursos de conexiones, consulte Crear una conexión a Azure DevOps (CLI) para crear un recurso de conexiones con la CLI.
  - Utilice el ejemplo `CreateSourceConnection` de configuración de acciones en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas para añadir la acción, como se muestra en Crear una canalización (CLI).

> **Note**
>
> También puede crear una conexión mediante la consola de Herramientas para desarrolladores, en Configuración. Consulte Crear una conexión.

Antes de empezar:

- Debe haber creado una cuenta con el proveedor del repositorio de terceros, como Azure DevOps
- Debe haber creado ya un repositorio de código de terceros, como un DevOps repositorio de Azure.

> **Note**
>
> DevOps Las conexiones de Azure solo proporcionan acceso a los repositorios que son propiedad de la DevOps cuenta de Azure que se utilizó para crear la conexión.
> Para instalar la conexión, debe tener permisos de administrador para la cuenta de Azure.

Temas


- Cree una conexión a Azure DevOps (consola)
- Crear una conexión a Azure DevOps (CLI)


## Cree una conexión a Azure DevOps (consola)

Siga estos pasos para usar la CodePipeline consola y agregar una acción de conexión a su DevOps repositorio de Azure.

### Paso 1: Crear o editar la canalización

Para crear o editar la canalización

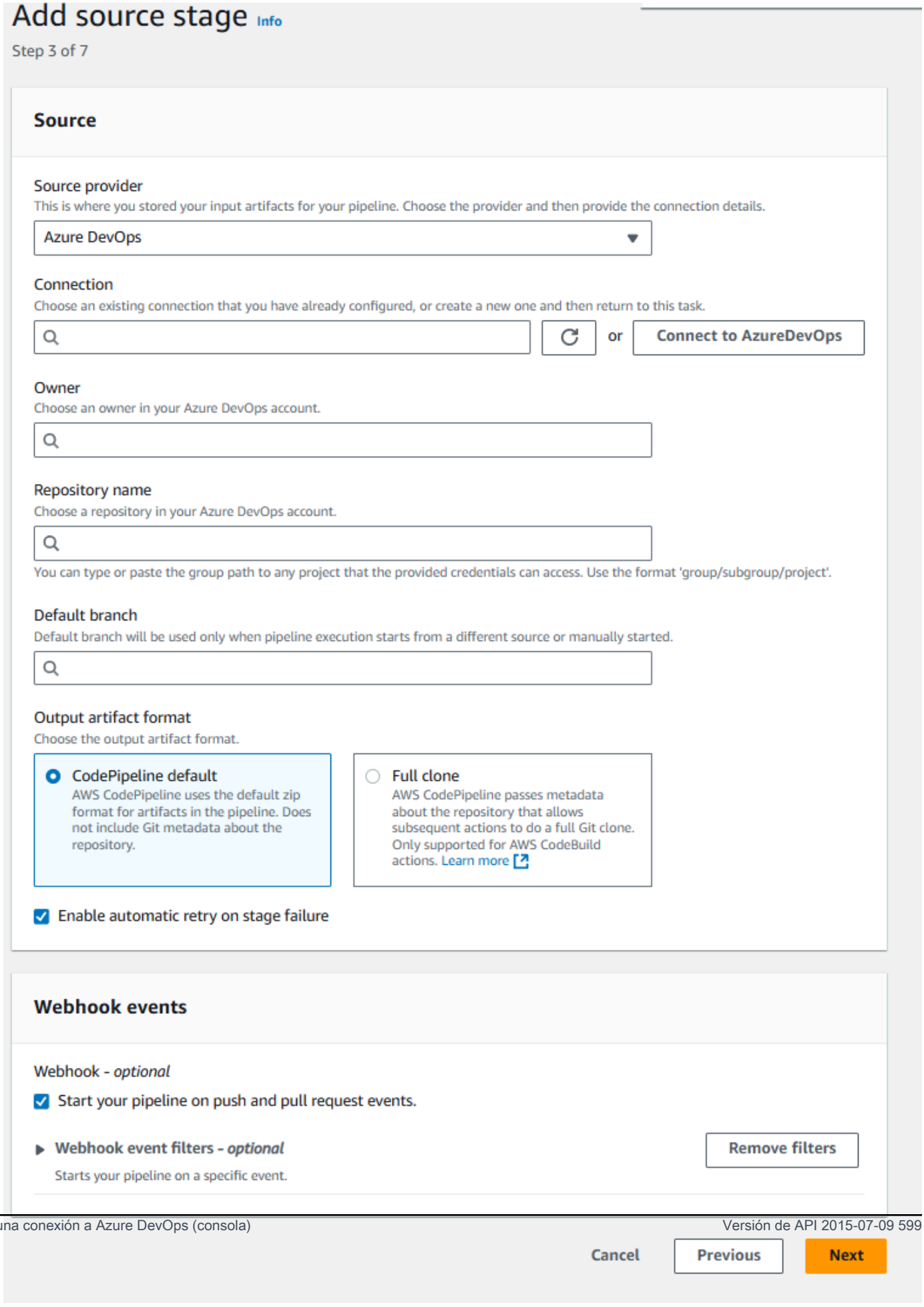
1. Inicie sesión en la CodePipeline consola.
2. Elija una de las siguientes opciones.
   - Elija Crear canalización. Siga los pasos de Crear una canalización para completar la primera pantalla y seleccione Siguiente. En la página de origen, en Proveedor de código fuente, elija Azure DevOps.
   - Elija editar una canalización existente. Elija Editar y, a continuación, elija Editar etapa. Elija añadir o editar su acción de origen. En la página Editar acción, en Nombre de la acción, introduzca el nombre de la acción. En el proveedor de acciones, elija Azure DevOps.
3. Realice una de las siguientes acciones:
   - En Conexión, si aún no ha creado una conexión con su proveedor, elija Conectarse a Azure DevOps. Continúe con el paso 2: cree una conexión a Azure DevOps.

- En Conexión, si ya ha creado una conexión con su proveedor, seleccione la conexión. Continúe con el Paso 3: Guardar la acción de origen para la conexión.

## Paso 2: Crear una conexión a Azure DevOps

Para crear una conexión a Azure DevOps

1. Para crear una conexión a un DevOps repositorio de Azure, en Seleccione un proveedor, elija Azure DevOps. En Nombre de la conexión, introduzca el nombre de la conexión que desea crear. Elija Conectarse a Azure DevOps y continúe con el paso 2.


Add source stage Info
Step 3 of 7
Source
Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
Azure DevOps
Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.
or
Connect to AzureDevOps
Owner
Choose an owner in your Azure DevOps account.
Repository name
Choose a repository in your Azure DevOps account.
You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.
Default branch
Default branch will be used only when pipeline execution starts from a different source or manually started.
Output artifact format
Choose the output artifact format.
CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.
Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions. Learn more
Enable automatic retry on stage failure
Webhook events
Webhook - optional
Start your pipeline on push and pull request events.
Webhook event filters - optional
Starts your pipeline on a specific event.
Remove filters
Cancel
Previous
Next

2. En la página de DevOps configuración de Connect to Azure, aparece el nombre de la conexión.

   En  DevOps Aplicaciones de Azure, elija una instalación de aplicación o seleccione Instalar una nueva aplicación para crear una.

   > **Note**
   >
   > Solo debe instalar la aplicación una vez para cada DevOps cuenta de Azure. Si ya ha instalado la aplicación del conector, elíjala y diríjase al último paso de esta sección.

3. Si se muestra la página de inicio de sesión de Microsoft, inicie sesión con sus credenciales y luego elija continuar.
4. En la página de instalación de la aplicación, aparece un mensaje que indica que la aplicación conectora está intentando conectarse a su DevOps cuenta de Azure.

   Elija Conceder acceso.
5. En la página de conexión, aparece el ID de conexión de su nueva instalación. Elija Conectar. La conexión creada se muestra en la lista de conexiones.

## Paso 3: Guarde la acción DevOps fuente de Azure

Siga estos pasos del asistente o de la página de Editar acción para guardar la acción de origen con la información de conexión.

Para completar y guardar la acción de origen con la conexión

1. En Repository name (Nombre del repositorio), elija el nombre del repositorio de terceros.
2. En los activadores de Pipeline, puede añadir activadores si su acción es una AWS CodeConnections acción. Para configurar los desencadenadores de canalización y, de forma opcional, filtrar con desencadenadores, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código.
3. En Output artifact format (Formato del artefacto de salida), debe elegir el formato de los artefactos.

   - Para almacenar los artefactos de salida de la DevOps acción de Azure mediante el método predeterminado, elija CodePipeline default. La acción obtiene acceso a los archivos del repositorio y almacena los artefactos en un archivo ZIP en el almacén de artefactos de canalización.

- Para almacenar un archivo JSON que contiene una referencia URL al repositorio de manera que las acciones posteriores puedan ejecutar comandos Git directamente, elija Clonación completa. Esta opción solo la pueden utilizar las acciones CodeBuild posteriores.

  Si elige esta opción, tendrá que actualizar los permisos de su función de servicio de CodeBuild proyectos, tal y como se muestra en Añade CodeBuild GitClone permisos para las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com la siguiente.

4. Seleccione Siguiente en el asistente o Guardar en la página Editar acción.

## Crear una conexión a Azure DevOps (CLI)

Puede usar AWS Command Line Interface (AWS CLI) para crear una conexión.

Para ello, utilice el comando create-connection.

> **Important**
>
> Una conexión creada a través del AWS CLI o AWS CloudFormation está en `PENDING` estado de forma predeterminada. Después de crear una conexión con la CLI o CloudFormation, utilice la consola para editar la conexión y establecer su estado`AVAILABLE`.

Creación de una conexión

1. Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows). Utilice el AWS CLI para ejecutar el create-connection comando, especificando el `--provider-type` y `--connection-name` para la conexión. En este ejemplo, el nombre del proveedor de terceros es `AzureDevOps` y el nombre especificado para la conexión es `MyConnection`.

   ```
   aws codeconnections create-connection --provider-type AzureDevOps --connection-name
    MyConnection
   ```

   Si se ejecuta correctamente, este comando devuelve la información del ARN de la conexión, que será similar a lo siguiente.

   ```
   {
       "ConnectionArn": "arn:aws:codeconnections:us-west-2:account_id:connection/
   aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
   ```

```
}
```

2. Utilice la consola para completar la conexión. Para obtener más información, consulte Actualización de una conexión pendiente.
3. De forma predeterminada, la canalización detecta los cambios al enviar el código al repositorio de origen de conexión. Para configurar la configuración del desencadenador de canalización para la publicación manual o para las etiquetas de Git, realiza una de las siguientes acciones:

   - Para configurar la configuración de los desencadenadores de canalización para que comience únicamente con una versión manual, añade la siguiente línea a la configuración:

     ```
     "DetectChanges": "false",
     ```

   - Para configurar los desencadenadores de canalización para filtrar con ellos, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código. Por ejemplo, lo siguiente añade etiquetas de Git al nivel de canalización de la definición de JSON de canalización. En este ejemplo, `release-v0` y `release-v1` son las etiquetas de Git que se deben incluir y `release-v2` es la etiqueta de Git que se debe excluir.

     ```
     "triggers": [
                 {
                     "providerType": "CodeStarSourceConnection",
                     "gitConfiguration": {
                         "sourceActionName": "Source",
                         "push": [
                             {
                                 "tags": {
                                     "includes": [
                                         "release-v0", "release-v1"
                                     ],
                                     "excludes": [
                                         "release-v2"
                                     ]
                                 }
                             }
                         ]
                     }
                 }
             ]
     ```

# Conexiones de Bitbucket Cloud

Las conexiones te permiten autorizar y establecer configuraciones que asocien a tu proveedor externo con tus AWS recursos. Para asociar su repositorio de terceros como origen de su canalización, debe usar una conexión.

> **Note**
>
> En lugar de crear o usar una conexión existente en su cuenta, puede usar una conexión compartida con otra Cuenta de AWS. Consulte Uso de una conexión compartida con otra cuenta.

> **Note**
>
> Esta función no está disponible en las regiones Asia Pacífico (Hong Kong), Asia Pacífico (Hyderabad), Asia Pacífico (Yakarta), Asia Pacífico (Melbourne), Asia Pacífico (Osaka), África (Ciudad del Cabo), Oriente Medio (Baréin), Oriente Medio (Emiratos Árabes Unidos), Europa (España), Europa (Zúrich), Israel (Tel Aviv) o AWS GovCloud (). US-West Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline. Para ver consideraciones sobre esta acción en la región de Europa (Milán), consulte la nota que aparece en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

Para añadir una acción fuente de Bitbucket Cloud CodePipeline, puedes elegir entre las siguientes opciones:

- Usa el asistente de creación de canalizaciones de la CodePipeline consola o la página de edición de acciones para elegir la opción de proveedor de Bitbucket. Consulte Creación de una conexión a Bitbucket Cloud (consola) para añadir la acción. La consola le ayuda a crear un recurso de conexiones.

  > **Note**
  >
  > Puede crear conexiones a un repositorio de Bitbucket Cloud. Los tipos de proveedores de Bitbucket instalados, como Bitbucket Server, no son compatibles.

- Use la CLI para agregar la configuración de acción para la acción `CreateSourceConnection` con el proveedor `Bitbucket` de la siguiente manera:
  - Para crear sus recursos de conexiones, consulte Creación de una conexión a Bitbucket Cloud (CLI) para crear un recurso de conexiones con la CLI.
  - Utilice el ejemplo `CreateSourceConnection` de configuración de acciones en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas para añadir la acción, como se muestra en Crear una canalización (CLI).

> **Note**
>
> También puede crear una conexión mediante la consola de Herramientas para desarrolladores, en Configuración. Consulte Crear una conexión.

Antes de empezar:

- Debe haber creado una cuenta con el proveedor del repositorio de terceros, como Bitbucket Cloud.
- Debe haber creado un repositorio de código de terceros, como un repositorio de Bitbucket Cloud.

> **Note**
>
> Las conexiones de Bitbucket Cloud solo proporcionan acceso a los repositorios que son propiedad de la cuenta de Bitbucket Cloud que se utilizó para crear la conexión.
> Si la aplicación se va a instalar en un espacio de trabajo de Bitbucket Cloud, necesita permisos Administrar espacio de trabajo. De lo contrario, no se mostrará la opción de instalar la aplicación.

Temas

- Creación de una conexión a Bitbucket Cloud (consola)
- Creación de una conexión a Bitbucket Cloud (CLI)

# Creación de una conexión a Bitbucket Cloud (consola)

Sigue estos pasos para usar la CodePipeline consola y añadir una acción de conexión a tu repositorio de Bitbucket.

> **Note**
>
> Puede crear conexiones a un repositorio de Bitbucket Cloud. Los tipos de proveedores de Bitbucket instalados, como Bitbucket Server, no son compatibles.

## Paso 1: Crear o editar la canalización

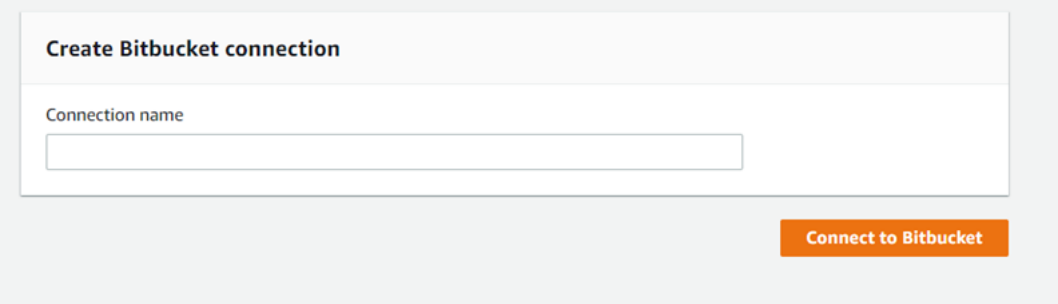
Para crear o editar la canalización

1. Inicia sesión en la CodePipeline consola.
2. Elija una de las siguientes opciones.
   - Elija Crear canalización. Siga los pasos de Crear una canalización para completar la primera pantalla y seleccione Siguiente. En la página origen, en Proveedor de origen, seleccione Bitbucket.
   - Elija editar una canalización existente. Elija Editar y, a continuación, elija Editar etapa. Elija añadir o editar su acción de origen. En la página Editar acción, en Nombre de la acción, introduzca el nombre de la acción. En Proveedor de acciones, seleccione Bitbucket.
3. Realice una de las siguientes acciones:
   - En Conexión, si aún no ha creado una conexión con su proveedor, seleccione Conectar a Bitbucket. Continúe con el Paso 2: Crear una conexión a Bitbucket.
   - En Conexión, si ya ha creado una conexión con su proveedor, seleccione la conexión. Continúe con el Paso 3: Guardar la acción de origen para la conexión.

## Paso 2: Crear una conexión a Bitbucket Cloud

Para crear una conexión a Bitbucket Cloud

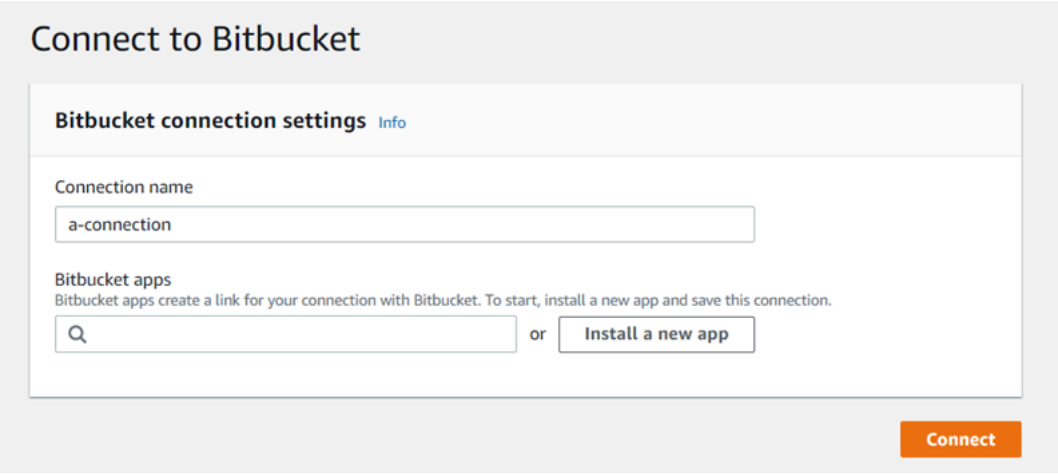
1. En la página de configuración de Conectar a Bitbucket, introduzca el nombre de su conexión y seleccione Conectar a Bitbucket.




Aparece el campo Aplicaciones de Bitbucket.

2. En Bitbucket apps (Aplicaciones de Bitbucket), elija la instalación de una aplicación o elija Install a new app (Instalar una aplicación nueva) para crear una.

> **Note**
>
> Solo instale la aplicación una vez para cada espacio de trabajo o cuenta de Bitbucket Cloud. Si ya ha instalado la aplicación de Bitbucket, elíjala y diríjase al paso 4.

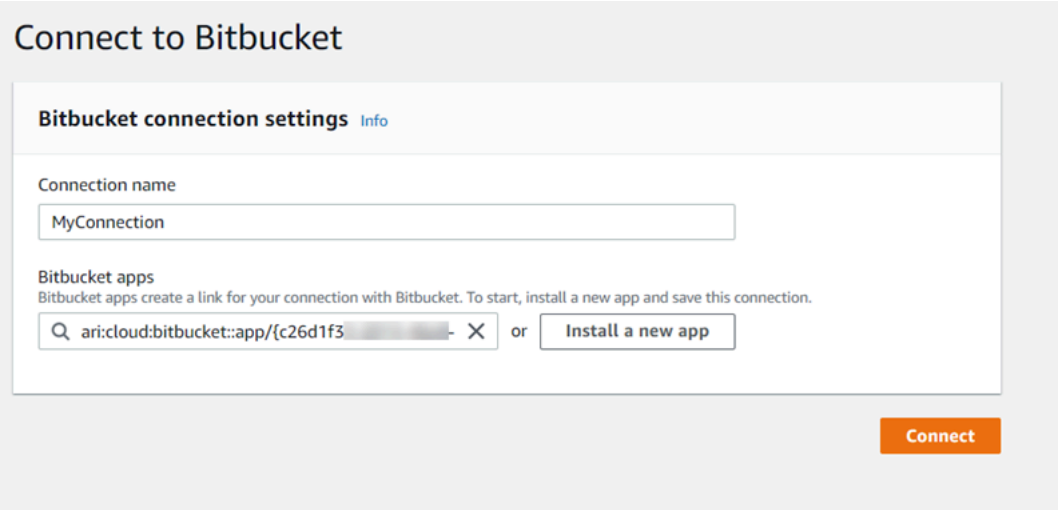



3. Si se muestra la página de inicio de sesión de Bitbucket Cloud, inicie sesión con sus credenciales y luego elija continuar.
4. En la página de instalación de la aplicación, aparece un mensaje que indica que la AWS CodeStar aplicación está intentando conectarse a tu cuenta de Bitbucket.

   Si utiliza un espacio de trabajo de Bitbucket, cambie la opción Authorize for (Autorizar para) para el espacio de trabajo. Solo se mostrarán los espacios de trabajo en los que tenga acceso de administrador.

Elija Conceder acceso.

5. En Bitbucket apps (Aplicaciones de Bitbucket), se muestra el ID de conexión de la instalación nueva. Elija Conectar. La conexión creada se muestra en la lista de conexiones.




## Paso 3: Guardar la acción origen de Bitbucket Cloud

Siga estos pasos del asistente o de la página de Editar acción para guardar la acción de origen con la información de conexión.

Para completar y guardar la acción de origen con la conexión

1. En Repository name (Nombre del repositorio), elija el nombre del repositorio de terceros.
2. En los activadores de Pipeline, puedes añadir activadores si tu acción es una AWS CodeConnections acción. Para configurar los desencadenadores de canalización y, de forma opcional, filtrar con desencadenadores, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código.
3. En Output artifact format (Formato del artefacto de salida), debe elegir el formato de los artefactos.

   - Para almacenar los artefactos de salida de la acción de Bitbucket Cloud mediante el método predeterminado, selecciona CodePipeline predeterminado. La acción obtiene acceso a los archivos del repositorio de Bitbucket Cloud y almacena los artefactos en un archivo ZIP en el almacén de artefactos de canalización.

- Para almacenar un archivo JSON que contiene una referencia URL al repositorio de manera que las acciones posteriores puedan ejecutar comandos Git directamente, elija Clonación completa. Esta opción solo la pueden utilizar las acciones CodeBuild posteriores.

  Si elige esta opción, tendrá que actualizar los permisos de su función de servicio de CodeBuild proyectos, tal y como se muestra en Añade CodeBuild GitClone permisos para las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com la siguiente.

4. Seleccione Siguiente en el asistente o Guardar en la página Editar acción.

## Creación de una conexión a Bitbucket Cloud (CLI)

Puedes usar AWS Command Line Interface (AWS CLI) para crear una conexión.

> **Note**
>
> Puede crear conexiones a un repositorio de Bitbucket Cloud. Los tipos de proveedores de Bitbucket instalados, como Bitbucket Server, no son compatibles.

Para ello, utilice el comando create-connection.

> **Important**
>
> Una conexión creada a través del AWS CLI o AWS CloudFormation está en `PENDING` estado de forma predeterminada. Después de crear una conexión con la CLI o CloudFormation, utilice la consola para editar la conexión y establecer su estado`AVAILABLE`.

Creación de una conexión

1. Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows). Utilice el AWS CLI para ejecutar el create-connection comando, especificando el `--provider-type` y `--connection-name` para la conexión. En este ejemplo, el nombre del proveedor de terceros es `Bitbucket` y el nombre especificado para la conexión es `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type Bitbucket --connection-name MyConnection
   ```

Si se ejecuta correctamente, este comando devuelve la información del ARN de la conexión, que será similar a lo siguiente.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Utilice la consola para completar la conexión. Para obtener más información, consulte Actualización de una conexión pendiente.
3. De forma predeterminada, la canalización detecta los cambios al enviar el código al repositorio de origen de conexión. Para configurar la configuración del desencadenador de canalización para la publicación manual o para las etiquetas de Git, realiza una de las siguientes acciones:

   - Para configurar la configuración de los desencadenadores de canalización para que comience únicamente con una versión manual, añade la siguiente línea a la configuración:

     ```
     "DetectChanges": "false",
     ```

   - Para configurar los desencadenadores de canalización para filtrar con ellos, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código. Por ejemplo, lo siguiente añade etiquetas de Git al nivel de canalización de la definición de JSON de canalización. En este ejemplo, `release-v0` y `release-v1` son las etiquetas de Git que se deben incluir y `release-v2` es la etiqueta de Git que se debe excluir.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
     ```

```
                        }
                    }
                ]
            }
        }
    ]
```

## GitHub conexiones

Utiliza las conexiones para autorizar y establecer configuraciones que asocien a su proveedor externo con sus AWS recursos.

> **Note**
>
> En lugar de crear o usar una conexión existente en su cuenta, puede usar una conexión compartida con otra Cuenta de AWS. Consulte Uso de una conexión compartida con otra cuenta.

> **Note**
>
> Esta función no está disponible en las regiones Asia Pacífico (Hong Kong), Asia Pacífico (Hyderabad), Asia Pacífico (Yakarta), Asia Pacífico (Melbourne), Asia Pacífico (Osaka), África (Ciudad del Cabo), Oriente Medio (Baréin), Oriente Medio (Emiratos Árabes Unidos), Europa (España), Europa (Zúrich), Israel (Tel Aviv) o AWS GovCloud (). US-West Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline. Para ver consideraciones sobre esta acción en la región de Europa (Milán), consulte la nota que aparece en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

Para añadir una acción de origen para tu repositorio GitHub o para tu repositorio de GitHub Enterprise Cloud CodePipeline, puedes elegir entre las siguientes opciones:

- Utilice el asistente de creación de canalizaciones de la CodePipeline consola o la página de edición de acciones para elegir la opción de proveedor GitHub (mediante la GitHub aplicación). Consulte Crea una conexión a GitHub (consola) para añadir la acción. La consola le ayuda a crear un recurso de conexiones.

> **Note**
>
> Para ver un tutorial que te explica cómo añadir una GitHub conexión y cómo usar la opción de clonación completa en tu canalización para clonar metadatos, consultaTutorial: Utilice un clon completo con una fuente de GitHub canalización.

- Utilice la CLI para agregar la configuración de acciones para la acción `CodeStarSourceConnection` con el proveedor `GitHub` con los pasos de la CLI que se muestran en Crear una canalización (CLI).

> **Note**
>
> También puede crear una conexión mediante la consola de Herramientas para desarrolladores, en Configuración. Consulte Crear una conexión.

Antes de empezar:

- Debes haber creado una cuenta con GitHub.
- Debe haber creado ya un repositorio GitHub de código.
- Si su rol de CodePipeline servicio se creó antes del 18 de diciembre de 2019, es posible que deba actualizar sus permisos `codestar-connections:UseConnection` para usarlos en AWS CodeStar las conexiones. Para obtener instrucciones, consulte Agrega permisos al rol de CodePipeline servicio.

> **Note**
>
> Para crear la conexión, debes ser el propietario de la GitHub organización. Para los repositorios que no pertenecen a una organización, debe ser el propietario del repositorio.

Temas

- Crea una conexión a GitHub (consola)
- Crear una conexión a GitHub (CLI)

# Crea una conexión a GitHub (consola)

Siga estos pasos para usar la CodePipeline consola y añadir una acción de conexión para su repositorio GitHub o el de GitHub Enterprise Cloud.

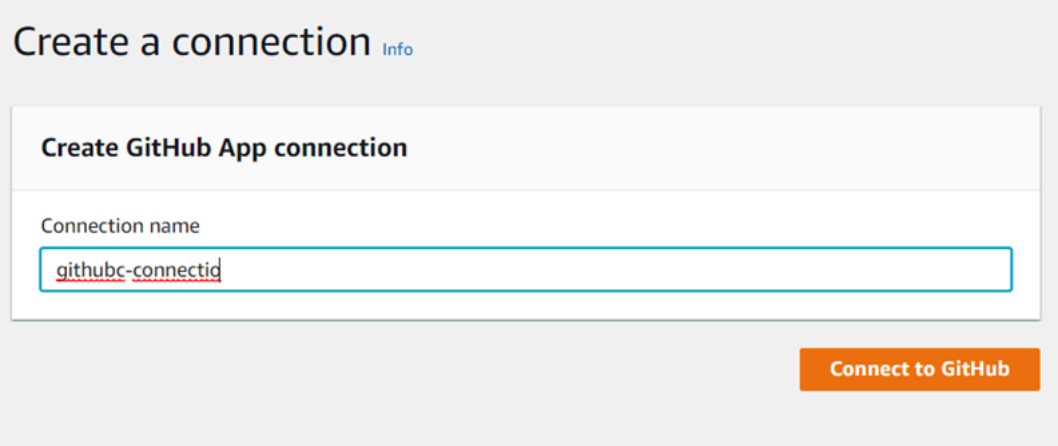
> **Note**
>
> En estos pasos, puedes seleccionar repositorios específicos en Acceso al repositorio. Los repositorios que no estén seleccionados no serán accesibles ni visibles para. CodePipeline

## Paso 1: Crear o editar la canalización

1. Inicie sesión en la CodePipeline consola.
2. Elija una de las siguientes opciones.
   - Elija Crear canalización. Siga los pasos de Crear una canalización para completar la primera pantalla y seleccione Siguiente. En la página de origen, en Proveedor de origen, selecciona GitHub (a través de GitHub la aplicación).
   - Elija editar una canalización existente. Elija Editar y, a continuación, elija Editar etapa. Elija añadir o editar su acción de origen. En la página Editar acción, en Nombre de la acción, introduzca el nombre de la acción. En el proveedor de acciones, elige GitHub (mediante GitHub la aplicación).
3. Realice una de las siguientes acciones:
   - En Conexión, si aún no ha creado una conexión con su proveedor, elija Conectar a GitHub. Continúe con el paso 2: Crear una conexión a GitHub.
   - En Conexión, si ya ha creado una conexión con su proveedor, seleccione la conexión. Continúe con el Paso 3: Guardar la acción de origen para la conexión.

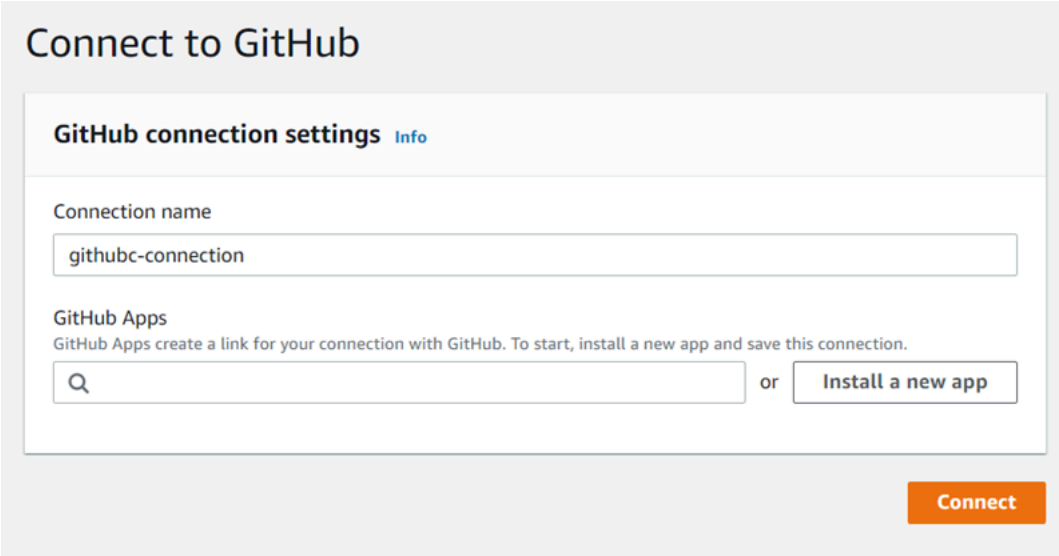
## Paso 2: Crea una conexión a GitHub

Una vez que haya decidido crear la conexión, aparecerá la GitHub página Conectar a.




Para crear una conexión a GitHub

1. En la configuración de la GitHub conexión, el nombre de la conexión aparece en Nombre de la conexión. Selecciona Conectar a GitHub. Aparece la página de solicitud de acceso.
2. Elija Autorizar AWS conector para GitHub. Aparece la página de conexión y muestra el campo GitHub Aplicaciones.




3. En GitHub Aplicaciones, selecciona la instalación de una aplicación o selecciona Instalar una nueva aplicación para crear una.

   Se instala una aplicación para todas las conexiones a un proveedor en particular. Si ya ha instalado el AWS conector para la GitHub aplicación, elíjalo y omita este paso.

> **Note**
>
> Si desea crear un token de acceso de usuario, asegúrese de que ya ha instalado el AWS Connector para la GitHub aplicación y, a continuación, deje vacío el campo de instalación de la aplicación. AWS CodeConnections utilizará el token de acceso de usuario para la conexión.

4. En la GitHub página Instalar AWS conector para, elige la cuenta en la que quieres instalar la aplicación.

> **Note**
>
> Solo instalas la aplicación una vez para cada GitHub cuenta. Si instaló la aplicación previamente, puede elegir Configurar para dirigirse a una página de modificación para la instalación de la aplicación o puede utilizar el botón Atrás para volver a la consola.

5. En la GitHub página Instalar el AWS conector para, deja los valores predeterminados y selecciona Instalar.
6. En la GitHub página Conectar a, el ID de conexión de la nueva instalación aparece en GitHub Aplicaciones. Elija Conectar.

## Paso 3: guarda la acción GitHub de origen

Siga estos pasos de la página de Editar acción para guardar la acción de origen con la información de conexión.

Para guardar la acción GitHub de origen

1. En Repository name (Nombre del repositorio), elija el nombre del repositorio de terceros.
2. En los activadores de Pipeline, puedes añadir activadores si tu acción es una AWS CodeConnections acción. Para configurar los desencadenadores de canalización y, de forma opcional, filtrar con desencadenadores, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código.
3. En Output artifact format (Formato del artefacto de salida), debe elegir el formato de los artefactos.

- Para almacenar los artefactos de salida de la GitHub acción mediante el método predeterminado, selecciona el CodePipeline predeterminado. La acción accede a los archivos del GitHub repositorio y almacena los artefactos en un archivo ZIP en el almacén de artefactos de Pipeline.
- Para almacenar un archivo JSON que contiene una referencia URL al repositorio de manera que las acciones posteriores puedan ejecutar comandos Git directamente, elija Clonación completa. Esta opción solo la pueden utilizar las acciones CodeBuild posteriores.

  Si elige esta opción, tendrá que actualizar los permisos de su función de servicio de CodeBuild proyectos, tal y como se muestra en Añade CodeBuild GitClone permisos para las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com la siguiente. Para ver un tutorial que muestra cómo utilizar la opción Clonación completa, consulte Tutorial: Utilice un clon completo con una fuente de GitHub canalización.

4. Seleccione Siguiente en el asistente o Guardar en la página Editar acción.

## Crear una conexión a GitHub (CLI)

Puede usar AWS Command Line Interface (AWS CLI) para crear una conexión.

Para ello, utilice el comando create-connection.

> **Important**
>
> Una conexión creada a través del AWS CLI o AWS CloudFormation está en `PENDING` estado de forma predeterminada. Después de crear una conexión con la CLI o CloudFormation, utilice la consola para editar la conexión y establecer su estado`AVAILABLE`.

Creación de una conexión

1. Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows). Utilice el AWS CLI para ejecutar el create-connection comando, especificando el `--provider-type` y `--connection-name` para la conexión. En este ejemplo, el nombre del proveedor de terceros es `GitHub` y el nombre especificado para la conexión es `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type GitHub --connection-name
    MyConnection
   ```

Si se ejecuta correctamente, este comando devuelve la información del ARN de la conexión, que será similar a lo siguiente.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Utilice la consola para completar la conexión. Para obtener más información, consulte Actualización de una conexión pendiente.
3. De forma predeterminada, la canalización detecta los cambios al enviar el código al repositorio de origen de conexión. Para configurar la configuración del desencadenador de canalización para la publicación manual o para las etiquetas de Git, realiza una de las siguientes acciones:

   - Para configurar la configuración de los desencadenadores de canalización para que comience únicamente con una versión manual, añade la siguiente línea a la configuración:

     ```
     "DetectChanges": "false",
     ```

   - Para configurar los desencadenadores de canalización para filtrar con ellos, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código. Por ejemplo, lo siguiente se añade al nivel de canalización de la definición de JSON de canalización. En este ejemplo, `release-v0` y `release-v1` son las etiquetas de Git que se deben incluir y `release-v2` es la etiqueta de Git que se debe excluir.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
     ```

```
                        }
                    }
                ]
            }
        }
    ]
```

## GitHub Conexiones de Enterprise Server

Las conexiones le permiten autorizar y establecer configuraciones que asocian a su proveedor externo con sus AWS recursos. Para asociar su repositorio de terceros como origen de su canalización, debe usar una conexión.

> **Note**
>
> En lugar de crear o usar una conexión existente en su cuenta, puede usar una conexión compartida con otra Cuenta de AWS. Consulte Uso de una conexión compartida con otra cuenta.

> **Note**
>
> Esta función no está disponible en las regiones Asia Pacífico (Hong Kong), Asia Pacífico (Hyderabad), Asia Pacífico (Yakarta), Asia Pacífico (Melbourne), Asia Pacífico (Osaka), África (Ciudad del Cabo), Oriente Medio (Baréin), Oriente Medio (Emiratos Árabes Unidos), Europa (España), Europa (Zúrich), Israel (Tel Aviv) o AWS GovCloud (). US-West Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline. Para ver consideraciones sobre esta acción en la región de Europa (Milán), consulte la nota que aparece en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

Para añadir una acción de origen a GitHub Enterprise Server CodePipeline, puedes elegir entre las siguientes opciones:

- Utilice el asistente de creación de canalización de la CodePipeline consola o la página de acciones de edición para elegir la opción de proveedor de GitHub Enterprise Server. Consulte Cree una

conexión a GitHub Enterprise Server (consola) para añadir la acción. La consola le ayuda a crear un recurso de host y un recurso de conexiones.

- Use la CLI para agregar la configuración de acciones para la acción `CreateSourceConnection` con el proveedor `GitHubEnterpriseServer` y crear sus recursos:
  - Para crear sus recursos de conexiones, consulte Crear un host y una conexión a GitHub Enterprise Server (CLI) para crear un recurso de host y un recurso de conexiones con la CLI.
  - Utilice el ejemplo `CreateSourceConnection` de configuración de acciones en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas para añadir la acción, como se muestra en Crear una canalización (CLI).

> **Note**
>
> También puede crear una conexión mediante la consola de Herramientas para desarrolladores, en Configuración. Consulte Crear una conexión.

Antes de empezar:

- Debe haber creado una cuenta con GitHub Enterprise Server e instalado la instancia de GitHub Enterprise Server en su infraestructura.

  > **Note**
  >
  > Cada VPC solo se puede asociar a un host (instancia de GitHub Enterprise Server) a la vez.

- Debe haber creado ya un repositorio de código con GitHub Enterprise Server.

Temas

- Cree una conexión a GitHub Enterprise Server (consola)
- Crear un host y una conexión a GitHub Enterprise Server (CLI)

# Cree una conexión a GitHub Enterprise Server (consola)

Siga estos pasos para usar la CodePipeline consola y añadir una acción de conexión al repositorio de GitHub Enterprise Server.

> **Note**
>
> GitHub Las conexiones de Enterprise Server solo proporcionan acceso a los repositorios propiedad de la cuenta de GitHub Enterprise Server que se utilizó para crear la conexión.

Antes de empezar

Para establecer una conexión de host a GitHub Enterprise Server, debe haber completado los pasos para crear un recurso de host para su conexión. Consulte Administrar hosts para conexiones.

## Paso 1: Crear o editar la canalización

Para crear o editar la canalización

1. Inicie sesión en la CodePipeline consola.
2. Elija una de las siguientes opciones.
   - Elija Crear canalización. Siga los pasos de Crear una canalización para completar la primera pantalla y seleccione Siguiente. En la página de origen, en Proveedor de origen, selecciona GitHub Enterprise Server.
   - Elija editar una canalización existente. Elija Editar y, a continuación, elija Editar etapa. Elija añadir o editar su acción de origen. En la página Editar acción, en Nombre de la acción, introduzca el nombre de la acción. En Action provider, elija GitHub Enterprise Server.
3. Realice una de las siguientes acciones:
   - En Conexión, si aún no ha creado una conexión con su proveedor, elija Connect to GitHub Enterprise Server. Continúe con el paso 2: cree una conexión a GitHub Enterprise Server.
   - En Conexión, si ya ha creado una conexión con su proveedor, seleccione la conexión. Continúe con el Paso 3: Guardar la acción de origen para la conexión.

## Cree una conexión a GitHub Enterprise Server

Después de elegir crear la conexión, se muestra la página Connect to GitHub Enterprise Server.

> **Important**
>
> AWS CodeConnections no es compatible con la versión 2.22.0 de GitHub Enterprise Server debido a un problema conocido en la versión. Para conectarse, actualice a la versión 2.22.1 o a la última versión disponible.

Para conectarse a Enterprise Server GitHub

1. En Connection name (Nombre de la conexión), ingrese el nombre para la conexión.
2. En URL, ingrese el punto de conexión para el servidor.

   > **Note**
   >
   > Si la URL proporcionada ya se ha utilizado para configurar un servidor GitHub empresarial para una conexión, se le pedirá que elija el ARN del recurso de host que se creó anteriormente para ese punto final.

3. Si lanzó su servidor en una Amazon VPC y desea conectarse a su VPC, elija Use a VPC (Utilizar una VPC) y complete lo siguiente.

   a. En VPC ID (ID de la VPC), elija el ID de su VPC. Asegúrese de elegir la VPC para la infraestructura en la que está instalada la instancia de GitHub Enterprise Server o una VPC con acceso a la instancia de GitHub Enterprise Server a través de VPN o Direct Connect.

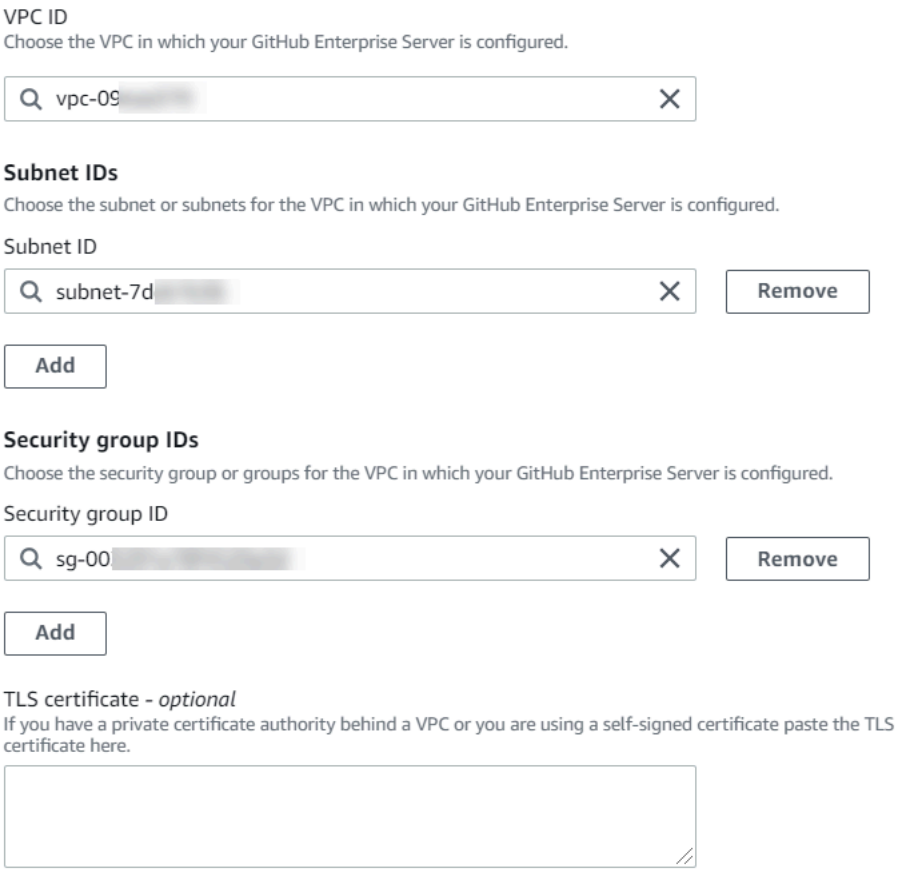
   b. En Subnet ID (ID de la subred), elija Add (Agregar). En el campo, elija el ID de la subred que desea utilizar para el alojamiento. Puede elegir hasta 10 subredes.

      Asegúrese de elegir la subred para la infraestructura en la que está instalada la instancia de GitHub Enterprise Server o una subred con acceso a la instancia de GitHub Enterprise Server instalada a través de VPN o Direct Connect.

   c. En Security group IDs (ID del grupo de seguridad), elija Add (Agregar). En el campo, elija el grupo de seguridad que desea utilizar para el alojamiento. Puede elegir hasta 10 grupos de seguridad.

Asegúrese de elegir el grupo de seguridad para la infraestructura en la que está instalada la instancia de GitHub Enterprise Server o un grupo de seguridad con acceso a la instancia de GitHub Enterprise Server instalada a través de VPN o Direct Connect.

d. Si tiene configurada una VPC privada y ha configurado su instancia de GitHub Enterprise Server para realizar la validación de TLS mediante una entidad de certificación no pública, introduzca su ID de certificado en el certificado TLS. El valor del certificado TLS debe ser la clave pública del certificado.




4. Elija Connect to GitHub Enterprise Server. La conexión creada se muestra con un estado Pendiente. Se crea un recurso de alojamiento para la conexión con la información del servidor que usted proporcionó. Se utiliza la URL para el nombre del alojamiento.
5. Elija Update pending connection (Actualizar conexión pendiente).
6. Si se le solicita, en la página de inicio de sesión de GitHub Enterprise, inicie sesión con sus credenciales de GitHub Enterprise.
7. En la página Crear GitHub aplicación, elige un nombre para la aplicación.
8. En la página de GitHub autorización, selecciona Autorizar<app-name>.

9. En la página de instalación de aplicaciones, se muestra un mensaje que indica que la aplicación Connector está lista para instalarse. Si tiene varias organizaciones, es posible que deba elegir la organización en la que desea instalar la aplicación.

   Elija la configuración del repositorio donde desea instalar la aplicación. Elija Instalar.

10. La página de conexión muestra la conexión creada en un estado Disponible.

### Paso 3: Guarde la acción de origen de GitHub Enterprise Server

Siga estos pasos del asistente o de la página de Editar acción para guardar la acción de origen con la información de conexión.

Para completar y guardar la acción de origen con la conexión

1. En Repository name (Nombre del repositorio), elija el nombre del repositorio de terceros.
2. En los activadores de Pipeline, puede añadir activadores si su acción es una AWS CodeConnections acción. Para configurar los desencadenadores de canalización y, de forma opcional, filtrar con desencadenadores, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código.
3. En Output artifact format (Formato del artefacto de salida), debe elegir el formato de los artefactos.

   - Para almacenar los artefactos de salida de la acción de GitHub Enterprise Server mediante el método predeterminado, selecciona el CodePipelinepredeterminado. La acción accede a los archivos del repositorio de GitHub Enterprise Server y almacena los artefactos en un archivo ZIP en el almacén de artefactos de Pipeline.
   - Para almacenar un archivo JSON que contiene una referencia URL al repositorio de manera que las acciones posteriores puedan ejecutar comandos Git directamente, elija Clonación completa. Esta opción solo la pueden utilizar las acciones CodeBuild posteriores.

4. Seleccione Siguiente en el asistente o Guardar en la página Editar acción.

## Crear un host y una conexión a GitHub Enterprise Server (CLI)

Puede usar el AWS Command Line Interface (AWS CLI) para crear una conexión.

Para ello, utilice el comando create-connection.

> **Important**
>
> Una conexión creada a través del AWS CLI o AWS CloudFormation está en `PENDING` estado de forma predeterminada. Después de crear una conexión con la CLI o CloudFormation, utilice la consola para editar la conexión y establecer su estado`AVAILABLE`.

Puede usar el AWS Command Line Interface (AWS CLI) para crear un host para las conexiones instaladas.

> **Note**
>
> Solo puede crear un host una vez por cuenta de GitHub Enterprise Server. Todas las conexiones a una cuenta específica de GitHub Enterprise Server utilizarán el mismo host.

Se utiliza un alojamiento para representar el punto de conexión de la infraestructura donde está instalado el proveedor de terceros. Tras completar la creación del host con la CLI, el host pasa a tener el estado Pendiente. A continuación, configure o registre el host para que pase a un estado Disponible. Una vez que el alojamiento esté disponible, complete los pasos para crear una conexión.

Para ello, utilice el comando create-host.

> **Important**
>
> Un host creado a través de AWS CLI está en `Pending` estado de forma predeterminada. Después de crear un host con la CLI, utilice la consola o la CLI para configurar el host de manera que su estado cambie a `Available`.

Creación de un host

1. Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows). Utilice el AWS CLI para ejecutar el create-host comando, especificando el `--name`--`provider-type`, y `--provider-endpoint` para la conexión. En este ejemplo, el nombre del proveedor de terceros es `GitHubEnterpriseServer` y el punto de conexión es `my-instance.dev`.

```
aws codestar-connections create-host --name MyHost --provider-type
 GitHubEnterpriseServer --provider-endpoint "https://my-instance.dev"
```

Si se ejecuta correctamente, este comando devuelve la información del nombre de recurso de Amazon (ARN) del alojamiento, que será similar a lo siguiente.

```
{
    "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
Host-28aef605"
}
```

Después de este paso, el alojamiento se encuentra en estado `PENDING`.

2. Utilice la consola para completar la configuración del alojamiento y que el estado del alojamiento cambie a `Available`.

Para crear una conexión a GitHub Enterprise Server

1. Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows). Utilice el AWS CLI para ejecutar el create-connection comando, especificando el `--host-arn` y `--connection-name` para la conexión.

```
aws codestar-connections create-connection --host-arn arn:aws:codestar-
connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
 MyConnection
```

Si se ejecuta correctamente, este comando devuelve la información del ARN de la conexión, que será similar a lo siguiente.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad"
}
```

2. Utilice la consola para configurar la conexión pendiente.
3. De forma predeterminada, la canalización detecta los cambios al enviar el código al repositorio de origen de conexión. Para configurar la configuración del desencadenador de canalización para la publicación manual o para las etiquetas de Git, realiza una de las siguientes acciones:

- Para configurar la configuración de los desencadenadores de canalización para que comience únicamente con una versión manual, añade la siguiente línea a la configuración:

```
"DetectChanges": "false",
```

- Para configurar los desencadenadores de canalización para filtrar con ellos, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código. Por ejemplo, lo siguiente se añade al nivel de canalización de la definición de JSON de canalización. En este ejemplo, `release-v0` y `release-v1` son las etiquetas de Git que se deben incluir y `release-v2` es la etiqueta de Git que se debe excluir.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "tags": {
                                "includes": [
                                    "release-v0", "release-v1"
                                ],
                                "excludes": [
                                    "release-v2"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
```

# GitLab.com conexiones

Las conexiones le permiten autorizar y establecer configuraciones que asocian a su proveedor externo con sus AWS recursos. Para asociar su repositorio de terceros como origen de su canalización, debe usar una conexión.

> **Note**
>
> En lugar de crear o usar una conexión existente en su cuenta, puede usar una conexión compartida con otra Cuenta de AWS. Consulte Uso de una conexión compartida con otra cuenta.

> **Note**
>
> Esta función no está disponible en las regiones Asia Pacífico (Hong Kong), Asia Pacífico (Hyderabad), Asia Pacífico (Yakarta), Asia Pacífico (Melbourne), Asia Pacífico (Osaka), África (Ciudad del Cabo), Oriente Medio (Baréin), Oriente Medio (Emiratos Árabes Unidos), Europa (España), Europa (Zúrich), Israel (Tel Aviv) o AWS GovCloud (). US-West Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline. Para ver consideraciones sobre esta acción en la región de Europa (Milán), consulte la nota que aparece en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

Para añadir una acción GitLab.com de origen CodePipeline, puedes elegir entre las siguientes opciones:

- Utilice el asistente de creación de canalizaciones de la CodePipeline consola o la página de edición de acciones para elegir la opción GitLabde proveedor. Consulte Cree una conexión a GitLab.com (consola) para añadir la acción. La consola le ayuda a crear un recurso de conexiones.
- Use la CLI para agregar la configuración de acción para la `CreateSourceConnection` acción con el proveedor `GitLab` de la siguiente manera:
  - Para crear sus recursos de conexiones, consulte Crear una conexión a GitLab.com (CLI) para crear un recurso de conexiones con la CLI.
  - Utilice el ejemplo `CreateSourceConnection` de configuración de acciones en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas para añadir la acción, como se muestra en Crear una canalización (CLI).

> **Note**
>
> También puede crear una conexión mediante la consola de Herramientas para desarrolladores, en Configuración. Consulte Crear una conexión.

> **Note**
>
> Al autorizar la instalación de esta conexión GitLab.com, concedes a nuestro servicio permisos para procesar tus datos accediendo a tu cuenta, y puedes revocar los permisos en cualquier momento desinstalando la aplicación.

Antes de empezar:

- Debe haber creado ya una cuenta con. GitLab.com

  > **Note**
  >
  > Las conexiones solo dan acceso a los repositorios que pertenecen a la cuenta que se utilizó para crear y autorizar la conexión.

  > **Note**
  >
  > Puede crear conexiones a un repositorio en el que tenga el rol de propietario y GitLab, a continuación, la conexión se puede utilizar con el repositorio con recursos como CodePipeline: En el caso de los repositorios en grupos, no es necesario que sea el propietario del grupo.

- Para especificar una fuente para la canalización, debe haber creado ya un repositorio en gitlab.com.

Temas

- Cree una conexión a GitLab.com (consola)
- Crear una conexión a GitLab.com (CLI)

# Cree una conexión a GitLab.com (consola)

Sigue estos pasos para usar la CodePipeline consola y añadir una acción de conexión para tu proyecto (repositorio) en GitLab.

Para crear o editar la canalización

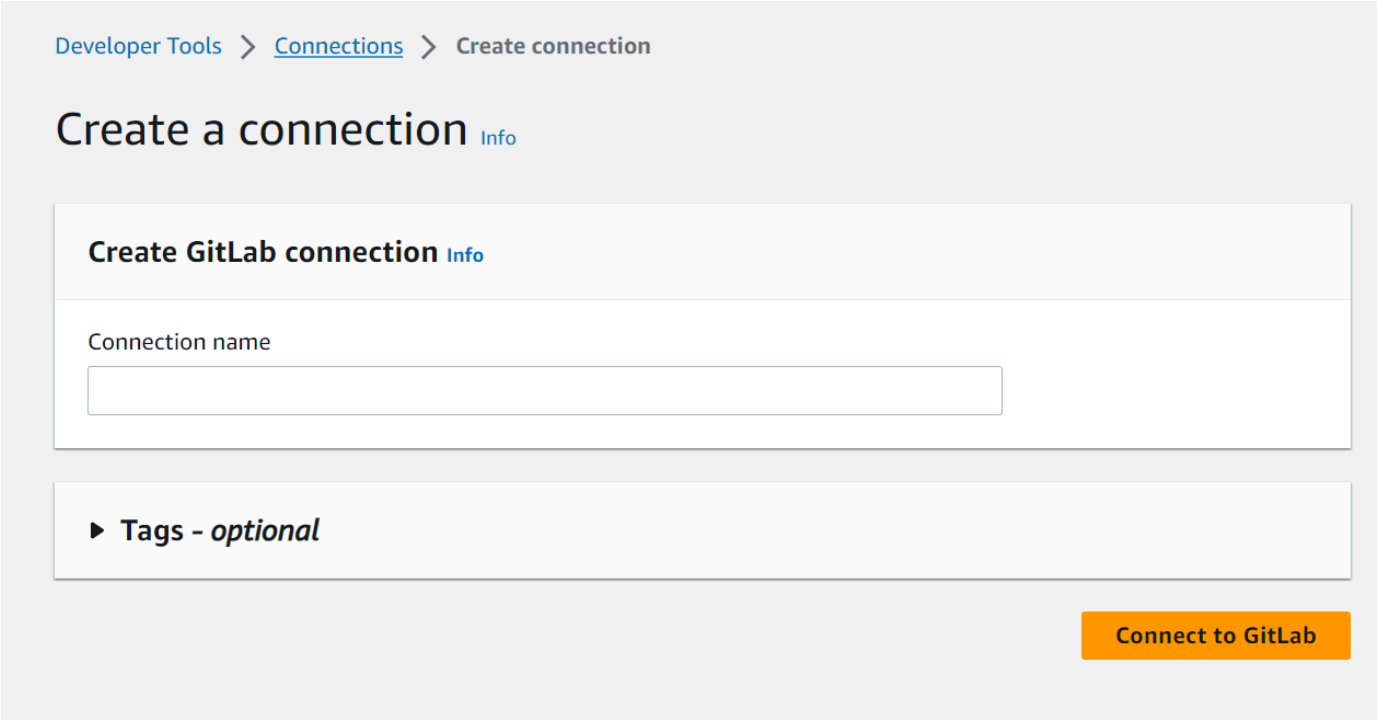
1. Inicia sesión en la CodePipeline consola.
2. Elija una de las siguientes opciones.

   - Elija Crear canalización. Siga los pasos de Crear una canalización para completar la primera pantalla y seleccione Siguiente. En la página de origen, en Proveedor de código fuente, selecciona GitLab.
   - Elija editar una canalización existente. Elija Editar y, a continuación, elija Editar etapa. Elija añadir o editar su acción de origen. En la página Editar acción, en Nombre de la acción, introduzca el nombre de la acción. En Proveedor de acción, seleccione GitLab.
3. Realice una de las siguientes acciones:

   - En Conexión, si aún no ha creado una conexión con su proveedor, elija Conectar a GitLab. Continúe con el paso 4 para crear la conexión.
   - En Conexión, si ya ha creado una conexión con su proveedor, seleccione la conexión. Continúe con el paso 9.

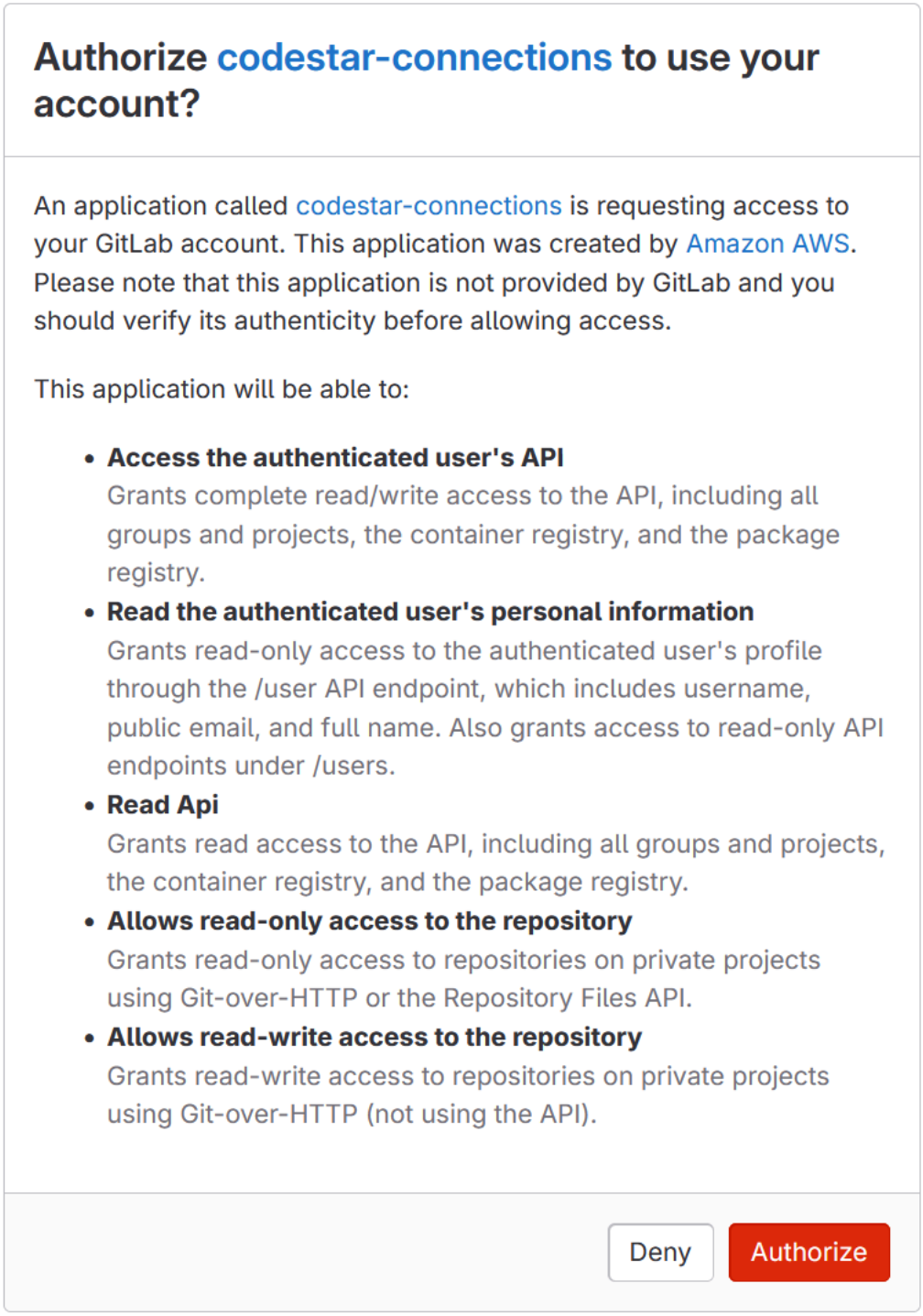
   > **Note**
   >
   > Si cierra la ventana emergente antes de crear una GitLab.com conexión, debe actualizar la página.

4. Para crear una conexión a un GitLab.com repositorio, en Seleccione un proveedor, elija GitLab. En Nombre de la conexión, introduzca el nombre de la conexión que desea crear. Selecciona Conectar a GitLab.




5. Cuando aparezca la página de inicio de GitLab.com sesión, inicia sesión con tus credenciales y, a continuación, selecciona Iniciar sesión.
6. Si es la primera vez que autorizas la conexión, aparecerá una página de autorización con un mensaje solicitando la autorización de la conexión para acceder a tu GitLab.com cuenta.

   Seleccione Autorizar.




7. El navegador vuelve a la página de la consola de conexiones. En Crear GitLab conexión, la nueva conexión se muestra en el nombre de la conexión.
8. Selecciona Conectar a GitLab.

   Volverá a la CodePipeline consola.

> **Note**
>
> Cuando la GitLab.com conexión se haya creado correctamente, aparecerá un aviso de éxito en la ventana principal.
> Si no ha iniciado sesión anteriormente GitLab en la máquina actual, tendrá que cerrar manualmente la ventana emergente.

9. En Nombre del repositorio, elija el nombre de su proyecto GitLab especificando la ruta del proyecto con el espacio de nombres. Por ejemplo, para un repositorio de grupo, introduzca el nombre del repositorio en el siguiente formato: `group-name/repository-name`. Para obtener más información sobre la ruta y el espacio de nombres, consulta el `path_with_namespace` campo de #get -single-project. https://docs.gitlab.com/ee/api/projects.html Para obtener más información sobre el espacio de nombres de, consulte. GitLab https://docs.gitlab.com/ee/user/namespace/

   > **Note**
   >
   > Para los grupos de GitLab, debe especificar manualmente la ruta del proyecto con el espacio de nombres. Por ejemplo, para un repositorio con el nombre `myrepo` en un grupo `mygroup`, introduzca lo siguiente: `mygroup/myrepo`. Puedes encontrar la ruta del proyecto con el espacio de nombres en la URL de. GitLab

10. En los activadores de Pipeline, puedes añadir activadores si tu acción es una AWS CodeConnections acción. Para configurar los desencadenadores de canalización y, de forma opcional, filtrar con desencadenadores, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código.
11. En Nombre de ramificación, elija la ramificación en la que desea que la canalización detecte los cambios de origen.

    > **Note**
    >
    > Si el nombre de la rama no se completa automáticamente, significa que no tienes acceso de propietario al repositorio. El nombre del proyecto no es válido o la conexión utilizada no tiene acceso al project/repository.

12. En Output artifact format (Formato del artefacto de salida), debe elegir el formato de los artefactos.

- Para almacenar los artefactos de salida de la GitLab.com acción mediante el método predeterminado, elija CodePipeline default. La acción accede a los archivos del GitLab.com repositorio y almacena los artefactos en un archivo ZIP en el almacén de artefactos de Pipeline.
- Para almacenar un archivo JSON que contiene una referencia URL al repositorio de manera que las acciones posteriores puedan ejecutar comandos Git directamente, elija Clonación completa. Esta opción solo la pueden utilizar las acciones posteriores de CodeBuild .

  Si eliges esta opción, tendrás que actualizar los permisos de tu rol de servicio del CodeBuild proyecto, tal y como se muestra en la siguiente. Añade CodeBuild GitClone permisos para las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com Para ver un tutorial que muestra cómo utilizar la opción Clonación completa, consulte Tutorial: Utilice un clon completo con una fuente de GitHub canalización.

13. Seleccione guardar la acción de origen y continuar.

## Crear una conexión a GitLab.com (CLI)

Puede usar AWS Command Line Interface (AWS CLI) para crear una conexión.

Para ello, utilice el comando create-connection.

> **Important**
>
> Una conexión creada a través del AWS CLI o AWS CloudFormation está en `PENDING` estado de forma predeterminada. Después de crear una conexión con la CLI o CloudFormation, utilice la consola para editar la conexión y establecer su estado`AVAILABLE`.

Creación de una conexión

1. Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows). Utilice el AWS CLI para ejecutar el create-connection comando, especificando el `--provider-type` y `--connection-name` para la conexión. En este ejemplo, el nombre del proveedor de terceros es `GitLab` y el nombre especificado para la conexión es `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type GitLab --connection-name
    MyConnection
   ```

Si se ejecuta correctamente, este comando devuelve la información del ARN de la conexión, que será similar a lo siguiente.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Utilice la consola para completar la conexión. Para obtener más información, consulte Actualización de una conexión pendiente.
3. De forma predeterminada, la canalización detecta los cambios al enviar el código al repositorio de origen de conexión. Para configurar la configuración del desencadenador de canalización para la publicación manual o para las etiquetas de Git, realiza una de las siguientes acciones:

   - Para configurar la configuración de los desencadenadores de canalización para que comience únicamente con una versión manual, añade la siguiente línea a la configuración:

     ```
     "DetectChanges": "false",
     ```

   - Para configurar los desencadenadores de canalización para filtrar con ellos, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código. Por ejemplo, lo siguiente se añade al nivel de canalización de la definición de JSON de canalización. En este ejemplo, `release-v0` y `release-v1` son las etiquetas de Git que se deben incluir y `release-v2` es la etiqueta de Git que se debe excluir.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
     ```

```
                        }
                    }
                ]
            }
        }
    ]
```

# Conexiones para GitLab autogestión

Las conexiones le permiten autorizar y establecer configuraciones que asocian a su proveedor externo con sus AWS recursos. Para asociar su repositorio de terceros como origen de su canalización, debe usar una conexión.

> **Note**
>
> En lugar de crear o usar una conexión existente en su cuenta, puede usar una conexión compartida con otra Cuenta de AWS. Consulte Uso de una conexión compartida con otra cuenta.

> **Note**
>
> Esta función no está disponible en las regiones Asia Pacífico (Hong Kong), Asia Pacífico (Hyderabad), Asia Pacífico (Yakarta), Asia Pacífico (Melbourne), Asia Pacífico (Osaka), África (Ciudad del Cabo), Oriente Medio (Baréin), Oriente Medio (Emiratos Árabes Unidos), Europa (España), Europa (Zúrich), Israel (Tel Aviv) o AWS GovCloud (). US-West Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline. Para ver consideraciones sobre esta acción en la región de Europa (Milán), consulte la nota que aparece en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

Para añadir una acción de origen GitLab autogestionada CodePipeline, puedes elegir entre las siguientes opciones:

- Usa el asistente de creación de canalizaciones de la CodePipeline consola o la página de edición de acciones para elegir la opción de proveedor GitLabautogestionado. Consulte Cree una

conexión GitLab autogestionada (consola) para añadir la acción. La consola le ayuda a crear un recurso de host y un recurso de conexiones.

- Use la CLI para agregar la configuración de acciones para la acción `CreateSourceConnection` con el proveedor `GitLabSelfManaged` y crear sus recursos:
  - Para crear sus recursos de conexiones, consulte Cree un host y una conexión a la red GitLab autogestionada (CLI) para crear un recurso de host y un recurso de conexiones con la CLI.
  - Utilice el ejemplo `CreateSourceConnection` de configuración de acciones en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas para añadir la acción, como se muestra en Crear una canalización (CLI).

> **Note**
>
> También puede crear una conexión mediante la consola de Herramientas para desarrolladores, en Configuración. Consulte Crear una conexión.

Antes de empezar:

- Debe haber creado ya una cuenta GitLab y disponer de GitLab Enterprise Edition o GitLab Community Edition con una instalación autogestionada. Para obtener más información, consulte https://docs.gitlab.com/ee/subscriptions/self_managed/.

> **Note**
>
> Las conexiones solo dan acceso a la cuenta que se utilizó para crear y autorizar la conexión.

> **Note**
>
> Puede crear conexiones a un repositorio en el que tenga el rol de propietario y GitLab, a continuación, la conexión se puede utilizar con recursos como CodePipeline: En el caso de los repositorios en grupos, no es necesario que sea el propietario del grupo.

- Debe haber creado ya un token de acceso GitLab personal (PAT) con ese `api` alcance. Si el modo de administración está habilitado en la GitLab instancia, la PAT también debe incluir el `admin_mode` ámbito. Para obtener más información, consulte los ámbitos de los identificadores de acceso personales y de los identificadores de acceso en el GitLab sitio web. Debe ser administrador para crear y utilizar el PAT.

  > **Note**
  >
  > Su PAT se utiliza para autorizar el host y las conexiones no la almacenan ni lo utilizan de ningún otro modo. Para configurar un host, puede crear un PAT temporal y, después de configurar el host, puede eliminarlo.

- Puede elegir configurar el host con antelación. Puede configurar un host con y sin VPC. Para obtener detalles sobre la configuración de la VPC e información adicional sobre la creación de un host, consulte Crear un host.

Temas


- Cree una conexión GitLab autogestionada (consola)
- Cree un host y una conexión a la red GitLab autogestionada (CLI)


## Cree una conexión GitLab autogestionada (consola)

Sigue estos pasos para usar la CodePipeline consola y añadir una acción de conexión a tu repositorio GitLab autogestionado.

> **Note**
>
> GitLab Las conexiones autogestionadas solo proporcionan acceso a los repositorios propiedad de la cuenta GitLab autogestionada que se utilizó para crear la conexión.

Antes de empezar

Para que una conexión de host se GitLab administre automáticamente, debe haber completado los pasos para crear un recurso de host para su conexión. Consulte Administrar hosts para conexiones.

## Paso 1: Crear o editar la canalización

Para crear o editar la canalización

1. Inicie sesión en la CodePipeline consola.
2. Elija una de las siguientes opciones.
   - Elija Crear canalización. Siga los pasos de Crear una canalización para completar la primera pantalla y seleccione Siguiente. En la página Fuente, en Proveedor de fuentes, selecciona GitLab Autogestionable.
   - Elija editar una canalización existente. Elija Editar y, a continuación, elija Editar etapa. Elija añadir o editar su acción de origen. En la página Editar acción, en Nombre de la acción, introduzca el nombre de la acción. En el proveedor de acciones, elige GitLab autogestionado.
3. Realice una de las siguientes acciones:
   - En Conexión, si aún no ha creado una conexión con su proveedor, elija Conectar para GitLab autogestionarse. Continúe con el paso 2: cree una conexión para GitLab autogestionarse.
   - En Conexión, si ya ha creado una conexión con su proveedor, selecciónela y, a continuación, continúe con el paso 3: Guardar la acción de origen GitLab autogestionada.

## Paso 2: Crea una conexión a la red autogestionada GitLab

Una vez que haya decidido crear la conexión, aparecerá la página  GitLabConectarse a la autogestión.

Para conectarse a la red autogestionada GitLab

1. En Connection name (Nombre de la conexión), ingrese el nombre para la conexión.
2. En URL, ingrese el punto de conexión para el servidor.

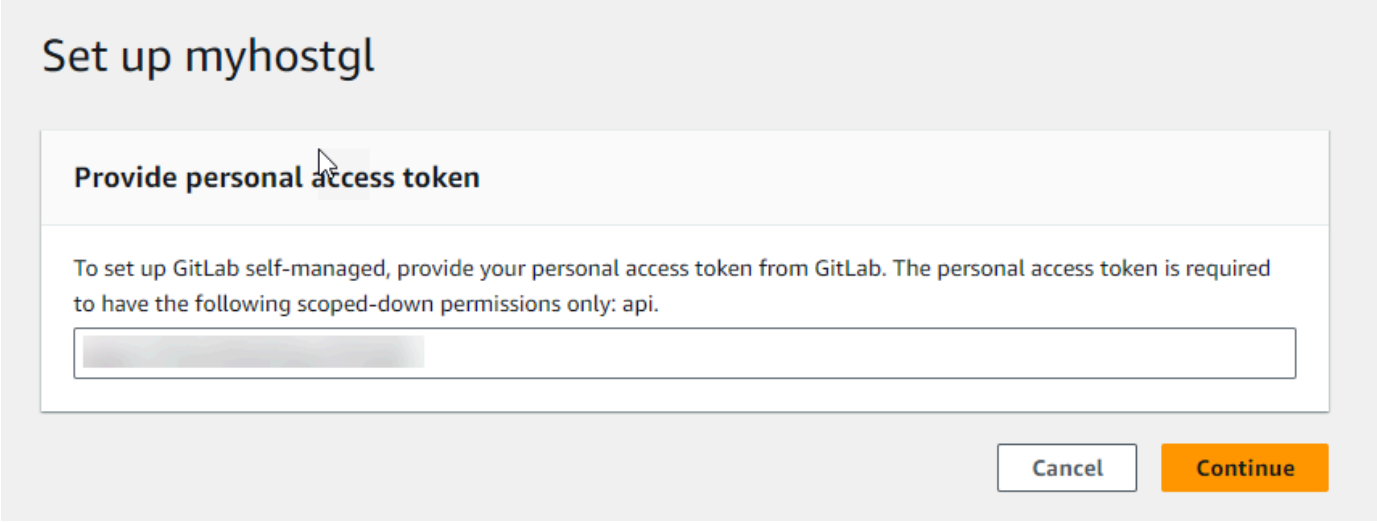
   > **Note**
   >
   > Si la dirección URL proporcionada ya se utilizó para configurar un host para una conexión, se le pedirá que elija el ARN del recurso de host que se creó previamente para ese punto de conexión.

3. Si lanzó su servidor en una Amazon VPC y desea conectarse a su VPC, elija Utilizar una VPC y complete la siguiente información para la VPC.

4. Elige Conectar para GitLab autogestionarse. La conexión creada se muestra con un estado Pendiente. Se crea un recurso de alojamiento para la conexión con la información del servidor que usted proporcionó. Se utiliza la URL para el nombre del alojamiento.
5. Elija Update pending connection (Actualizar conexión pendiente).
6. Si se abre una página con un mensaje de redireccionamiento para confirmar que desea continuar con el proveedor, seleccione Continuar. Introduzca la autorización del proveedor.
7. Aparece una *host_name* página de configuración. En Proporcionar un token de acceso personal, indique el `api` alcance a su GitLab PAT. Si el modo de administración está habilitado en la GitLab instancia, la PAT también debe incluir el `admin_mode` ámbito. Para obtener más información, consulta los ámbitos de los tokens de acceso en el GitLab sitio web.

   > **Note**
   >
   > Solo un administrador puede crear y usar el PAT.

   Elija Continuar.

   
   
8. La página de conexión muestra la conexión creada en un estado Disponible.

## Paso 3: Guarda tu acción de origen GitLab autogestionada

Siga estos pasos del asistente o de la página de Editar acción para guardar la acción de origen con la información de conexión.

Para completar y guardar la acción de origen con la conexión

1. En Repository name (Nombre del repositorio), elija el nombre del repositorio de terceros.
2. En los activadores de Pipeline, puedes añadir activadores si tu acción es una AWS CodeConnections acción. Para configurar los desencadenadores de canalización y, de forma opcional, filtrar con desencadenadores, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código.
3. En Output artifact format (Formato del artefacto de salida), debe elegir el formato de los artefactos.
   - Para almacenar los artefactos de salida de la acción GitLab autogestionada mediante el método predeterminado, selecciona el CodePipelinepredeterminado. La acción obtiene acceso a los archivos del repositorio y almacena los artefactos en un archivo ZIP en el almacén de artefactos de canalización.
   - Para almacenar un archivo JSON que contiene una referencia URL al repositorio de manera que las acciones posteriores puedan ejecutar comandos Git directamente, elija Clonación completa. Esta opción solo la pueden utilizar las acciones CodeBuild posteriores.
4. Seleccione Siguiente en el asistente o Guardar en la página Editar acción.

## Cree un host y una conexión a la red GitLab autogestionada (CLI)

Puede usar el AWS Command Line Interface (AWS CLI) para crear una conexión.

Para ello, utilice el comando create-connection.

> **Important**
>
> Una conexión creada a través del AWS CLI o AWS CloudFormation está en `PENDING` estado de forma predeterminada. Después de crear una conexión con la CLI o CloudFormation, utilice la consola para editar la conexión y establecer su estado`AVAILABLE`.

Puede usar el AWS Command Line Interface (AWS CLI) para crear un host para las conexiones instaladas.

Se utiliza un alojamiento para representar el punto de conexión de la infraestructura donde está instalado el proveedor de terceros. Tras completar la creación del host con la CLI, el host pasa a

tener el estado Pendiente. A continuación, configure o registre el host para que pase a un estado Disponible. Una vez que el alojamiento esté disponible, complete los pasos para crear una conexión.

Para ello, utilice el comando create-host.

> **Important**
>
> Un host creado a través de AWS CLI está en `Pending` estado de forma predeterminada. Después de crear un host con la CLI, utilice la consola o la CLI para configurar el host de manera que su estado cambie a `Available`.

Creación de un host

1. Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows). Utilice el AWS CLI para ejecutar el create-host comando, especificando el `--name` `--provider-type`, y `--provider-endpoint` para la conexión. En este ejemplo, el nombre del proveedor de terceros es `GitLabSelfManaged` y el punto de conexión es `my-instance.dev`.

   ```
   aws codestar-connections create-host --name MyHost --provider-type
    GitLabSelfManaged --provider-endpoint "https://my-instance.dev"
   ```

   Si se ejecuta correctamente, este comando devuelve la información del nombre de recurso de Amazon (ARN) del alojamiento, que será similar a lo siguiente.

   ```
   {
       "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
   Host-28aef605"
   }
   ```

   Después de este paso, el alojamiento se encuentra en estado `PENDING`.

2. Utilice la consola para completar la configuración del alojamiento y que el estado del alojamiento cambie a `Available`.

Para crear una conexión GitLab autogestionada

1. Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows). Utilice el AWS CLI para ejecutar el create-connection comando, especificando el `--host-arn` y `--connection-name` para la conexión.

   ```
   aws codestar-connections create-connection --host-arn arn:aws:codestar-
   connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
    MyConnection
   ```

   Si se ejecuta correctamente, este comando devuelve la información del ARN de la conexión, que será similar a lo siguiente.

   ```
   {
       "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
   aEXAMPLE-8aad"
   }
   ```

2. Utilice la consola para configurar la conexión pendiente.
3. De forma predeterminada, la canalización detecta los cambios al enviar el código al repositorio de origen de conexión. Para configurar la configuración del desencadenador de canalización para la publicación manual o para las etiquetas de Git, realiza una de las siguientes acciones:

   - Para configurar la configuración de los desencadenadores de canalización para que comience únicamente con una versión manual, añade la siguiente línea a la configuración:

     ```
     "DetectChanges": "false",
     ```

   - Para configurar los desencadenadores de canalización para filtrar con ellos, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código. Por ejemplo, lo siguiente se añade al nivel de canalización de la definición de JSON de canalización. En este ejemplo, `release-v0` y `release-v1` son las etiquetas de Git que se deben incluir y `release-v2` es la etiqueta de Git que se debe excluir.

     ```
     "triggers": [
                 {
                     "providerType": "CodeStarSourceConnection",
                     "gitConfiguration": {
                         "sourceActionName": "Source",
     ```

```
                "push": [
                    {
                        "tags": {
                            "includes": [
                                "release-v0", "release-v1"
                            ],
                            "excludes": [
                                "release-v2"
                            ]
                        }
                    }
                ]
            }
        }
    ]
```

## Uso de una conexión compartida con otra cuenta

En AWS RAM, puede crear y administrar una conexión compartida. Esto permite compartir conexiones entre Cuentas de AWS para acceder a repositorios de terceros. Esto permite utilizar una sola conexión en las canalizaciones de CodePipeline en todas las cuentas y, al mismo tiempo, reduce la necesidad de que los usuarios gestionen y administren conexiones independientes en cada cuenta.

Para usar conexiones compartidas en CodePipeline, haga lo siguiente.

- Cree una conexión mediante la consola de Herramientas para desarrolladores, en Configuración. Consulte Crear una conexión.
- Configure el recurso compartido mediante AWS RAM. Consulte Uso compartido de conexiones con Cuentas de AWS.
- Cuando se utiliza el asistente Crear canalización o la página Editar acción de la consola CodePipeline para elegir el proveedor de conexión, como la opción de proveedor de Bitbucket, se puede elegir la conexión que se ha compartido con la cuenta de destino.

# Automatización del inicio de las canalizaciones mediante desencadenadores y filtrado

Los desencadenadores permiten configurar la canalización para que se inicie con un tipo de evento concreto o filtrado, por ejemplo, cuando se detecta un cambio en una ramificación específica o en una solicitud de extracción. Los activadores se pueden configurar para las acciones de origen con conexiones que utilizan la `CodeStarSourceConnection` acción en CodePipeline, por ejemplo GitHub, Bitbucket y GitLab. Para obtener más información acerca de las acciones de origen que utilizan conexiones, consulte Agrega proveedores de fuentes de terceros a las canalizaciones mediante AWS CodeConnections.

Las acciones de origen, como CodeCommit S3, utilizan la detección automática de cambios para iniciar las canalizaciones cuando se realiza un cambio. Para obtener más información, consulte CodeCommit acciones de origen y EventBridge.

Puede especificar desencadenadores mediante la consola o la CLI.

Los tipos de filtros se especifican de la siguiente manera:

- Sin filtro

  Esta configuración de desencadenador inicia la canalización con cualquier inserción a la ramificación predeterminada especificada como parte de la configuración de la acción.
- Especificar filtro

  Añada un filtro que inicie la canalización con un filtro específico, por ejemplo, en los nombres de las ramificaciones de una inserción de código, y obtenga la confirmación exacta. Esto también configura la canalización para que no se inicie automáticamente cuando se produzcan cambios.
  - Inserción
    - Las combinaciones de filtros válidas son las siguientes:
      - Etiquetas de Git

        Include o exclude
      - sucursales

        Include o exclude
      - ramas + rutas de archivos

Include o exclude

- Solicitud de extracción
  - Las combinaciones de filtros válidas son las siguientes:
    - ramas

      Include o exclude
    - ramas + rutas de archivos

      Include o exclude
- No detectar cambios

  Este tipo de filtro no añade ningún desencadenador y la canalización no se inicia automáticamente aunque se produzcan cambios.

La siguiente tabla proporciona opciones de filtro válidas para cada tipo de evento. En la tabla también se muestran qué configuraciones de desencadenador son verdaderas o falsas de forma predeterminada para la detección automática de cambios en la configuración de la acción.

| Configuración de desencadenadores | Tipo de evento | Opciones de filtro | Detección de cambios |
|---|---|---|---|
| Agregar un desencadenador, sin filtro | none | none | true |
| Agregar un desencadenador: filtrar en la inserción de código | evento de inserción | Etiquetas de Git, ramificaciones, rutas de archivo | false |
| Agregar un desencadenador: filtrar las solicitudes de inserción | solicitudes de extracción | ramificaciones, rutas de archivo | false |

| Configuración de desencadenadores | Tipo de evento | Opciones de filtro | Detección de cambios |
|---|---|---|---|
| Sin desencadenador: no detectar | none | none | false |

> **Note**
>
> Este tipo de desencadenador utiliza la detección automática de cambios (como el tipo de desencadenador `Webhook`). Los proveedores de acciones fuente que utilizan este tipo de activador son conexiones configuradas para la inserción de código (Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com y GitLab autogestionadas).

Para ver las definiciones de los campos y más referencias sobre los desencadenadores, consulte

Para obtener una lista de las definiciones de campos de la estructura JSON, consulte `desencadenadores`.

Para el filtrado, se admiten patrones de expresiones regulares en formato glob, tal y como se detalla en Trabajar con patrones glob en la sintaxis.

> **Note**
>
> En algunos casos, en el caso de las canalizaciones con desencadenadores que se filtran en rutas de archivo, es posible que una canalización no se inicie al crear por primera vez una ramificación con un filtro de ruta de archivo. Para obtener más información, consulte Es posible que las canalizaciones con conexiones que utilicen el filtrado de desencadenadores por rutas de archivo no se inicien al crear la ramificación.

## Consideraciones sobre los filtros de desencadenadores

Las siguientes consideraciones se aplican cuando se utilizan desencadenadores.

- No puede agregar más de un desencadenador por acción de origen.

- Puede agregar varios tipos de filtros a un desencadenador. Para ver un ejemplo, consulta 4: un desencadenador con dos tipos de filtros de inserción con conflictos de inclusiones y exclusiones.
- En el caso de un desencadenador con filtros de ramificaciones y rutas de archivo, al insertar la ramificación por primera vez, la canalización no se ejecutará debido a que no se podrá acceder a la lista de archivos modificados para la ramificación recién creada.
- La fusión de una solicitud de extracción podría desencadenar dos ejecuciones de canalización en los casos en los que las configuraciones de los desencadenadores de solicitudes de inserción (filtro de ramificaciones) y de extracción (filtro de ramificaciones) se cruzasen entre sí.
- Si un filtro desencadena la canalización con eventos de solicitud de extracción, para un tipo de evento de solicitud de extracción cerrada, es posible que el proveedor de repositorios externo de su conexión tenga un estado diferente para un evento de fusión. Por ejemplo, en Bitbucket, el evento de Git de una fusión no es un evento de cierre de solicitud de extracción. Sin embargo GitHub, en el caso de fusionar una solicitud de extracción, se cierra el proceso. Para obtener más información, consulte Eventos de solicitud de extracción para desencadenadores por proveedor.
- Cuando varias acciones de origen de una canalización hacen referencia a diferentes ramas del mismo repositorio a través de una conexión, solo una rama activa la canalización de forma fiable. La suscripción a webhook de la conexión se registra para la combinación de canalización y repositorio, no para cada sucursal. Como solución alternativa, usa una canalización independiente para cada rama.

# Eventos de solicitud de extracción para desencadenadores por proveedor

La siguiente tabla proporciona un resumen de los eventos de Git (por ejemplo, para el cierre de solicitudes de extracción) que dan como resultado tipos de eventos de solicitud de extracción, por proveedor.

| | Proveedor de repositorios de la conexión | | | |
|---|---|---|---|---|
| Evento PR para el desencadenador | Bitbucket | GitHub | GHES | GitLab |
| Abrir: esta opción | La creación de una solicitud de | La creación de una solicitud de | La creación de una solicitud de | La creación de una solicitud de |

| | Proveedor de repositorios de la conexión | | | |
|---|---|---|---|---|
| Evento PR para el desencadenador | Bitbucket | GitHub | GHES | GitLab |
| desencadena la canalización cuando se crea una solicitud de extracción para la ruta de la rama o el archivo. | extracción da como resultado un evento de Git abierto. | extracción da como resultado un evento de Git abierto. | extracción da como resultado un evento de Git abierto. | extracción da como resultado un evento de Git abierto. |
| Actualizar: esta opción desencadena la canalización cuando se publica una revisión de una solicitud de extracción para la ruta de la ramificación o el archivo. | La publicación de una actualización da como resultado un evento de Git actualizado. | La publicación de una actualización da como resultado un evento de Git actualizado. | La publicación de una actualización da como resultado un evento de Git actualizado. | La publicación de una actualización da como resultado un evento de Git actualizado. |

| | Proveedor de repositorios de la conexión | | | |
|---|---|---|---|---|
| Evento PR para el desencadenador | Bitbucket | GitHub | GHES | GitLab |
| Cerrado: esta opción activa la canalización cuando se cierra una solicitud de extracción para la branch/file ruta. | La fusión de una solicitud de extracción en Bitbucket produce un evento de Git cerrado. Importante: Si se cierra manualmente una solicitud de extracción en Bitbucket sin fusión, no se produce un evento de Git cerrado. | Al fusionar o cerrar manualmente una solicitud de extracción, se produce un evento de Git cerrado. | Al fusionar o cerrar manualmente una solicitud de extracción, se produce un evento de Git cerrado. | Al fusionar o cerrar manualmente una solicitud de extracción, se produce un evento de Git cerrado. |

# Ejemplos de filtros de desencadenador

En el caso de una configuración de Git con filtros para los tipos de eventos de solicitudes de inserción y extracción, los filtros especificados pueden entrar en conflicto entre sí. A continuación, se muestran ejemplos de combinaciones de filtros válidas para eventos de solicitudes de inserción y extracción. Un desencadenador puede contener varios tipos de filtros, como dos tipos de filtros de inserción en la configuración del desencadenador, y los tipos de filtro de solicitud de inserción y extracción utilizarán una operación OR entre ellos, lo que significa que cualquier coincidencia iniciará la canalización. Del mismo modo, cada tipo de filtro puede incluir varios filtros, como filePaths y ramificaciones; dichos filtros utilizarán una operación AND, lo que significa que solo una coincidencia completa iniciará la canalización. Cada tipo de filtro puede contener inclusiones y exclusiones, donde las exclusiones tienen prioridad sobre las inclusiones. Si la ruta de una rama o un archivo coincide

con un patrón de exclusión, no se activará la canalización, aunque también coincida con un patrón de inclusión. Cuando una confirmación cambia varios archivos, cada uno de ellos se evalúa de forma independiente con respecto al filtro; si se aprueba algún archivo modificado (coincide con una inclusión y no coincide con una exclusión), se iniciará la canalización. Los nombres incluidos en la opción de inclusión/exclusión, como los nombres de las ramificaciones, utilizan una operación OR. La siguiente lista resume las operaciones de cada parte del objeto de configuración de Git.

Para obtener una lista de las definiciones de campos de la estructura JSON y una referencia detallada de las inclusiones y exclusiones, consulte `desencadenadores`.

Example 1: un tipo de filtro con filtros para las ramificaciones y las rutas de archivo (operación AND)

En el caso de tipos de filtro únicos, como el de solicitud de extracción, se pueden combinar los filtros; dichos filtros utilizarán una operación AND, lo que significa que solo una coincidencia completa iniciará la canalización. En el siguiente ejemplo se muestra una configuración de Git para un tipo de evento de inserción con dos filtros diferentes (`filePaths` y `branches`). En el siguiente ejemplo, `filePaths` utilizará la operación AND con `branches`:

```
{
  "filePaths": {
    "includes": ["common/**/*.js"]
  },
  "branches": {
    "includes": ["feature/**"]
  }
}
```

Con la configuración de Git anterior, este ejemplo muestra un evento que iniciará la ejecución de la canalización, ya que la operación AND se realiza correctamente. En otras palabras, se incluye la ruta del archivo `common/app.js` para el filtro, que inicia la canalización como AND aunque la ramificación `refs/heads/feature/triggers` especificada no haya tenido ningún impacto.

```
{
  "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "common/app.js"
      ]
```

```
      ...
    }
  ]
}
```

En este ejemplo, se muestra un evento para un desencadenador con la configuración anterior que no iniciará la ejecución de la canalización debido a que se puede filtrar por ramificación pero no por ruta.

```
{
   "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "src/Main.java"
      ]
      ...
    }
  ]
}
```

Example 2: Las exclusiones tienen prioridad sobre las inclusiones

Dentro de un único filtro, las exclusiones tienen prioridad sobre las inclusiones. El siguiente ejemplo muestra una configuración de Git con un único filtro (`branches`) dentro del objeto de configuración. Esto significa que si una rama coincide con un patrón de exclusión (`feature-branch`en el ejemplo), la canalización no se activará, aunque también coincida con un patrón de inclusión. Si el patrón de inclusión coincide y el patrón de no exclusión coincide, como en el caso de la `main` rama, se activará la canalización.

En el siguiente ejemplo de JSON:

- La inserción de una confirmación para la ramificación `main` desencadenará la canalización.
- La inserción de una confirmación para la ramificación `feature-branch` no desencadenará la canalización.

```
{
  "branches": {
      "Includes": [
```

```
            "main"
        ],
        "Excludes": [
            "feature-branch"
        ]
    }
```

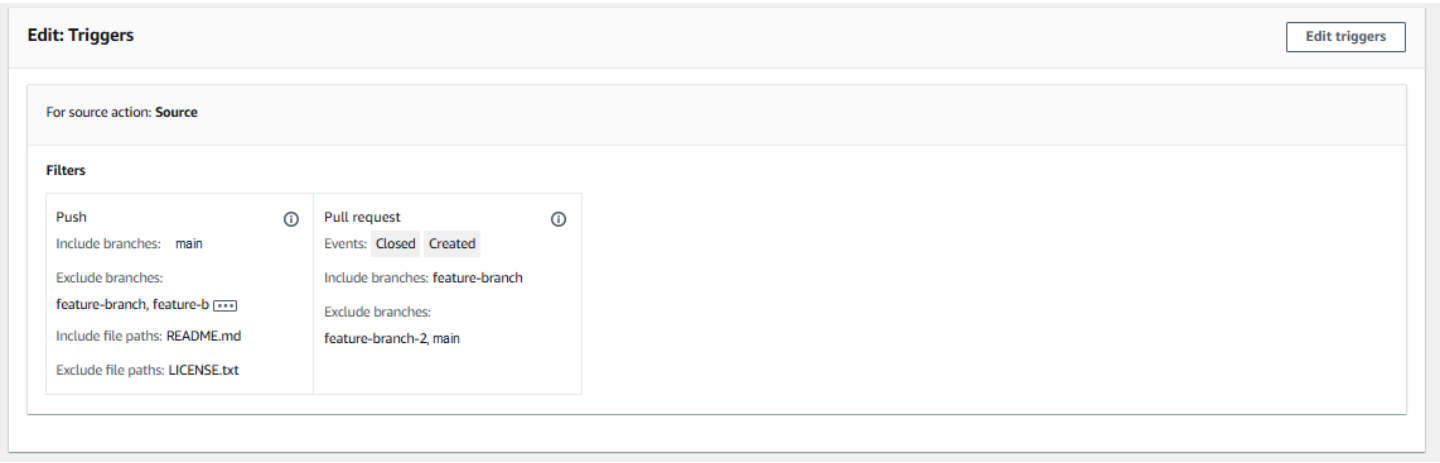
Example 3: Un disparador con tipos de filtros de solicitud de inserción y extracción (operación OR), filtros para rutas de archivos y ramas (operación AND) y includes/excludes (excluye tener prioridad)

Los objetos de configuración de los desencadenadores, como un desencadenador que contenga un tipo de evento de inserción y un tipo de evento de solicitud de extracción, utilizan una operación OR entre los dos tipos de eventos. En el siguiente ejemplo, se muestra una configuración de desencadenador con un tipo de evento de inserción con la ramificación `main` incluida y un tipo de evento de solicitud de extracción con la misma ramificación `main` excluida. Además, el tipo de evento de inserción tiene una ruta de archivo `LICENSE.txt` excluida y una ruta de archivo `README.MD` incluida. Para el segundo tipo de evento, una solicitud de extracción que esté `Closed` o esté `Created` en la `feature-branch` sucursal (incluida) inicia la canalización, y la canalización no se inicia al crear o cerrar una solicitud de extracción en la sucursal `feature-branch-2` o `main` sucursales (excluida). Dentro de cada tipo de evento, las exclusiones tienen prioridad sobre las inclusiones. Por ejemplo, en el caso de un evento de solicitud de extracción en la `feature-branch` sucursal (incluido en la solicitud de extracción), el tipo de evento de inserción excluye la `feature-branch` rama, por lo que la inserción no activará la canalización.

En el ejemplo siguiente:

- La inserción de una confirmación en la ramificación `main` (incluida) para la ruta del archivo `README.MD` (incluida), se desencadenará la canalización.
- En la ramificación `feature-branch` (excluida), la inserción de una confirmación no desencadenará la canalización.
- En la ramificación incluida, la edición de la ruta del archivo `README.MD` (incluida) desencadena la canalización.
- En la rama incluida, editar solo la ruta del `LICENSE.TXT` archivo (excluida) no activa la canalización. Sin embargo, si la misma confirmación también cambia `README.MD` (incluida), la canalización se activará porque cada archivo se evalúa de forma independiente.
- En la `feature-branch` sucursal, al cerrar una solicitud de extracción, se activará la canalización, ya que `feature-branch` se incluye para el tipo de evento de solicitud de extracción y para el tipo de evento CLOSED matches.

En la siguiente imagen se muestra la configuración.




A continuación, se muestra el código JSON de ejemplo de la configuración.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "branches": {
                                "includes": [
                                    "main"
                                ],
                                "excludes": [
                                    "feature-branch",
                                    "feature-branch-2"
                                ]
                            },
                            "filePaths": {
                                "includes": [
                                    "README.md"
                                ],
                                "excludes": [
                                    "LICENSE.txt"
                                ]
                            }
                        }
                    ],
                    "pullRequest": [
                        {
```

```
                        "events": [
                            "CLOSED",
                            "OPEN"
                        ],
                        "branches": {
                            "includes": [
                                "feature-branch"
                            ],
                            "excludes": [
                                "feature-branch-2",
                                "main"
                            ]
                        }
                    }
                ]
            }
        }
    ]
},
```

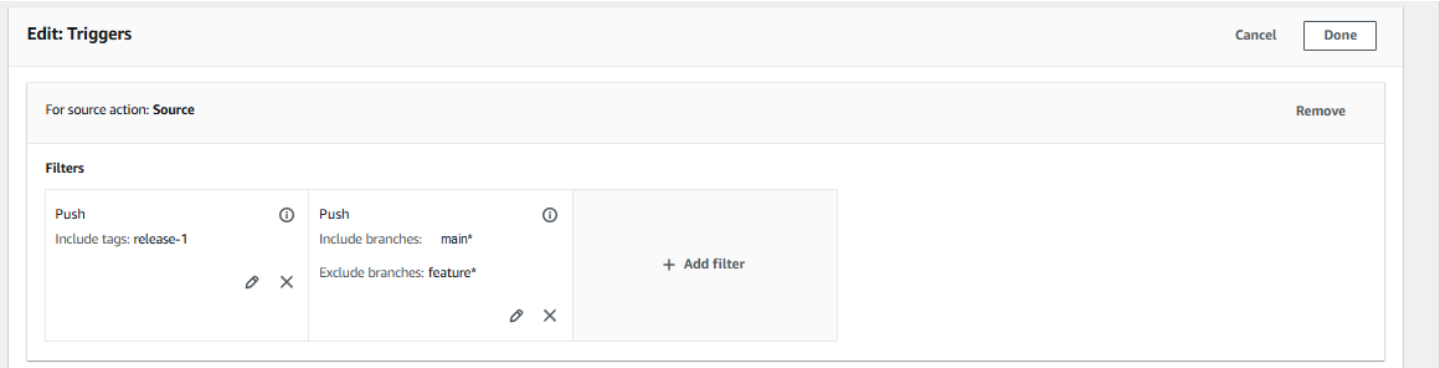
Example 4: un desencadenador con dos tipos de filtros de inserción con conflictos de inclusiones y exclusiones

En la siguiente imagen, se muestra un tipo de filtro de inserción que especifica el filtro en la etiqueta `release-1` (incluida). Se ha añadido un segundo tipo de filtro de inserción que especifica que se debe filtrar por la ramificación `main` (incluida) y no iniciarse si se produce una inserción para las ramificaciones `feature*` (excluidas).

Para el ejemplo siguiente:

- Al pulsar una tecla de apertura de la `feature-branch` rama `release-1` (incluida en el caso del primer filtro de inserción) (excluida la del segundo filtro de inserción), se activará la canalización, ya que los dos tipos de filtros de inserción utilizan una operación OR entre ellos y el primer filtro de inserción (etiqueta`release-1`) coincide. `feature*`
- Si se inserta una versión desde la ramificación `main` (incluida para el segundo filtro de inserción), se iniciará la canalización.

En el siguiente ejemplo de la página de edición, se muestran los dos tipos de filtros de inserción y su configuración para inclusiones y exclusiones.




A continuación, se muestra el código JSON de ejemplo de la configuración.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "tags": {
                                "includes": [
                                    "release-1"
                                ]
                            }
                        },
                        {
                            "branches": {
                                "includes": [
                                    "main*"
                                ],
                                "excludes": [
                                    "feature*"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    },
```

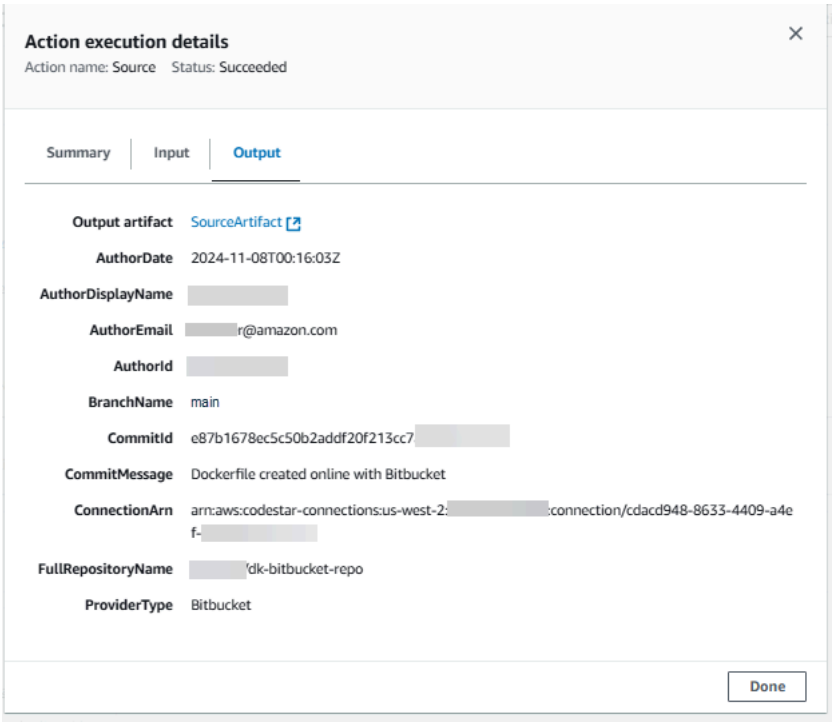
Example 5: El disparador está configurado mientras que la configuración de acción predeterminada BranchName se utiliza para un inicio manual

El campo predeterminado de configuración de acciones `BranchName` define la ramificación que se usará cuando la canalización se inicie manualmente, mientras que se pueden usar desencadenadores con filtros para cualquier ramificación o ramificaciones que especifique.

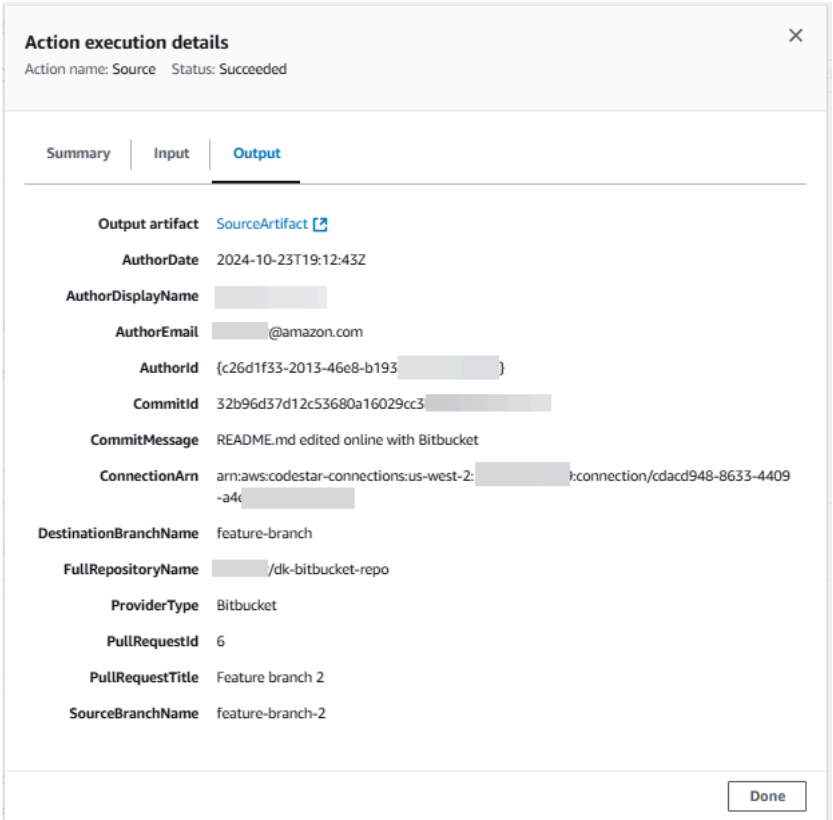
A continuación, se muestra el código JSON de ejemplo para la configuración de acciones con el campo `BranchName`.

```
{
                "name": "Source",
                "actions": [
                    {
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "provider": "CodeStarSourceConnection",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "BranchName": "main",
                            "ConnectionArn": "ARN",
                            "DetectChanges": "false",
                            "FullRepositoryId": "owner-name/my-bitbucket-repo",
                            "OutputArtifactFormat": "CODE_ZIP"
                        },
                        "outputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "inputArtifacts": [],
                        "region": "us-west-2",
                        "namespace": "SourceVariables"
                    }
                ],
```

El siguiente ejemplo de salida de acción muestra que se utilizó la ramificación main predeterminada cuando la canalización se inició manualmente.




En el siguiente ejemplo de salida de acción, se muestra la solicitud de extracción y la ramificación que se utilizó para el desencadenador cuando se filtró por solicitud de extracción.



# Agregación de un desencadenador con la inserción de código sin filtro

Los desencadenadores permiten configurar la canalización para que se inicie con un tipo de evento concreto, como una solicitud de extracción o inserción de código. Los desencadenadores se pueden configurar para las acciones de origen con conexiones que utilizan la CodeStarSourceConnection acción en CodePipeline, por ejemplo GitHub, Bitbucket y GitLab.

Agregación de un desencadenador (consola)

1. Inicia sesión en la CodePipeline consola Consola de administración de AWS y ábrela en http://console.aws.amazon. com/codesuite/codepipeline/home.

   Se muestran los nombres y el estado de todas las canalizaciones asociadas a tu AWS cuenta.
2. En Nombre, elija el nombre de la canalización que desea editar. De lo contrario, siga estos pasos en el asistente de creación de canalizaciones.
3. En la página de detalles de la canalización, elija Edit.
4. En la página Editar, seleccione la acción de origen que desee editar. Seleccione Editar activadores. Elija la opción para agregar un desencadenador.
5. En Tipo de desencadenador, elija Sin filtro.

# Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código

Puede configurar filtros para que los desencadenadores de canalización inicien las ejecuciones de canalización para diferentes eventos de Git, como la inserción de etiquetas o ramificaciones, los cambios en rutas de archivo específicas, las solicitudes de extracción abiertas en una ramificación específica, etc. Puede usar la AWS CodePipeline consola o los update-pipeline comandos create-pipeline y de la AWS CLI para configurar los filtros de activación.

> **Note**
>
> El campo `BranchName` de configuración una acción define una sola rama, mientras que los desencadenadores con filtros se pueden usar para las ramas que se especifiquen. En el caso de una canalización en la que se utilizan desencadenadores para filtrar las ramas

por solicitud de inserción o extracción, la canalización no utilizará la rama predeterminada del campo `BranchName` en la configuración de la acción. Sin embargo, la rama del campo `BranchName` en la configuración de la acción es la predeterminada cuando la canalización se inicia manualmente. Para ver un ejemplo, consulta 5: El disparador está configurado mientras que la configuración de acción predeterminada BranchName se utiliza para un inicio manual.

Puede especificar filtros para los siguientes tipos de desencadenadores:

- Inserción

  Un desencadenador de inserción inicia una canalización cuando se envía un cambio al repositorio de origen. La ejecución utilizará la confirmación de la ramificación a la que se realice la inserción (es decir, la ramificación de destino). Puede filtrar desencadenadores de inserción en ramificaciones, rutas de archivo o etiquetas de Git.

- Solicitud de extracción

  Un desencadenador de solicitud de extracción inicia una canalización cuando se abre, actualiza o cierra una solicitud de extracción en el repositorio de origen. La ejecución utilizará la confirmación de la ramificación de origen a la que se realice la extracción (es decir, la ramificación de origen). Puede filtrar los desencadenadores de las solicitudes de extracción por ramificaciones y rutas de archivo.

  Los tipos de eventos admitidos para las solicitudes de extracción se incluyen a continuación. Todos los demás eventos de solicitudes de extracción no se tienen en cuenta.

  - Opened (Abiertos)
  - Actualizado
  - Cerrado (fusionado)

  > **Note**
  >
  > El comportamiento de algunos eventos de solicitud de extracción puede variar según el proveedor. Para obtener más información, consulte Eventos de solicitud de extracción para desencadenadores por proveedor.

La definición de canalización permite combinar diferentes filtros en la misma configuración de desencadenador de inserción. Para obtener más detalles acerca de la definición de canalización, consulte Agregación de filtros para los tipos de eventos de solicitud de inserción y extracción (CLI). Para obtener una lista de las definiciones de campos, consulte los desencadenadores en la Referencia de la estructura de canalización de esta guía.

Temas


- Agregación de filtros para los tipos de eventos de solicitud de inserción y extracción (consola)
- Agregación de filtros para los tipos de eventos de solicitud de inserción y extracción (CLI)
- Añade filtros para los tipos de eventos de solicitud push y pull (CloudFormation plantillas)


## Agregación de filtros para los tipos de eventos de solicitud de inserción y extracción (consola)

Puede utilizar la consola para agregar filtros para los eventos de inserción, así como incluir o excluir ramificaciones o rutas de archivo.

Agregación de filtros (consola)

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres y el estado de todas las canalizaciones asociadas a su AWS cuenta.
2. En Nombre, elija el nombre de la canalización que desea editar. De lo contrario, siga estos pasos en el asistente de creación de canalizaciones.
3. En la página de detalles de la canalización, elija Edit.
4. En la página Editar, seleccione la acción de origen que desee editar. Seleccione Editar activadores. Elija Especificar filtro.
5. En Tipo de evento, elija Insertar entre las siguientes opciones.

   - Seleccione Insertar para iniciar la canalización cuando se inserte un cambio en su repositorio de origen. Al elegir esta opción, los campos especifican filtros para las ramificaciones y las rutas de archivo o las etiquetas de Git.
   - Seleccione Solicitud de extracción para iniciar una canalización cuando se abra, actualice o cierre una solicitud de extracción en el repositorio de origen. Al elegir esta opción, los campos especifican filtros para las ramificaciones de destino y las rutas de archivo.

6. En Inserción, en Tipo de filtro, elija una de las siguientes opciones.

   - Seleccione Ramificación para especificar las ramificaciones del repositorio de origen que el desencadenador supervisa para saber cuándo iniciar la ejecución de un flujo de trabajo. En Incluir, introduzca los patrones de los nombres de las ramificaciones en formato glob que desee especificar para la configuración de desencadenador a fin de iniciar la canalización al realizar cambios en las ramificaciones especificadas. En Excluir, introduzca los patrones regex de los nombres de las ramificaciones en formato glob que desee especificar para que la configuración del desencadenador los ignore y no se inicie la canalización al realizar cambios en las ramificaciones especificadas. Para obtener más información, consulte Trabajar con patrones glob en la sintaxis.

     > **Note**
     >
     > Si tanto la inclusión como la exclusión tienen el mismo patrón, la opción predeterminada es excluir el patrón.

     Puede usar patrones glob para definir los nombres de las ramificaciones. Por ejemplo, use `main*` para hacer coincidir todas las ramificaciones que comiencen por `main`. Para obtener más información, consulte Trabajar con patrones glob en la sintaxis.

     En el caso de un desencadenador de inserción, especifique las ramificaciones en las que va a realizar la inserción, es decir, las ramificaciones de destino. Para activar una solicitud de extracción, especifica las sucursales de destino en las que vas a abrir una solicitud de extracción.

   - (Opcional) En Rutas de archivo, especifique las rutas de archivo para su desencadenador. Introduzca los nombres en Incluir y Excluir según corresponda.

     Puede usar patrones glob para definir los nombres de las rutas de archivo. Por ejemplo, use `prod*` para hacer coincidir todas las rutas de archivo que comiencen por `prod`. Para obtener más información, consulte Trabajar con patrones glob en la sintaxis.

   - Para configurar la configuración del desencadenador de canalización para que comience con las etiquetas de Git, seleccione Etiquetas. En Incluir, introduzca los patrones de los nombres de las etiquetas en formato glob que desee especificar para la configuración del desencadenador a fin de iniciar la canalización al liberar la etiqueta o etiquetas especificadas. En Excluir, introduzca los patrones regex de los nombres de las etiquetas en formato glob

que desee especificar para que la configuración del desencadenador los ignore y no inicie la canalización al liberar la etiqueta o etiquetas especificadas. Si tanto la inclusión como la exclusión tienen el mismo patrón de etiquetas, la opción predeterminada es excluir el patrón de etiquetas.

7. En Inserción, en Tipo de filtro, elija una de las siguientes opciones.

   - Seleccione Ramificación para especificar las ramificaciones del repositorio de origen que el desencadenador supervisa para saber cuándo iniciar la ejecución de un flujo de trabajo. En Incluir, introduzca los patrones de los nombres de las ramificaciones en formato glob que desee especificar para la configuración de desencadenador a fin de iniciar la canalización al realizar cambios en las ramificaciones especificadas. En Excluir, introduzca los patrones regex de los nombres de las ramificaciones en formato glob que desee especificar para que la configuración del desencadenador los ignore y no se inicie la canalización al realizar cambios en las ramificaciones especificadas. Para obtener más información, consulte Trabajar con patrones glob en la sintaxis.

     > **Note**
     >
     > Si tanto la inclusión como la exclusión tienen el mismo patrón, la opción predeterminada es excluir el patrón.

     Puede usar patrones glob para definir los nombres de las ramificaciones. Por ejemplo, use `main*` para hacer coincidir todas las ramificaciones que comiencen por `main`. Para obtener más información, consulte Trabajar con patrones glob en la sintaxis.

     En el caso de un desencadenador de inserción, especifique las ramificaciones en las que va a realizar la inserción, es decir, las ramificaciones de destino. Para activar una solicitud de extracción, especifica las sucursales de destino en las que vas a abrir una solicitud de extracción.

   - (Opcional) En Rutas de archivo, especifique las rutas de archivo para su desencadenador. Introduzca los nombres en Incluir y Excluir según corresponda.

     Puede usar patrones glob para definir los nombres de las rutas de archivo. Por ejemplo, use `prod*` para hacer coincidir todas las rutas de archivo que comiencen por `prod`. Para obtener más información, consulte Trabajar con patrones glob en la sintaxis.

- Elija Solicitud de extracción para establecer la configuración de los desencadenadores de modo que se empiece con los eventos de solicitud de extracción que se especifiquen.

## Agregación de filtros para los tipos de eventos de solicitud de inserción y extracción (CLI)

Puede actualizar el archivo JSON de la canalización para agregar filtros a los desencadenadores.

Si quieres usar el AWS CLI para crear o actualizar tu canalización, usa el `update-pipeline` comando `create-pipeline` o.

El siguiente ejemplo de estructura JSON proporciona una referencia para las definiciones de campo mediante `create-pipeline`.

Para obtener una lista de las definiciones de campos, consulte los desencadenadores en la Referencia de la estructura de canalización de esta guía.

```
{
    "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"
                        }
                    }
                ]
            }
        ]
    }
}
```

## Añade filtros para los tipos de eventos de solicitud push y pull (CloudFormation plantillas)

Puedes actualizar el recurso de canalización en CloudFormation para añadir un filtrado de desencadenantes.

A continuación se muestra un ejemplo de un fragmento de plantilla que proporciona una referencia de YAML para las definiciones de los campos de desencadenadores. Para obtener una lista de las definiciones de campos, consulte los desencadenadores en la Referencia de la estructura de canalización de esta guía.

Para ver un ejemplo completo de plantilla para la configuración de una fuente de conexión y un filtro de activación, consulta la sección Canalización con dos etapas y configuración de activadores en la Guía del CloudFormation usuario.

```
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
```

# Agregación de un desencadenador para desactivar la detección de cambios

Los desencadenadores permiten configurar la canalización para que se inicie con un tipo de evento concreto, como una solicitud de extracción o inserción de código. Los desencadenadores se pueden configurar para las acciones de origen con conexiones que utilizan la `CodeStarSourceConnection` acción en CodePipeline, por ejemplo GitHub, Bitbucket y. GitLab

Agregación de un desencadenador para desactivar la detección de cambios (consola)

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en. http://console.aws.amazon.com/codesuite/codepipeline/home

   Se muestran los nombres y el estado de todas las canalizaciones asociadas a su AWS cuenta.
2. En Nombre, elija el nombre de la canalización que desea editar. De lo contrario, siga estos pasos en el asistente de creación de canalizaciones.
3. En la página de detalles de la canalización, elija Edit.
4. En la página Editar, seleccione la acción de origen que desee editar. Seleccione Editar activadores. Elija la opción para agregar un desencadenador.
5. En Tipo de desencadenador, elija No detectar cambios.

# Inicio y detención manuales de canalizaciones

Puede utilizar la AWS CodePipeline consola o la AWS CLI para iniciar y detener las canalizaciones manualmente. Existen métodos automatizados para iniciar canalizaciones con la detección de cambios basada en eventos, como se detalla en , o la configuración de desencadenador, como se detalla en .

Temas


- Inicie una canalización en CodePipeline
- Detener la ejecución de una canalización en CodePipeline


## Inicie una canalización en CodePipeline

Cada ejecución de la canalización se puede iniciar según un desencadenador diferente. Cada ejecución de la canalización puede tener un tipo de desencadenador diferente, en función de cómo se inicie la canalización. El tipo de desencadenador de cada ejecución se muestra en el historial de ejecuciones de una canalización. Los tipos de desencadenadores pueden depender del proveedor de la acción de origen, de la siguiente manera:

> **Note**
>
> No puede especificar más de un desencadenador por acción de origen.

- Creación de canalización: cuando se crea una canalización, la ejecución de una canalización se inicia automáticamente. Este es el tipo de desencadenador `CreatePipeline` en el historial de ejecuciones.
- Cambios en los objetos revisados: esta categoría representa el tipo de desencadenador `PutActionRevision` en el historial de ejecuciones.
- Detección de cambios en la ramificación y confirmación de una inserción de código: esta categoría representa el tipo de desencadenador `CloudWatchEvent` en el historial de ejecuciones. Cuando se detecta un cambio en una confirmación de origen y en una ramificación del repositorio de origen, se inicia la canalización. Este tipo de desencadenador utiliza la detección automática de cambios. Los proveedores de acciones de origen que utilizan este tipo de desencadenador son S3 y CodeCommit. Este tipo también se usa para un cronograma que inicia su canalización. Consulte Inicio de una canalización según una programación.

- Sondeo para detectar cambios en la fuente: esta categoría representa el tipo de desencadenador `PollForSourceChanges` en el historial de ejecuciones. Cuando se detecta un cambio en una confirmación de origen y se ramifica en el repositorio de origen mediante un sondeo, se inicia la canalización. No se recomienda este tipo de desencadenador y se debe migrar para utilizar la detección automática de cambios. Los proveedores de acciones de origen que utilizan este tipo de desencadenador son S3 y CodeCommit.
- Eventos de Webhook para orígenes de terceros: esta categoría representa el tipo de desencadenador `Webhook` en el historial de ejecuciones. Cuando un evento de webhook detecta un cambio, se inicia su canalización. Este tipo de desencadenador utiliza la detección automática de cambios. Los proveedores de acciones de origen que utilizan este tipo de desencadenador son conexiones configuradas para la inserción de código (Bitbucket Cloud GitHub, GitHub Enterprise Server y GitLab autogestionables). GitLab.com
- Eventos de WebhookV2 para orígenes de terceros: esta categoría representa el tipo de desencadenador `WebhookV2` en el historial de ejecuciones. Este tipo es para las ejecuciones que se activan en función de los desencadenadores definidos en la definición de canalización. Cuando se detecta un lanzamiento con una etiqueta de Git específica, se inicia su proceso. Puede utilizar una etiqueta de Git para marcar una confirmación con un nombre u otro identificador que ayude a otros usuarios a comprender su importancia. También puede utilizar etiquetas de Git para identificar una determinada confirmación en el historial de un repositorio. Este tipo de desencadenador desactiva la detección automática de cambios. Los proveedores de acciones de origen que utilizan este tipo de activación son conexiones configuradas para etiquetas Git (Bitbucket Cloud GitHub, GitHub Enterprise Server y). GitLab.com
- Iniciar una canalización manualmente: esta categoría representa el tipo de desencadenador **`StartPipelineExecution`** en el historial de ejecuciones. Puedes usar la consola o la AWS CLI para iniciar una canalización de forma manual. Para obtener información, consulte Iniciar la canalización manualmente.
- RollbackStage: Esta categoría representa el tipo de `RollbackStage` desencadenador en el historial de ejecución. Puede utilizar la consola o la AWS CLI para hacer retroceder una fase de forma manual o automática. Para obtener información, consulte Configuración de la reversión de etapas.

Cuando agrega una acción de origen a una canalización que utiliza tipos de desencadenadores de detección automática de cambios, las acciones funcionan con recursos adicionales. La creación de cada acción de origen se detalla en secciones independientes, gracias a estos recursos adicionales para la detección de cambios. Para obtener más información sobre los recursos de detección de

cambios que se aplican a cada proveedor de origen, consulte Acciones de origen y métodos de detección de cambios.

Temas


- Iniciar la canalización manualmente
- Inicio de una canalización según una programación
- Iniciar una canalización con una anulación de revisión de código fuente


## Iniciar la canalización manualmente

De forma predeterminada, las canalizaciones se inician automáticamente al crearlas y, después, cada vez que se realiza un cambio en un repositorio de código fuente. Sin embargo, puede que desee ejecutar otra vez la revisión más reciente en la canalización. Puedes usar la CodePipeline consola o el start-pipeline-execution comando AWS CLI and para volver a ejecutar manualmente la revisión más reciente en tu proceso.

Temas


- Iniciar manualmente una canalización (consola)
- Iniciar manualmente una canalización (CLI)


### Iniciar manualmente una canalización (consola)

Para iniciar manualmente una canalización y ejecutar la revisión más reciente a través de una canalización

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En Name, elija el nombre de la canalización que desea iniciar.
3. En la página de detalles de la canalización, elija Liberar cambio. Si la canalización está configurada para transmitir parámetros (variables de canalización), al seleccionar Liberar cambio se abre la ventana Liberar cambio. En Variables de canalización, en el campo o campos de las variables a nivel de canalización, introduzca el valor o los valores que desea transferir para la ejecución de esta canalización. Para obtener más información, consulte Referencia de variables.

   Esto inicia la revisión más reciente disponible en cada ubicación de código fuente especificada en una acción de código fuente a través de la canalización.

### Iniciar manualmente una canalización (CLI)

Para iniciar manualmente una canalización y ejecutar la versión más reciente de un artefacto a través de una canalización

1. Abra un terminal (Linux, macOS o Unix) o el símbolo del sistema (Windows) y utilice la AWS CLI para ejecutar el comando start-pipeline-execution, especificando el nombre de la canalización que desea iniciar. Por ejemplo, para comenzar a ejecutar el último cambio a través de una canalización denominada *MyFirstPipeline*:

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

   Para iniciar una canalización en la que las variables estén configuradas a nivel de canalización, utilice el comando start-pipeline-execution con el argumento opcional --variables para iniciar la canalización y añada las variables que se utilizarán en la ejecución. Por ejemplo, para añadir una variable `var1` con un valor de 1, utilice el siguiente comando:

   ```
   aws codepipeline start-pipeline-execution --name MyFirstPipeline --variables
    name=var1,value=1
   ```

2. Para comprobar que la ejecución se ha realizado correctamente, vea el objeto que se devuelve. Este comando devuelve un ID de ejecución, similar al siguiente:

   ```
   {
       "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
   }
   ```

   > **Note**
   >
   > Una vez que haya iniciado la canalización, puede supervisar su progreso en la CodePipeline consola o ejecutando el get-pipeline-state comando. Para obtener más información, consulte Ver canalizaciones (consola) y Visualización de los detalles y el historial de la canalización (CLI).

## Inicio de una canalización según una programación

Puede configurar una regla EventBridge para iniciar una canalización según un cronograma.

## Crea una EventBridge regla que programe el inicio de tu canalización (consola)

Para crear una EventBridge regla con una programación como origen del evento

1. Abre la EventBridge consola de Amazon en https://console.aws.amazon.com/events/.
2. En el panel de navegación, seleccione Reglas.
3. Elija Crear regla y, a continuación, en Detalle de regla, elija Programación.
4. Configure la programación utilizando un intervalo o una expresión establecidos. Para obtener más información, consulte Programar expresiones para reglas.
5. En Targets, elige CodePipeline.
6. Introduzca el ARN de canalización para la ejecución de la canalización.

> **Note**
>
> Puede encontrar el ARN de la canalización en Configuración de la consola. Consulte Ver el ARN de la canalización y el ARN del rol de servicio (consola).

7. Elija una de las siguientes opciones para crear o especificar un rol de servicio de IAM que EventBridge otorgue permisos para invocar el objetivo asociado a su EventBridge regla (en este caso, el objetivo es CodePipeline).
   - Elige Crear un rol nuevo para este recurso específico a fin de crear un rol de servicio que otorgue EventBridge permisos para iniciar las ejecuciones de tu canalización.
   - Elige Usar el rol existente para ingresar un rol de servicio que otorgue EventBridge permisos para iniciar las ejecuciones de tu canalización.
8. Seleccione Configurar los detalles.
9. En la página Configure rule details (Configurar detalles de regla), escriba un nombre y una descripción para la regla y, a continuación, elija State (Estado) para habilitarla.
10. Si está satisfecho con la regla, elija Create rule (Crear regla).

## Crea una EventBridge regla que programe el inicio de tu canalización (CLI)

Para usar la AWS CLI para crear una regla, ejecuta el put-rule comando y especifica:

- Un nombre que identifique de forma inequívoca la regla que está creando. Este nombre debe ser único en todas las canalizaciones que crees y que estén CodePipeline asociadas a tu AWS cuenta.
- La expresión de programación para la regla.

Para crear una EventBridge regla con una programación como fuente de eventos

1. Llame al comando put-rule e incluya los parámetros `--name` y `--schedule-expression`.

   Ejemplos:

   El siguiente comando de ejemplo se utiliza --schedule-expression para crear una regla denominada `MyRule2` que filtra EventBridge según una programación.

   ```
   aws events put-rule --schedule-expression 'cron(15 10 ? * 6L 2002-2005)' --name
    MyRule2
   ```

2. Para agregarlo CodePipeline como destino, ejecute el put-targets comando e incluya los siguientes parámetros:

   - El parámetro `--rule` se usa con el `rule_name` que creó con el comando put-rule.
   - El parámetro `--targets` se usa con el `Id` del destino de la lista de destinos y el ARN de la canalización de destino.

   El siguiente comando de muestra especifica que, para la regla denominada `MyCodeCommitRepoRule`, el destino `Id` se compone del número uno, lo que indica que, en lo que puede ser una lista de destinos de la regla, se trata del destino 1. El comando de muestra también especifica un ARN de ejemplo para la canalización. La canalización se inicia cuando se produce algún cambio en el repositorio.

   ```
   aws events put-targets --rule MyCodeCommitRepoRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

3. Conceda los permisos EventBridge para utilizarlos CodePipeline para invocar la regla. Para obtener más información, consulte Uso de políticas basadas en recursos para Amazon. EventBridge

a. Utilice el siguiente ejemplo para crear la política de confianza que permite que EventBridge asuma el rol de servicio. Denomínelo `trustpolicyforEB.json`.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Utilice el comando para crear el rol `Role-for-MyRule` y asocie la política de confianza.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Cree el JSON de la política de permisos tal y como se muestra en este ejemplo para la canalización denominada `MyFirstPipeline`. Ponga un nombre a la política de permisos `permissionspolicyforEB.json`.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
```

```
            ]
        }
    ]
}
```

d. Utilice el siguiente comando para asociar la nueva política de permisos `CodePipeline-Permissions-Policy-for-EB` al rol `Role-for-MyRule` que ha creado.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-Permissions-Policy-For-EB --policy-document file://permissionspolicyforCWE.json
```

## Iniciar una canalización con una anulación de revisión de código fuente

Puede usar las anulaciones para iniciar una canalización con un identificador de revisión de código fuente específico que proporcione para la ejecución de la canalización. Por ejemplo, si quieres iniciar una canalización que procese un identificador de confirmación específico de tu CodeCommit fuente, puedes añadir el identificador de confirmación como anulación al iniciar el proceso.

> **Note**
>
> También puedes crear una anulación de fuente mediante la entrada de transformación de entrada EventBridge para usar la entrada `revisionValue` en tu evento de canalización, donde la variable de evento de origen de tu clave de objeto, confirmación o ID de imagen `revisionValue` se deriva de la variable de evento de origen. Para obtener más información, consulte el paso opcional para la entrada de transformación de entrada, que se incluye en los procedimientos de Acciones y recursos de origen de Amazon ECR EventBridge, Conexión a acciones de origen de Amazon S3 con un origen habilitado para eventos o CodeCommit acciones de origen y EventBridge.

Existen cuatro tipos de revisión de código fuente para `revisionType`:

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

> **Note**
>
> En el caso de las revisiones de código fuente de los tipos `COMMIT_ID` y `IMAGE_DIGEST`, el identificador de revisión de código fuente se aplica a todo el contenido del repositorio, en todas las ramificaciones.

> **Note**
>
> En el caso de las revisiones de código fuente `S3_OBJECT_VERSION_ID` y `S3_OBJECT_KEY` los tipos, se puede usar cualquiera de los dos tipos de forma independiente o se pueden usar juntos para anular el código fuente con un ID de versión específico ObjectKey . Para `S3_OBJECT_KEY`, el parámetro de configuración `AllowOverrideForS3ObjectKey` debe estar establecido en `true`. Para obtener más información sobre los parámetros de configuración del origen de S3, consulte Parámetros de configuración.

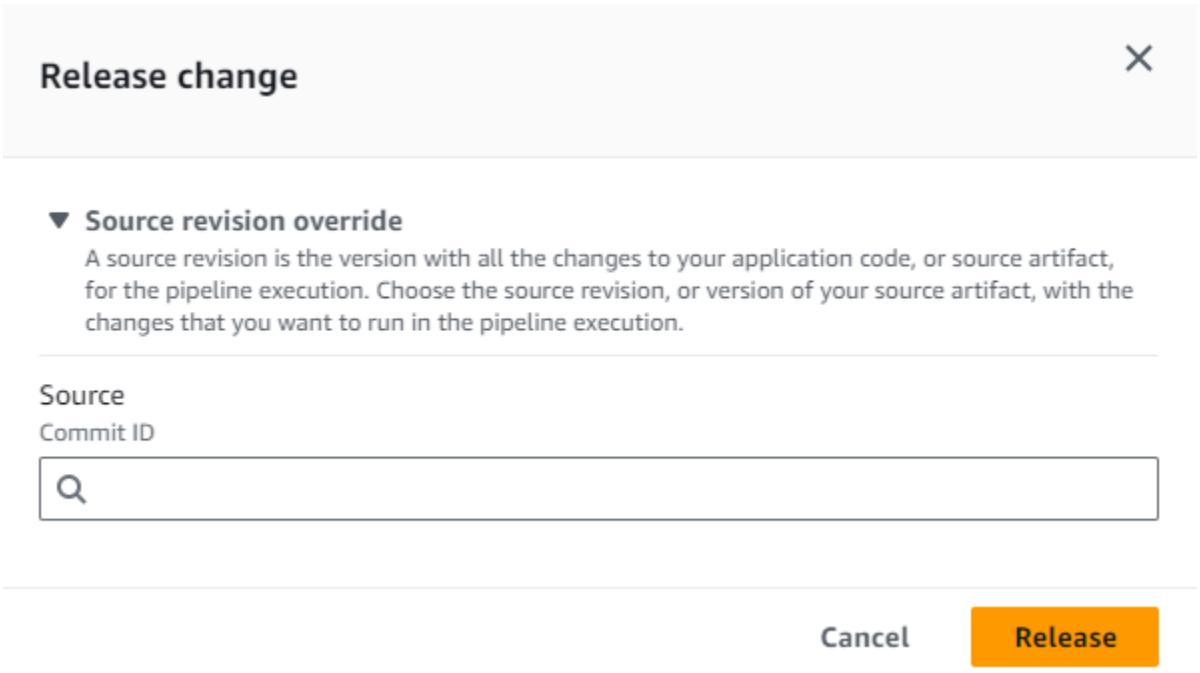
Temas

- Iniciar una canalización con una anulación de revisión de código fuente (consola)
- Iniciar una canalización con una anulación de revisión de código fuente (CLI)

## Iniciar una canalización con una anulación de revisión de código fuente (consola)

Para iniciar manualmente una canalización y ejecutar la revisión más reciente a través de una canalización

1. Inicia sesión en Consola de administración de AWS y abre la consola en CodePipeline . http://console.aws.amazon.com/codesuite/codepipeline/home
2. En Name, elija el nombre de la canalización que desea iniciar.
3. En la página de detalles de la canalización, elija Liberar cambio. Si selecciona Cambio de versión, se abre la ventana Cambio de versión. Para Anular la revisión del código fuente, elija la flecha para ampliar el campo. En Fuente, introduzca el ID de revisión del código fuente. Por ejemplo, si tu canalización tiene una CodeCommit fuente, elige el ID de confirmación en el campo que quieras usar.




## Iniciar una canalización con una anulación de revisión de código fuente (CLI)

Para iniciar una canalización manualmente y ejecutar el ID de revisión de código fuente especificado para un artefacto a través de una canalización

1. Abra un terminal (Linux, macOS o Unix) o el símbolo del sistema (Windows) y utilice la AWS CLI para ejecutar el comando start-pipeline-execution, especificando el nombre de la canalización que desea iniciar. También utiliza el argumento --source-revisions para proporcionar el ID de revisión de código fuente. La revisión del código fuente se compone de actionName, revisionType y revisionValue. Los valores de revisionType válidos son `COMMIT_ID | IMAGE_DIGEST | S3_OBJECT_VERSION_ID | S3_OBJECT_KEY`.

   En el ejemplo siguiente, para empezar a ejecutar el cambio especificado a través de una canalización denominadacodecommit-pipeline, el comando siguiente especifica el nombre de la acción de origen, Source, el tipo de `COMMIT_ID` revisión y el identificador de confirmación de`78a25c18755ccac3f2a9eec099dEXAMPLE`.

   ```
   aws codepipeline start-pipeline-execution --name codecommit-pipeline --source-
   revisions
    actionName=Source,revisionType=COMMIT_ID,revisionValue=78a25c18755ccac3f2a9eec099dEXAMPLE
    --region us-west-1
   ```

2. Para comprobar que la ejecución se ha realizado correctamente, vea el objeto que se devuelve. Este comando devuelve un ID de ejecución, similar al siguiente:

```
{
    "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
}
```

> **Note**
>
> Una vez iniciada la canalización, puedes supervisar su progreso en la CodePipeline consola o ejecutando el get-pipeline-state comando. Para obtener más información, consulte Ver canalizaciones (consola) y Visualización de los detalles y el historial de la canalización (CLI).

# Detener la ejecución de una canalización en CodePipeline

Cuando una ejecución de canalización comienza a ejecutarse por una canalización, entra en una etapa cada vez y bloquea la etapa mientras se están ejecutando todas las ejecuciones de acciones de la etapa. Estas acciones en curso se deben controlar de tal manera que, cuando se detenga la ejecución de la canalización, se permita que las acciones se completen o se abandonen.

Hay dos formas de detener una ejecución de canalización:

- Detener y esperar: AWS CodePipeline espera a detener la ejecución hasta que se hayan completado todas las acciones en curso (es decir, que las acciones tengan un `Failed` estado `Succeeded` o). Esta opción conserva las acciones en curso. La ejecución se encuentra en un estado `Stopping` hasta que se completen las acciones en curso. A continuación, la ejecución se encuentra en un estado `Stopped`. La etapa se desbloquea después de completar las acciones.

  Si elige detener y esperar, y cambia de opinión mientras la ejecución se encuentra todavía en estado `Stopping`, puede elegir abandonar.
- Detener y abandonar: AWS CodePipeline detiene la ejecución sin esperar a que se completen las acciones en curso. La ejecución se encuentra en un estado `Stopping` durante un tiempo muy corto mientras que las acciones en curso se abandonan. Después de que la ejecución se detenga, la ejecución de la acción se encuentra en un estado `Abandoned` mientras que la ejecución de la canalización se encuentra en un estado `Stopped`. La etapa se desbloquea.

En el caso de una ejecución de canalización en un estado `Stopped`, se pueden reintentar las acciones en la etapa en la que se detuvo la ejecución.

> **Warning**
>
> Esta opción puede conducir a tareas con error o fuera de secuencia.

Temas


- Detener la ejecución de una canalización (consola)
- Detener una ejecución entrante (consola)
- Detener la ejecución de una canalización (CLI)
- Detener una ejecución entrante (CLI)


## Detener la ejecución de una canalización (consola)

Puede utilizar la consola para detener la ejecución de una canalización. Elija una ejecución y, a continuación, elija el método para detener la ejecución de la canalización.

> **Note**
>
> También puede detener una ejecución de canalización que sea una ejecución entrante. Para obtener más información sobre cómo detener una ejecución entrante, consulte Detener una ejecución entrante (consola).

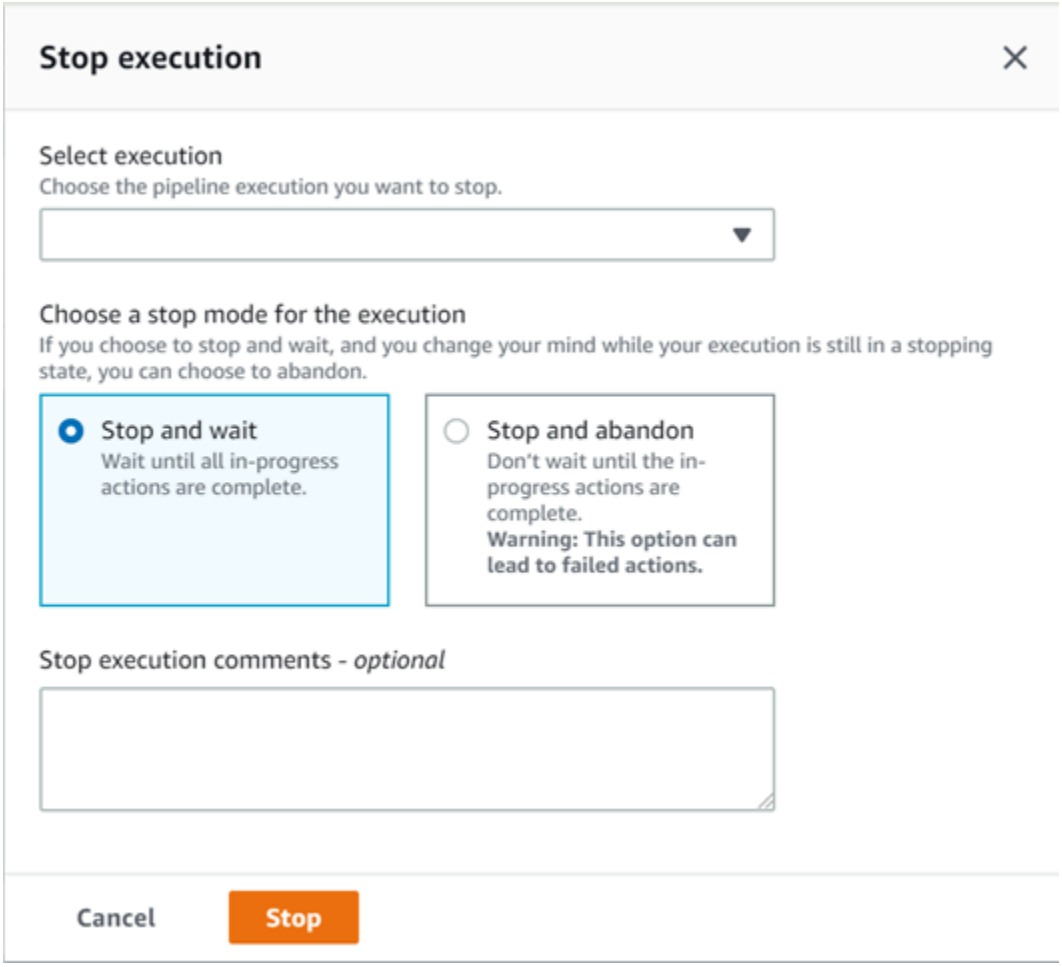
1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Realice una de las siguientes acciones:

> **Note**
>
> Antes de detener una ejecución, le recomendamos que desactive la transición antes de la etapa. De esta manera, cuando la etapa se desbloquea debido a la ejecución detenida, la etapa no acepta una ejecución de canalización posterior.

- En Name (Nombre), elija el nombre de la canalización con la ejecución que desea detener. En la página de detalles de canalización, elija Stop execution (Detener ejecución).
- Seleccione Ver historial. En la página de historial, elija Stop execution (Detener ejecución).

3. En la página Stop execution (Detener ejecución) en Select execution (Seleccionar ejecución), elija la ejecución que desea detener.

> **Note**
>
> La ejecución solo se muestra si todavía está en curso. No se muestran las ejecuciones que ya están completadas.




4. En Select an action to apply to execution (Seleccione una acción para aplicar a la ejecución), elija una de las opciones siguientes:

- Para asegurarse de que la ejecución no se detiene hasta que se hayan completado todas las acciones en curso, elija Stop and wait (Detener y esperar).

  > **Note**
  >
  > No puede optar por detener y esperar si la ejecución ya está en un estado Stopping (Deteniéndose), pero puede optar por detener y abandonar.

- Para detener sin esperar a que se completen las acciones en curso, elija Stop and abandon (Detener y abandonar).

  > **Warning**
  >
  > Esta opción puede conducir a tareas con error o fuera de secuencia.

5. (Opcional) Escriba comentarios. Estos comentarios, junto con el estado de ejecución, se muestran en la página de historial de la ejecución.
6. Elija Detener.

   > **Important**
   >
   > Esta acción no se puede deshacer.

7. Vea el estado de ejecución en la visualización de canalizaciones de la siguiente manera:

   - Si elige detener y esperar, la ejecución seleccionada continúa hasta que se completen las acciones en curso.
     - El mensaje de banner correcto se muestra en la parte superior de la consola.
     - En la etapa actual, las acciones en curso continúan en un estado `InProgress`. Mientras las acciones están en curso, la ejecución de la canalización se encuentra en un estado `Stopping`.

       Después de que las acciones se completan (es decir, la acción se realiza correcta o incorrectamente), la ejecución de canalización cambia a un estado `Stopped` y la acción cambia a un estado `Failed` o `Succeeded`. También puede ver el estado de la acción en la página de detalles de ejecución. Puede ver el estado de ejecución en la página de historial de ejecución o en la página de detalles de ejecución.

- La ejecución de canalización cambia a un estado `Stopping` brevemente y, a continuación, cambia a un estado `Stopped`. Puede ver el estado de ejecución en la página de historial de ejecución o en la página de detalles de ejecución.

- Si elige detener y abandonar, la ejecución no espera a que se completen las acciones en curso.
  - El mensaje de banner correcto se muestra en la parte superior de la consola.
  - En la etapa actual, las acciones en curso cambian a un estado de `Abandoned`. También puede ver el estado de la acción en la página de detalles de ejecución.
  - La ejecución de canalización cambia a un estado `Stopping` brevemente y, a continuación, cambia a un estado `Stopped`. Puede ver el estado de ejecución en la página de historial de ejecución o en la página de detalles de ejecución.

Puede ver el estado de ejecución de canalización en la vista de historial de ejecución y en la vista de historial detallado.

## Detener una ejecución entrante (consola)

Puede utilizar la consola para detener una ejecución entrante. Una ejecución entrante es una ejecución en proceso que espera entrar en una fase en la que la transición se ha desactivado. Cuando la transición está habilitada, una ejecución entrante `InProgress` continúa ingresando a la etapa. Una ejecución entrante que `Stopped` no ingresa en la fase.

> **Note**
>
> Una vez detenida una ejecución entrante, no se puede volver a intentar.

Si no ve ninguna ejecución entrante, significa que no hay ninguna ejecución pendiente en una fase de transición desactivada.

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se mostrarán los nombres de todas las canalizaciones asociadas a su AWS cuenta.
2. Elige el nombre de la canalización cuya ejecución entrante quieres detener. Realice una de las siguientes acciones:

- En la vista de canalización, elija el ID de ejecución entrante y, a continuación, elija detener la ejecución.
- Seleccione la canalización y elija Ver historial. En el historial de ejecuciones, elija el identificador de ejecución entrante y, a continuación, elija detener la ejecución.

3. En el modo de detener la ejecución, siga los pasos de la sección anterior para seleccionar el ID de ejecución y especificar el método de detención.

   Utilice el comando get-pipeline-state para ver el estado de la ejecución entrante.

## Detener la ejecución de una canalización (CLI)

Para usar el AWS CLI para detener manualmente una canalización, usa el stop-pipeline-execution comando con los siguientes parámetros:

- ID de ejecución (obligatorio)
- Comentarios (opcional)
- Nombre de canalización (obligatorio)
- Indicador de abandono (opcional, el valor predeterminado es false)

Formato de comando:

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --pipeline-
execution-id Execution_ID [--abandon | --no-abandon] [--reason STOP_EXECUTION_REASON]
```

1. Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows).
2. Para detener la ejecución de canalización, elija una de las opciones siguientes:

   - Para asegurarse de que la ejecución no se detiene hasta que se hayan completado todas las acciones en curso, elija detener y esperar. Puede hacerlo si incluye el parámetro no-abandon. Si no especifica el parámetro, el comando es detener y esperar de forma predeterminada. Utilice el AWS CLI para ejecutar el stop-pipeline-execution comando, especificando el nombre de la canalización y el identificador de ejecución. Por ejemplo, para detener una canalización denominada *MyFirstPipeline* con la opción de detener y esperar especificada:

```
aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
pipeline-execution-id d-EXAMPLE --no-abandon
```

Por ejemplo, para detener una canalización denominada *MyFirstPipeline*, con la opción predeterminada de detener y esperar, y elegir la inclusión de comentarios:

```
aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
pipeline-execution-id d-EXAMPLE --reason "Stopping execution after the build
 action is done"
```

> **Note**
>
> No puede elegir detener y esperar si la ejecución ya se encuentra en un estado Stopping (Deteniéndose). Puede optar por detener y abandonar una ejecución que ya se encuentra en un estado Stopping (Deteniéndose).

- Para detener sin esperar a que se completen las acciones en curso, elija detener y abandonar. Incluya el parámetro abandon. Utilice el AWS CLI para ejecutar el stop-pipeline-execution comando, especificando el nombre de la canalización y el identificador de ejecución.

  Por ejemplo, para detener una canalización denominada *MyFirstPipeline*, especificando la opción de abandonar y eligiendo la inclusión de comentarios:

  ```
  aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
  pipeline-execution-id d-EXAMPLE --abandon --reason "Stopping execution for a bug
   fix"
  ```

## Detener una ejecución entrante (CLI)

Puede utilizar la CLI para detener una ejecución entrante. Una ejecución entrante es una ejecución en proceso que espera entrar en una fase en la que la transición se ha desactivado. Cuando la transición está habilitada, una ejecución entrante `InProgress` continúa ingresando a la etapa. Una ejecución entrante que `Stopped` no ingresa en la fase.

> **Note**
>
> Una vez detenida una ejecución entrante, no se puede volver a intentar.

Si no ve ninguna ejecución entrante, significa que no hay ninguna ejecución pendiente en una fase de transición desactivada.

Para detener manualmente una ejecución entrante, utilice el stop-pipeline-execution comando con los siguientes parámetros: AWS CLI

- ID de ejecución entrante(obligatorio)
- Comentarios (opcional)
- Nombre de canalización (obligatorio)
- Indicador de abandono (opcional, el valor predeterminado es false)

Formato de comando:

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --
pipeline-execution-id Inbound_Execution_ID [--abandon | --no-abandon] [--
reason STOP_EXECUTION_REASON]
```

Siga los pasos del procedimiento anterior para introducir el comando y especificar el método de detención.

Utilice el comando get-pipeline-state para ver el estado de la ejecución entrante.

# Consulta del historial y configuración del modo de ejecución de las canalizaciones

Para analizar el progreso de la canalización, puedes ver el historial de ejecución de la canalización y de las acciones. Para configurar el método de lanzamiento para procesar las ejecuciones de canalización, configure el modo de ejecución.

Temas


- Ver las ejecuciones en CodePipeline
- Configuración o cambio del modo de ejecución de una canalización


## Ver las ejecuciones en CodePipeline

Puede usar la AWS CodePipeline consola o la AWS CLI para ver el estado de ejecución, ver el historial de ejecuciones y volver a intentar las etapas o acciones fallidas.

Temas


- Consulta del historial de ejecución de una canalización (consola)
- Consulta del estado de ejecución (consola)
- Ver una ejecución entrante (consola)
- Consulta de las revisiones de código fuente de las ejecuciones de una canalización (consola)
- Consulta de las ejecuciones de una acción (consola)
- Consulta de artefactos de acción y de información del almacén de artefactos (consola)
- Visualización de los detalles y el historial de la canalización (CLI)


### Consulta del historial de ejecución de una canalización (consola)

Puedes usar la CodePipeline consola para ver una lista de todas las canalizaciones de tu cuenta. También puede ver los detalles de cada canalización, incluido cuándo se ejecutaron por última vez las acciones en la canalización, si una transición entre etapas está habilitada o deshabilitada y si alguna acción ha producido un error, así como otra información. Asimismo, puede ver una página de historial que muestra información sobre todas las ejecuciones de canalizaciones cuyo historial se ha registrado.

> **Note**
>
> Al cambiar entre modos de ejecución específicos, es posible que la vista y el historial de la canalización cambien. Para obtener más información, consulte Configuración o cambio del modo de ejecución de una canalización.

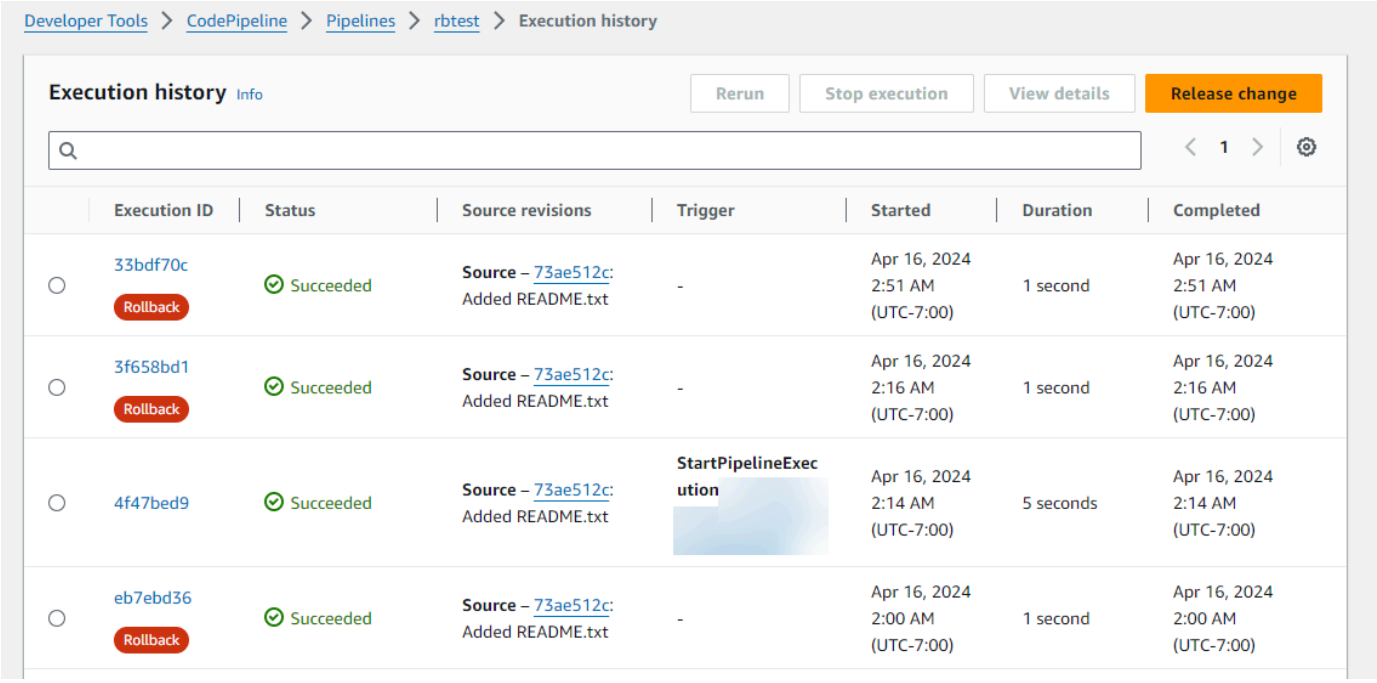
El historial de ejecución se conserva durante un máximo de 12 meses.

Puede utilizar la consola para ver el historial de las ejecuciones de una canalización, incluidos el estado, las revisiones de código fuente y los detalles sobre los tiempos para cada ejecución.

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres de todas las canalizaciones asociadas a su AWS cuenta, junto con su estado.
2. En Name, elija el nombre de la canalización.
3. Seleccione Ver historial.

> **Note**
>
> En el caso de una canalización en modo de ejecución PARALELO, la vista de canalización principal no muestra la estructura de la canalización ni las ejecuciones en curso. En el caso de una canalización en modo de ejecución PARALELO, para acceder a la estructura de la canalización hay que seleccionar el ID de la ejecución que se quiere ver en la página del historial de ejecuciones. Elija Historial en el menú de navegación de la izquierda, elija el ID de ejecución para la ejecución en paralelo y, a continuación, consulte la canalización en la pestaña Visualización.




4. Consulte el estado, las revisiones de origen, los detalles de cambio y los desencadenadores relacionados con cada ejecución de la canalización. Las ejecuciones de canalización que se hayan revertido mostrarán el tipo de ejecución Reversión en la pantalla de detalles de la consola. En el caso de la ejecución fallida que desencadenó la reversión automática, se muestra el identificador de la ejecución fallida.
5. Elija una ejecución. La vista de detalle muestra los detalles de ejecución, la pestaña Cronología, la pestaña Visualización y la pestaña Variables. Los valores de las variables a nivel de canalización se resuelven en el momento de la ejecución de la canalización y se pueden ver en el historial de ejecuciones de cada ejecución.

> **Note**
>
> Las variables de salida de las acciones de la canalización se pueden ver en la pestaña Variables de salida, en el historial de cada ejecución de acciones.

## Consulta del estado de ejecución (consola)

Puede ver el estado de una canalización en el campo Status (Estado) de la página del historial de ejecución. Elija un enlace de ID de ejecución y consulte el estado de la acción.

A continuación se indican los estados válidos para las canalizaciones, las etapas y las acciones:

> **Note**
>
> Los siguientes estados de canalización también se aplican a una ejecución de canalización que sea una ejecución entrante. Para ver una ejecución entrante y su estado, consulte. Ver una ejecución entrante (consola)

Pipeline-level estados

| Estado de la canalización | Description (Descripción) |
| --- | --- |
| InProgress | La ejecución de la canalización está en curso. |
| Detención | La ejecución de canalización se detiene debido a una solicitud para detener y esperar, o bien detener y abandonar, la ejecución de canalización. |
| Stopped | El proceso de detención se ha completado y la ejecución de canalización se detiene. |
| Realizado correctamente | La ejecución de la canalización finalizó correctamente. |
| Superseded | Mientras la ejecución de esta canalización esperaba a que se completas e la siguiente etapa, una nueva ejecución de la canalización avanzó y continuó a través de la canalización en su lugar. |
| Con error | La ejecución de la canalización no finalizó correctamente. |

Stage-level estados

| Estado de la etapa | Description (Descripción) |
| --- | --- |
| InProgress | La etapa se está ejecutando actualmente. |

| Estado de la etapa | Description (Descripción) |
| --- | --- |
| Detención | La ejecución de etapa se detiene debido a una solicitud para detener y esperar, o bien detener y abandonar, la ejecución de canalización. |
| Stopped | El proceso de detención se ha completado y la ejecución de etapa se detiene. |
| Realizado correctamente | La etapa ha finalizado correctamente. |
| Con error | La etapa no ha finalizado correctamente. |

Action-level estados

| Estado de la acción | Description (Descripción) |
| --- | --- |
| InProgress | La acción se está ejecutando actualmente. |
| Abandonados | La acción se abandona debido a una solicitud de detener y abandonar la ejecución de canalización. |
| Realizado correctamente | La acción ha finalizado correctamente. |
| Con error | Para las acciones de aprobación, el estado FAILED significa que la acción fue rechazada por el revisor o que ha tenido un error debido a una configuración incorrecta de la acción. |

## Ver una ejecución entrante (consola)

Puede utilizar la consola de para ver el estado de y los detalles de una ejecución entrante. Cuando la transición está habilitada o la etapa pasa a estar disponible, una ejecución entrante `InProgress` continúa y entra en la etapa. Una ejecución entrante con un estado `Stopped` no entra en la etapa. El estado de una ejecución entrante cambia a `Failed` si se edita la canalización. Al editar una canalización, todas las ejecuciones en curso no continúan y el estado de la ejecución cambia a `Failed`.

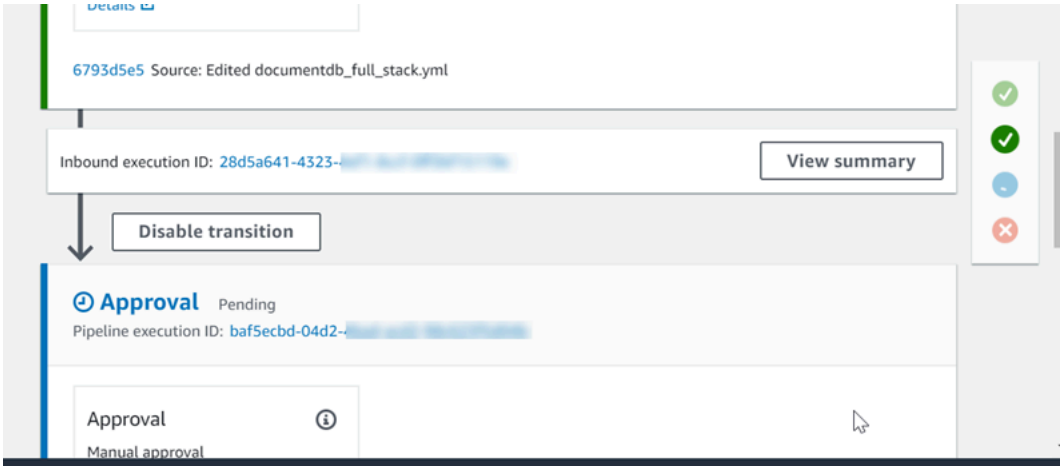
Si no ve ninguna ejecución entrante, significa que no hay ninguna ejecución pendiente en una fase de transición desactivada.

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se mostrarán los nombres de todas las canalizaciones asociadas a su AWS cuenta.

2. Elija el nombre de la canalización para la que quiere ver la ejecución de entrada. Realice una de las siguientes acciones:

   - Elija View (Ver). En el diagrama de canalización, en el campo ID de ejecución de entrada situado delante de la transición deshabilitada, puede ver el ID de ejecución entrante.




     Seleccione Ver resumen para ver los detalles de la ejecución, como el ID de ejecución, el desencadenador de origen y el nombre de la siguiente etapa.

   - Seleccione la canalización y elija Ver historial.

## Consulta de las revisiones de código fuente de las ejecuciones de una canalización (consola)

Puede ver los detalles sobre los artefactos de código fuente (artefacto de salida originado en la primera etapa de una canalización) que se utilizan en una ejecución de una canalización. Estos detalles incluyen identificadores, como los identificadores de confirmaciones, los comentarios de protección y, si se utiliza la CLI, los números de versión de las acciones de compilación de la canalización. Para algunos tipos de revisión, puede ver y abrir la URL de la confirmación. Las revisiones de código fuente constan de la siguiente información:

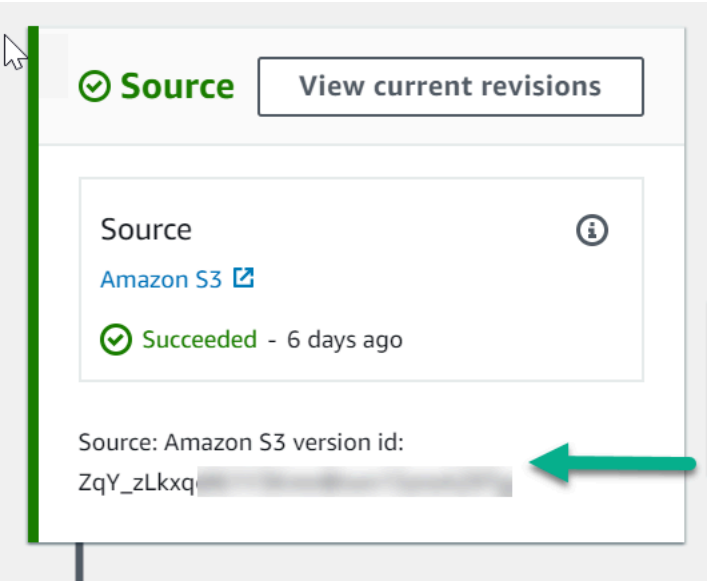
- Resumen: información resumida sobre la revisión más reciente del artefacto. Para GitHub los CodeCommit repositorios, el mensaje de confirmación. En los buckets o las acciones de Amazon S3, el contenido proporcionado por el usuario de una clave codepipeline-artifact-revision-summary especificada en los metadatos del objeto.
- revisionUrl: la URL de revisión para la revisión del artefacto (por ejemplo, la URL del repositorio externo).
- revisionId: el ID de revisión de la revisión del artefacto. Por ejemplo, para un cambio de fuente en un GitHub repositorio CodeCommit o, este es el ID de confirmación. En el caso de los artefactos almacenados en CodeCommit repositorios GitHub o repositorios, el ID de confirmación está vinculado a una página de detalles de la confirmación.

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

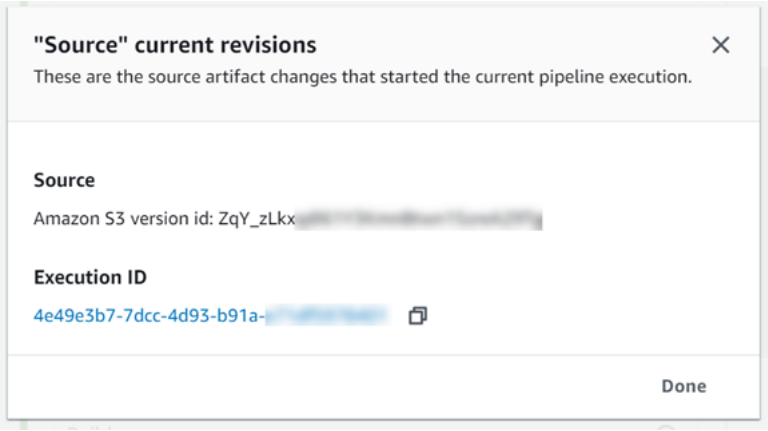
   Se mostrarán los nombres de todas las canalizaciones asociadas Cuenta de AWS a la suya.

2. Elija el nombre de la canalización cuyos detalles de revisión de origen desea ver. Realice una de las siguientes acciones:

   - Seleccione Ver historial. En Source revisions (Revisiones de código fuente) aparece el cambio en el código fuente para cada ejecución.
   - Ubique una acción cuyos detalles de revisión de origen desee ver y después encuentre la información de revisión al final de su etapa:



Elija Ver revisiones actuales para ver la información de origen. Con excepción de los artefactos almacenados en buckets de Amazon S3, los identificadores como ID de confirmación en esta vista de detalles de la información se vinculan con las páginas de información de origen de los artefactos.




## Consulta de las ejecuciones de una acción (consola)

Puede ver los detalles de una acción para una canalización, como, por ejemplo, el ID de ejecución, los artefactos de entrada, los artefactos de salida y el estado de la acción. Para ver los detalles de una acción, elija una canalización en la consola y, a continuación, elija un ID de ejecución.

1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres de todas las canalizaciones asociadas a su AWS cuenta.
2. Elija el nombre de la canalización para la que desea ver los detalles de una acción y, a continuación, elija View history (Ver historial).
3. En Execution ID (ID de ejecución), elija el ID de ejecución para el que desea ver los detalles de ejecución de una acción.
4. Puede ver la siguiente información en la pestaña Timeline (Línea temporal):

   a. En Action name (Nombre de acción), elija el enlace para abrir una página de detalles para la acción en la que puede ver el estado, el nombre de la etapa, el nombre de la acción, datos de configuración e información sobre los artefactos.

   b. En Provider (Proveedor), elija el enlace para ver los detalles del proveedor de la acción. Por ejemplo, en el ejemplo anterior, si elige entre CodeDeploy las etapas de preparación

o producción, se muestra la página de CodeDeploy consola de la CodeDeploy aplicación configurada para esa etapa.

## Consulta de artefactos de acción y de información del almacén de artefactos (consola)

Puede ver los detalles de los artefactos de entrada y de salida de una acción. También puede elegir un enlace que le lleva a la información de los artefactos de dicha acción. Dado que el almacén de artefactos utiliza el control de versiones, cada ejecución de la acción tiene una única ubicación de artefactos de entrada y de salida.

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres de todas las canalizaciones asociadas a su AWS cuenta.
2. Elija el nombre de la canalización para la que desea ver los detalles de una acción y, a continuación, elija View history (Ver historial).
3. En Execution ID (ID de ejecución), elija el ID de ejecución para el que desea ver los detalles de una acción.
4. En la pestaña Timeline (Línea temporal), en Action name (Nombre de acción), elija el enlace para abrir una página de detalles de la acción.
5. En la página de detalles, en Ejecución, vea el estado y los tiempos de la ejecución de la acción.
6. En la pestaña Configuración, consulta la configuración de recursos para la acción (por ejemplo, el nombre del proyecto de CodeBuild creación).
7. En la pestaña Artefactos, vea los detalles de los artefactos en Tipo de artefacto y en Proveedor de artefactos. Elija el enlace situado debajo de Artifact name (Nombre de artefacto) para ver los artefactos del almacén de artefactos.
8. En la pestaña Variables de salida, consulte las variables resueltas de las acciones en proceso para la ejecución de la acción.

## Visualización de los detalles y el historial de la canalización (CLI)

Puede ejecutar los siguientes comandos para ver los detalles acerca de sus canalizaciones y las ejecuciones de canalizaciones:

- list-pipelinescomando para ver un resumen de todas las canalizaciones asociadas a tu Cuenta de AWS.
- get-pipeline para repasar los detalles de una canalización individual.
- list-pipeline-executions para ver resúmenes de las ejecuciones más recientes de una canalización.
- get-pipeline-execution para ver información sobre la ejecución de una canalización, como los detalles de los artefactos, el ID de ejecución de la canalización y el nombre, la versión y el estado de la canalización.
- get-pipeline-state para ver el estado de la canalización, de las etapas y de las acciones.
- list-action-executions para ver los detalles de ejecución de las acciones de una canalización.

Temas


- Consulta el historial de ejecuciones con list-pipeline-executions (CLI)
- Vea el estado de la canalización con get-pipeline-state (CLI)
- Consulta el estado de la ejecución entrante con get-pipeline-state (CLI)
- Consulta las revisiones del estado y del código fuente con get-pipeline-execution (CLI)
- Vea las ejecuciones de acciones con list-action-executions (CLI)


## Consulta el historial de ejecuciones con  list-pipeline-executions (CLI)

Puede ver el historial de ejecución de una canalización.

- Para ver detalles acerca de las ejecuciones anteriores de una canalización, ejecute el comando list-pipeline-executions especificando el nombre único de la canalización. Por ejemplo, para ver detalles acerca del estado actual de una canalización denominada *MyFirstPipeline*, escriba lo siguiente:

  ```
  aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline
  ```

  Este comando devuelve información de resumen acerca de todas las ejecuciones de la canalización cuyo historial se ha registrado. El resumen incluye las horas de inicio y finalización, la duración y el estado.

  Las ejecuciones de canalización que se hayan revertido mostrarán el tipo de ejecución `Rollback`. En el caso de la ejecución fallida que desencadenó la reversión automática, se muestra el identificador de la ejecución fallida.

El siguiente ejemplo muestra los datos devueltos de una canalización denominada *MyFirstPipeline* que ha tenido tres ejecuciones:

```
{
    "pipelineExecutionSummaries": [
        {
            "pipelineExecutionId": "eb7ebd36-353a-4551-90dc-18ca5EXAMPLE",
            "status": "Succeeded",
            "startTime": "2024-04-16T09:00:28.185000+00:00",
            "lastUpdateTime": "2024-04-16T09:00:29.665000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "revision_ID",
                    "revisionSummary": "Added README.txt",
                    "revisionUrl": "console-URL"
                }
            ],
            "trigger": {
                "triggerType": "StartPipelineExecution",
                "triggerDetail": "trigger_ARN"
            },
            "executionMode": "SUPERSEDED"
        },
        {
            "pipelineExecutionId": "fcd61d8b-4532-4384-9da1-2aca1EXAMPLE",
            "status": "Succeeded",
            "startTime": "2024-04-16T08:58:56.601000+00:00",
            "lastUpdateTime": "2024-04-16T08:59:04.274000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "revision_ID",
                    "revisionSummary": "Added README.txt",
                    "revisionUrl": "console_URL"
                }
            ],
            "trigger": {
                "triggerType": "StartPipelineExecution",
                "triggerDetail": "trigger_ARN"
            },
            "executionMode": "SUPERSEDED"
```

```
        }
```

Para ver más detalles acerca de una ejecución de canalización, ejecute get-pipeline-execution especificando el ID único de dicha ejecución. Por ejemplo, para ver más detalles acerca de la primera ejecución en el ejemplo anterior, escriba lo siguiente:

```
aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
execution-id 7cf7f7cb-3137-539g-j458-d7eu3EXAMPLE
```

Este comando devuelve un resumen de la información acerca de la ejecución de una canalización, como los detalles de los artefactos, el ID de ejecución de la canalización y el nombre, la versión y el estado de la canalización.

El siguiente ejemplo muestra los datos obtenidos para una canalización denominada *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"
            }
        ]
    }
}
```

## Vea el estado de la canalización con **get-pipeline-state (CLI)**

Puede usar la CLI para ver el estado de la canalización, de las etapas y de las acciones.

- Para ver detalles acerca del estado actual de una canalización, ejecute el comando get-pipeline-state especificando el nombre único de la canalización. Por ejemplo, para ver detalles acerca del estado actual de una canalización denominada *MyFirstPipeline*, escriba lo siguiente:

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

Este comando devuelve el estado actual de todas las etapas de la canalización y el estado de las acciones en esas etapas.

El siguiente ejemplo muestra los datos obtenidos para una canalización de tres etapas denominada *MyFirstPipeline*, donde las primeras dos etapas y acciones se realizan correctamente, la tercera presenta errores y la transición entre la segunda y la tercera etapa está deshabilitada:

```
{
    "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"
        }
    ]
}
```

## Consulta el estado de la ejecución entrante con get-pipeline-state (CLI)

Puede usar la CLI para ver el estado de la ejecución entrante. Cuando la transición está habilitada o la etapa pasa a estar disponible, una ejecución entrante `InProgress` continúa y entra en la etapa. Una ejecución entrante con un estado `Stopped` no entra en la etapa. El estado de una ejecución entrante cambia a `Failed` si se edita la canalización. Al editar una canalización, todas las ejecuciones en curso no continúan y el estado de la ejecución cambia a `Failed`.

- Para ver detalles acerca del estado actual de una canalización, ejecute el comando get-pipeline-state especificando el nombre único de la canalización. Por ejemplo, para ver detalles acerca del estado actual de una canalización denominada *`MyFirstPipeline`*, escriba lo siguiente:

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

  Este comando devuelve el estado actual de todas las etapas de la canalización y el estado de las acciones en esas etapas. El resultado también muestra el ID de ejecución de la canalización en cada etapa y si hay un ID de ejecución entrante para una etapa con una transición deshabilitada.

  El siguiente ejemplo muestra los datos devueltos para una canalización en dos etapas denominada*`MyFirstPipeline`*, donde la primera etapa muestra una transición habilitada y una ejecución de canalización exitosa, y la segunda etapa, denominada`Beta`, muestra una transición deshabilitada y un identificador de ejecución entrante. La ejecución entrante puede tener un estado `InProgress`, `Stopped` o `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
}
```

## Consulta las revisiones del estado y del código fuente con get-pipeline-execution (CLI)

Puede ver los detalles sobre los artefactos de código fuente (artefactos de salida originados en la primera etapa de una canalización) que se utilizan en una ejecución de una canalización. Estos detalles incluyen identificadores, como los identificadores de confirmaciones, los comentarios de protección, el tiempo transcurrido desde la creación o la actualización del artefacto, cuándo se utiliza la CLI o los números de versión de las acciones de creación. Para algunos tipos de revisión, puede ver y abrir la URL de la confirmación de la versión del artefacto. Las revisiones de código fuente constan de la siguiente información:

- Resumen: información resumida sobre la revisión más reciente del artefacto. Para los AWS CodeCommit repositorios GitHub , el mensaje de confirmación. En los buckets o las acciones de Amazon S3, el contenido proporcionado por el usuario de una clave codepipeline-artifact-revision-summary especificada en los metadatos del objeto.
- revisionUrl: el ID de confirmación de la revisión del artefacto. En el caso de los artefactos almacenados en AWS CodeCommit repositorios GitHub o repositorios, el identificador de confirmación está vinculado a una página de detalles de la confirmación.

Puede ejecutar el comando get-pipeline-execution para ver información sobre las revisiones de código fuente más recientes incluidas en una ejecución de canalización. Ejecute en primer lugar el comando get-pipeline-state para ver los detalles de todas las etapas de una canalización y determinar el ID de ejecución de la etapa cuyos detalles de revisión de código fuente desea obtener. A continuación, utilice el ID de ejecución en el comando get-pipeline-execution. (Dado que es posible que las etapas de una canalización se hayan completado correctamente en diferentes ejecuciones de la canalización, pueden tener ID de ejecución distintos).

En otras palabras, si desea ver los detalles de los artefactos que se encuentran en la etapa Staging (Ensayo), ejecute el comando get-pipeline-state, determine cuál es el ID de ejecución actual de la etapa de ensayo y ejecute el comando get-pipeline-execution usando ese ID de ejecución.

Para ver estados y revisiones de origen en una canalización

1. Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows) y utilice la AWS CLI para ejecutar el comando get-pipeline-state. Para una canalización denominada *MyFirstPipeline*, debe escribir:

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

   Este comando devuelve el estado más reciente de una canalización, incluyendo el último ID de ejecución de canalización para cata etapa.

2. Para ver los detalles sobre una ejecución de una canalización, ejecute el comando get-pipeline-execution especificando el nombre único de la canalización y el ID de la ejecución de la canalización de cuyos artefactos desea ver los detalles. Por ejemplo, para ver detalles sobre la ejecución de una canalización denominada *MyFirstPipeline* con el ID de ejecución 3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE, debe escribir lo siguiente:

   ```
   aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
   execution-id 3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE
   ```

   Este comando devuelve información acerca de cada revisión de origen que es parte de la ejecución de canalización y de la información de identificación acerca de la canalización. Solo se incluye información acerca de las etapas de canalización que se incluyeron en esa ejecución. Puede que haya otras etapas en la canalización que no fueron parte de esa ejecución de canalización.

   En el siguiente ejemplo, se muestran los datos devueltos por una parte de la canalización denominada*MyFirstPipeline*, en la que un artefacto denominado "MyApp" se almacena en un GitHub repositorio:

3. 
   ```
   {
       "pipelineExecution": {
           "artifactRevisions": [
               {
                   "created": 1427298837.7689769,
                   "name": "MyApp",
   ```

```
                "revisionChangeIdentifier": "1427298921.3976923",
                "revisionId": "7636d59f3c461cEXAMPLE8417dbc6371",
                "revisionSummary": "Updating the application for feature 12-4820",
                "revisionUrl": "https://api.github.com/repos/anycompany/MyApp/git/
commits/7636d59f3c461cEXAMPLE8417dbc6371"
            }
        ],
        "pipelineExecutionId": "3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE",
        "pipelineName": "MyFirstPipeline",
        "pipelineVersion": 2,
        "status": "Succeeded",
        "executionMode": "SUPERSEDED",
        "executionType": "ROLLBACK",
        "rollbackMetadata": {
            "rollbackTargetPipelineExecutionId": "4f47bed9-6998-476c-a49d-
e60beEXAMPLE"
         }
    }
}
```

## Vea las ejecuciones de acciones con  list-action-executions (CLI)

Puede ver los detalles de ejecución de las acciones de una canalización, como, por ejemplo, el ID de ejecución de la acción, los artefactos de entrada, los artefactos de salida, el resultado de la ejecución y el estado. Debe proporcionar el filtro de ID de ejecución si desea obtener un listado con las acciones de una ejecución de la canalización:

> **Note**
>
> Hay un historial de ejecución detallado disponible para las ejecuciones realizadas el día 21 de febrero de 2019 o con posterioridad.

- Para ver las ejecuciones de las acciones de una canalización, realice una de las siguientes operaciones:
  - Para ver detalles sobre todas las ejecuciones de las acciones de una canalización, ejecute el comando list-action-executions especificando el nombre único de la canalización. Por ejemplo, para ver las ejecuciones de las acciones de una canalización denominada *MyFirstPipeline*, escriba lo siguiente:

```
aws codepipeline list-action-executions --pipeline-name MyFirstPipeline
```

A continuación, se muestra una parte de la salida de ejemplo de este comando:

```
{
    "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"
                }
            }
        },

. . .
```

- Para ver todas las ejecuciones de las acciones de una ejecución de una canalización, ejecute el comando list-action-executions y especifique el nombre único de la canalización y el ID de la ejecución. Por ejemplo, para ver las ejecuciones de acciones de un*Execution_ID*, introduzca lo siguiente:

```
aws codepipeline list-action-executions --pipeline-name MyFirstPipeline --filter
 pipelineExecutionId=Execution_ID
```

- A continuación, se muestra una parte de la salida de ejemplo de este comando:

```
{
    "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"
```

```
                    }
                ]
            },


. . .
```

## Configuración o cambio del modo de ejecución de una canalización

Es posible configurar el modo de ejecución de una canalización para especificar cómo se gestionan las ejecuciones múltiples.

Para obtener más información acerca de los modos de ejecución de una canalización, consulte Cómo funcionan las ejecuciones de canalización.

**Important**

En el caso de las canalizaciones en modo PARALELO, al editar el modo de ejecución de la canalización en el modo EN COLA o SUPERSEDED, el estado de la canalización no mostrará el estado actualizado como PARALELO. Para obtener más información, consulte Las canalizaciones que cambien del modo PARALELO a otro mostrarán un modo de ejecución anterior.

**Important**

Para las canalizaciones en modo PARALELO, al editar el modo de ejecución de la canalización en el modo EN COLA o SUPERSEDED, la definición de canalización para la canalización en cada modo no se actualizará. Para obtener más información, consulte Las canalizaciones en modo PARALELO tienen una definición de canalización desactualizada si se editan al cambiar al modo EN COLA o SUPERSEDED.

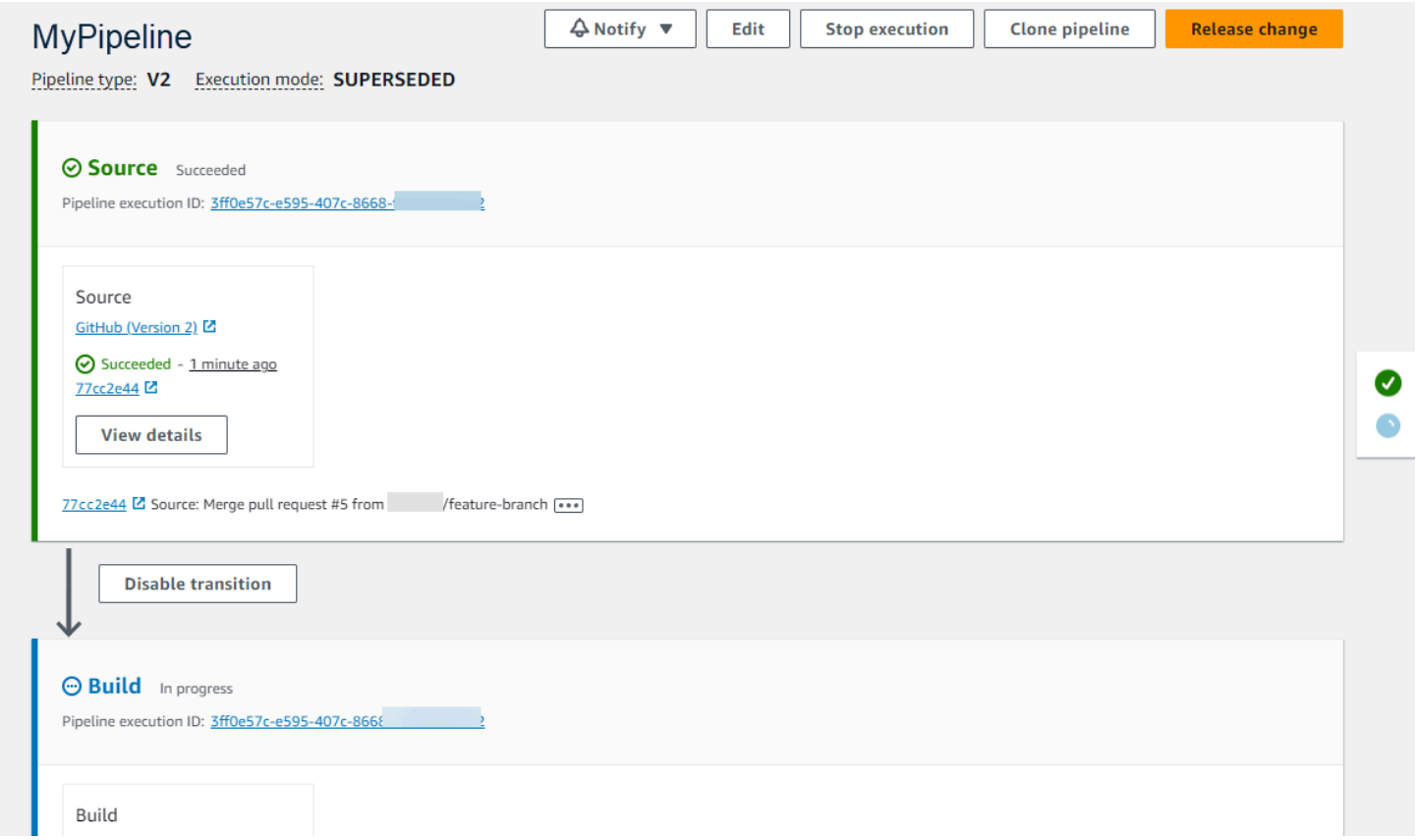
> **Important**
>
> Para las canalizaciones en modo PARALELO, la reversión de etapas no está disponible. Del mismo modo, las condiciones de error con un tipo de resultado de reversión no se pueden agregar a una canalización en modo PARALELO.

## Consideraciones para la visualización de los modos de ejecución

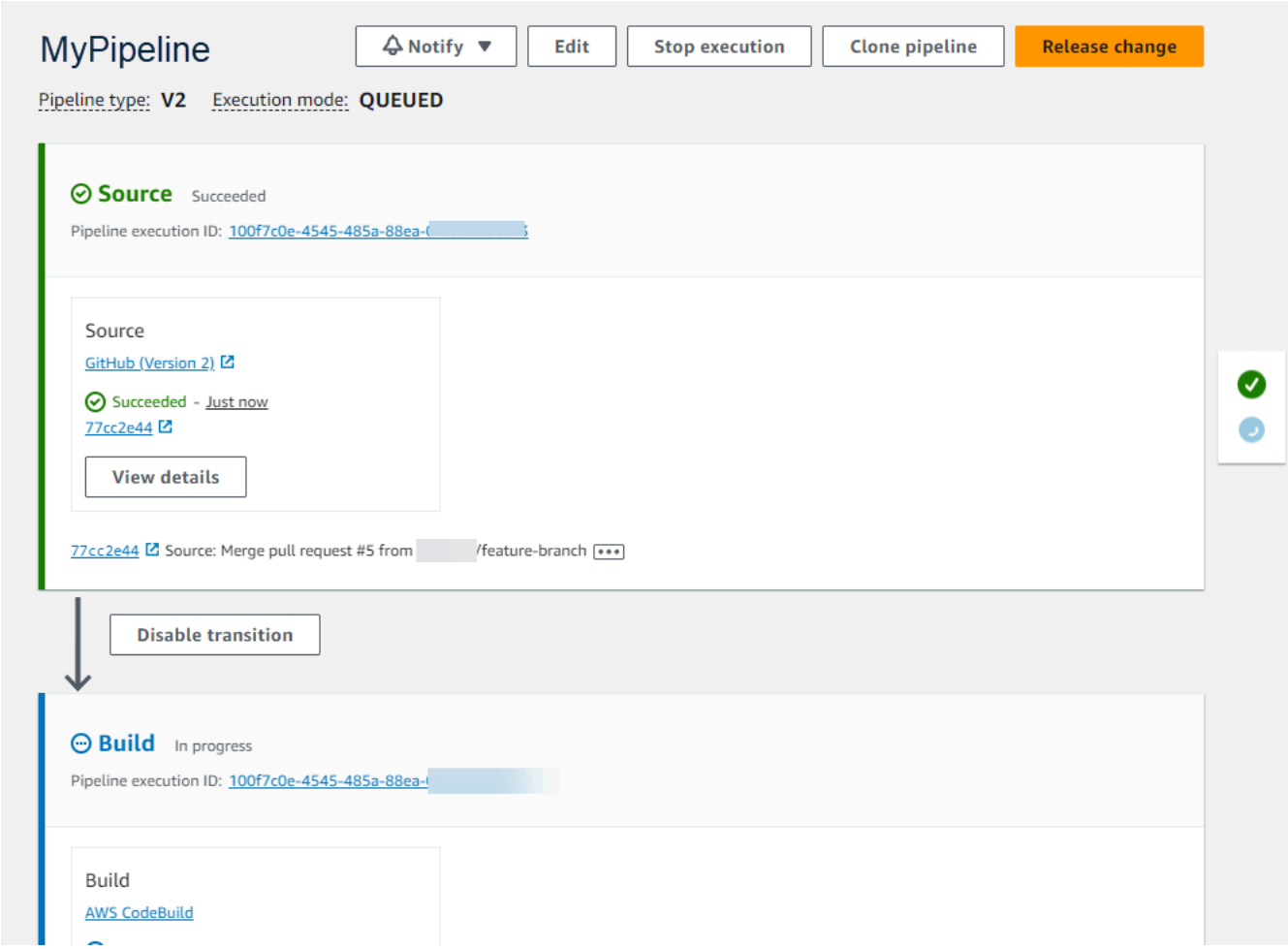
Hay que tener en cuenta varios aspectos para visualizar canalizaciones en modos de ejecución específicos.

En los modos SUPERSEDED y EN COLA, utilice la vista de canalización para ver las ejecuciones en curso, y haga clic en el identificador de la ejecución para ver los detalles y el historial. En el modo PARALELO, haga clic en el identificador de la ejecución para ver la ejecución en curso en la pestaña Visualización.

A continuación se muestra la vista del modo SUPERSEDED en. CodePipeline

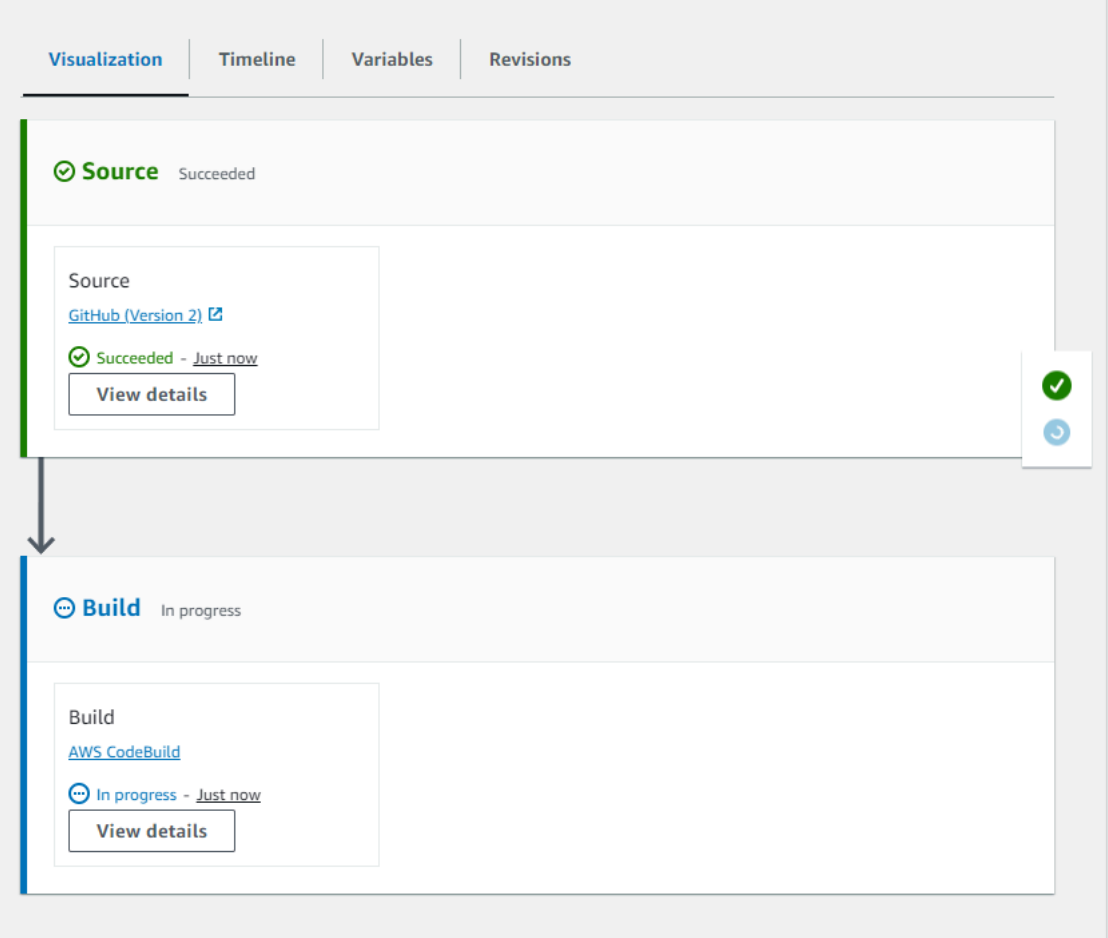



A continuación se muestra la vista del modo QUEUED en CodePipeline.




A continuación se muestra la vista del modo PARALLEL en CodePipeline.

> **Important**
>
> Para las canalizaciones en modo PARALELO, la reversión de etapas no está disponible. Del mismo modo, las condiciones de error con un tipo de resultado de reversión no se pueden agregar a una canalización en modo PARALELO.




## Consideraciones para cambiar de un modo de ejecución a otro

Hay que tener en cuenta los siguientes aspectos sobre las canalizaciones a la hora de cambiar de un modo de canalización a otro. Si se cambia de un modo de ejecución a otro en el modo Editar y, a continuación, se guarda el cambio, es posible que determinadas vistas o estados se ajusten.

Por ejemplo, al cambiar del modo PARALELO al modo EN COLA o SUPERSEDED, la ejecución iniciada en el modo PARALELO seguirá ejecutándose. Estas ejecuciones se pueden ver en la página del historial de ejecución. La vista de canalización mostrará la ejecución que se ejecutó anteriormente en los modos EN COLA o SUPERSEDED, o bien en un estado vacío.

Como otro ejemplo, al cambiar del modo QUEUED o SUPERSEDED al modo PARALLEL, ya no verás la página de canalización. view/state Para ver una ejecución en modo PARALELO, utilice la pestaña Visualización de la página de detalles de la ejecución. Las ejecuciones que se inicien en modo SUPERSEDED o QUEUED se cancelarán.

En la siguiente tabla se proporcionan más detalles.

| Cambiar modo | Detalles de las ejecuciones pendientes y activas | Detalles del estado de la canalización |
| --- | --- | --- |
| SUPERSEDED A SUPERSEDED / SUPERSEDED A EN COLA | • Las ejecuciones activas se cancelan una vez finalizadas las acciones en curso.<br>• Se cancelan las ejecuciones pendientes. | El estado de canalización, por ejemplo cancelado, se conserva entre la versión del primer modo y la del segundo modo. |
| EN COLA A EN COLA / EN COLA A SUPERSEDED | • Las ejecuciones activas se cancelan una vez finalizadas las acciones en curso.<br>• Se cancelan las ejecuciones pendientes. | El estado de canalización, por ejemplo cancelado, se conserva entre la versión del primer modo y la del segundo modo. |
| PARALELO a PARALELO | Se permite que todas las ejecuciones se ejecuten independientemente de las actualizaciones de las definiciones de canalización. | Vacío. El modo paralelo no tiene ningún estado de canalización. |
| SUPERSEDED A PARALELO / EN COLA A PARALELO | • Las ejecuciones activas se cancelan una vez finalizadas las acciones en curso.<br>• Se cancelan las ejecuciones pendientes. | Vacío. El modo paralelo no tiene ningún estado de canalización. |

## Configuración o cambio del modo de ejecución de una canalización (consola)

Puede utilizar la consola para configurar el modo de ejecución de una canalización.

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres y el estado de todas las canalizaciones asociadas a su AWS cuenta.

2. En Nombre, elija el nombre de la canalización que desea editar.
3. En la página de detalles de la canalización, elija Edit.
4. En la página Editar, seleccione Editar: propiedades de la canalización.
5. Elija el modo de la canalización.
   - Superseded
   - En cola (se requiere una canalización de tipo V2)
   - Paralelo (se requiere una canalización de tipo V2)
6. En la página Editar, elija Listo.

## Configuración del modo de ejecución de una canalización (CLI)

AWS CLI Para configurar el modo de ejecución de la canalización, utilice el `update-pipeline` comando `create-pipeline` o.

1. Abra una sesión de terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows) y ejecute el comando get-pipeline para copiar la estructura de canalización en un archivo JSON. Por ejemplo, para una canalización llamada **`MyFirstPipeline`**, escriba el siguiente comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando no devuelve nada, pero el archivo creado debería aparecer en el directorio en el que se ejecutó el comando.
2. Abra el archivo JSON en cualquier editor de texto sin formato y modifique la estructura del archivo para que refleje el modo de ejecución de la canalización que desee configurar, por ejemplo, el modo EN COLA.

   ```
   "executionMode": "QUEUED"
   ```

   El siguiente ejemplo muestra cómo configurar el modo de ejecución a EN COLA en una canalización de ejemplo con dos etapas.

   ```
   {
       "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. Si está trabajando con la estructura de la canalización recuperada mediante el comando get-pipeline, debe modificar la estructura del archivo JSON. Debe eliminar las líneas `metadata` del archivo para que el comando update-pipeline pueda utilizarlo. Quite la sección de la estructura de canalizaciones del archivo JSON (las líneas `"metadata": { }` y los campos `"created"`, `"pipelineARN"` y `"updated"` que contenga).

   Por ejemplo, quite las siguientes líneas de la estructura:

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

Guarde el archivo.

4. Para aplicar los cambios, ejecute el comando update-pipeline especificando el archivo JSON:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

Este comando devuelve la estructura completa de la canalización editada.

> **Note**
>
> El comando update-pipeline detiene la canalización. Si se está ejecutando una revisión en la canalización al ejecutar el comando update-pipeline, dicha ejecución se detiene. Debe comenzar la canalización actualizada manualmente para ejecutar dicha revisión en ella.

# Reversión o reintento de etapas

Puede usar la consola de AWS CodePipeline o la AWS CLI para revertir o volver a intentar manualmente una etapa o acciones de una etapa. Para configurar las condiciones de las etapas en las que desee utilizar la reversión o el reintento como resultado configurado, consulte Configuración de las condiciones de una etapa.

Temas


- Configuración del reintento de una etapa fallida o de acciones fallidas
- Configuración de la reversión de etapas


## Configuración del reintento de una etapa fallida o de acciones fallidas

Puede reintentar una etapa que haya fallado sin tener que volver a ejecutar una canalización desde el principio. Para ello, puede volver a intentar las acciones fallidas en una etapa o volver a intentar todas las acciones de la etapa empezando por la primera acción de la misma. Al volver a intentar las acciones fallidas en una etapa, todas las acciones que aún están en curso siguen funcionando y las fallidas se vuelven a activar. Al volver a intentar una etapa fallida desde la primera acción de la etapa, esta no puede tener ninguna acción en curso. Para poder volver a intentar una fase, es necesario que todas las acciones hayan fallado o que algunas hayan fallado y otras se hayan realizado correctamente.

> **Important**
>
> Al volver a intentar una etapa fallida, se vuelven a intentar todas las acciones de la etapa desde la primera acción de la etapa y, al reintentar las acciones fallidas, se reintentan todas las acciones fallidas de la etapa. Esto anula los artefactos de salida de las acciones que anteriormente se habían realizado correctamente en la misma ejecución.
> Aunque los artefactos se pueden anular, se conserva el historial de ejecución de las acciones que anteriormente se realizaron correctamente.

Si utiliza la consola para ver una canalización, aparece el botón Reintentar etapa o el botón Reintentar acciones fallidas donde las acciones con errores se pueden reintentar.

Si usa la AWS CLI, puede usar el get-pipeline-state comando para determinar si alguna acción ha fallado.

> **Note**
>
> En los casos siguientes, es posible que no se pueda reintentar las acciones:
>
> - Todas las acciones de la etapa se realizaron correctamente, por lo que la etapa no se encuentra en estado fallido.
> - La estructura global de la canalización cambió después de producirse el error en la etapa.
> - Ya se está intentando otro reintento en la etapa.

Temas


- Consideraciones sobre el reintento de etapas
- Reintento de una etapa fallida manualmente
- Configuración de una etapa para el reintento automático en caso de fallo


## Consideraciones sobre el reintento de etapas

Los aspectos sobre el reintento de etapas que se deben tener en cuenta son los siguientes:

- Solo se puede configurar el reintento automático en etapas fallidas para un reintento.
- Se puede configurar el reintento automático en etapas fallidas para todas las acciones, incluidas las acciones `Source`.

## Reintento de una etapa fallida manualmente

Puede reintentar manualmente una etapa fallida mediante la consola o la CLI.

También puede configurar una etapa para que se reintente automáticamente en caso de fallo, como se detalla en Configuración de una etapa para el reintento automático en caso de fallo.

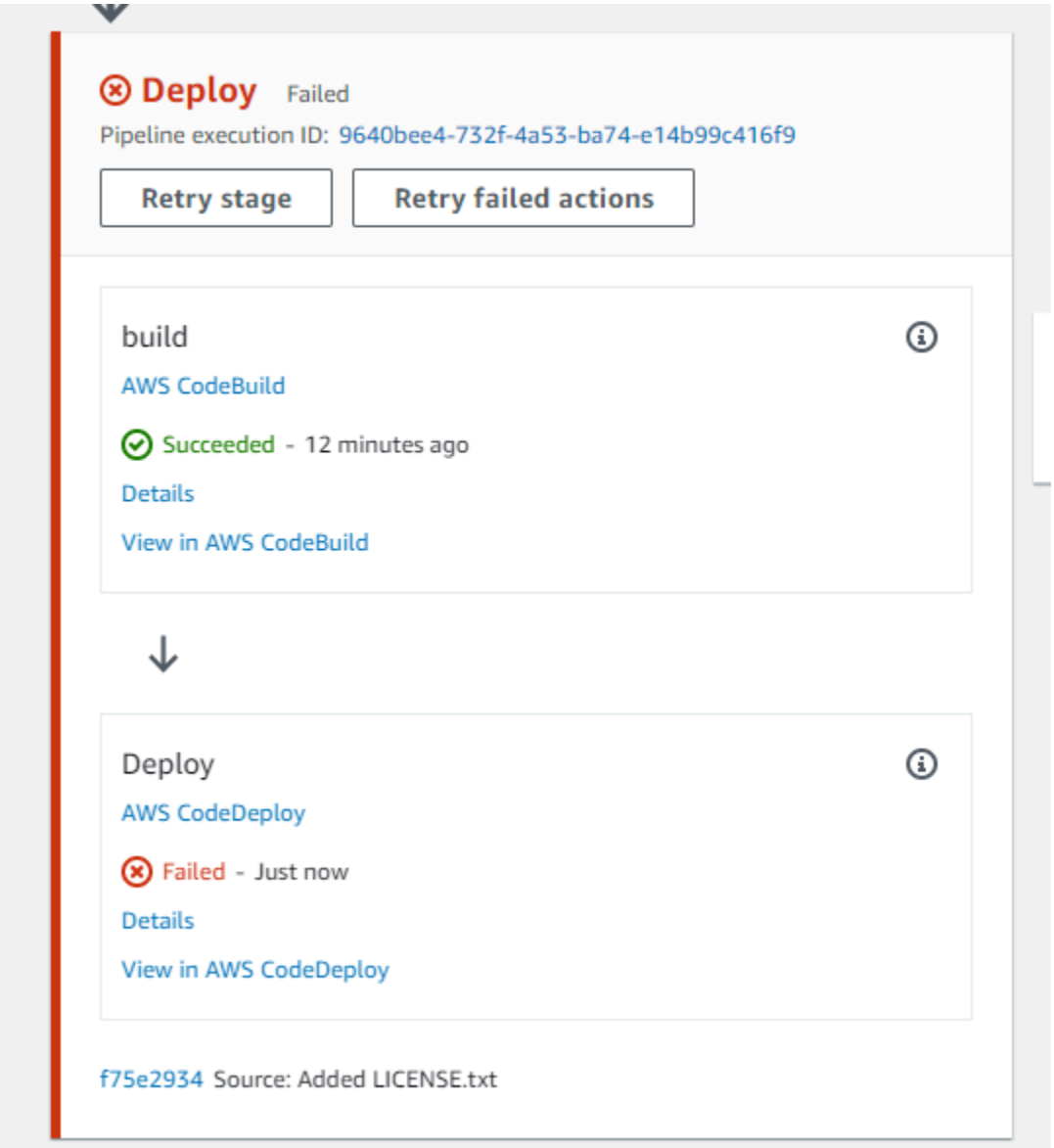
## Reintento de una etapa fallida manualmente (consola)

Para volver a intentar una etapa fallida o realizar acciones fallidas en una etapa (consola)

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres de todas las canalizaciones asociadas a su AWS cuenta.
2. En Name, elija el nombre de la canalización.
3. Ubique la etapa con la acción que ha dado error y después elija una de las siguientes opciones.

   - Para volver a intentar todas las acciones de la etapa, seleccione Reintentar etapa.
   - Para volver a intentar solo las acciones fallidas de la fase, seleccione Reintentar acciones fallidas.




Si todas las acciones que se vuelven a intentar en la etapa se completan correctamente, la canalización se sigue ejecutando.

## Reintento de una etapa fallida manualmente (CLI)

Para volver a intentar una etapa fallida o realizar acciones fallidas en una etapa - CLI

Para volver AWS CLI a intentar todas las acciones o todas las acciones fallidas, ejecuta el retry-stage-execution comando con los siguientes parámetros:

```
--pipeline-name <value>
--stage-name <value>
--pipeline-execution-id <value>
--retry-mode ALL_ACTIONS/FAILED_ACTIONS
```

> **Note**
>
> Los valores que puede utilizar para `retry-mode` son `FAILED_ACTIONS` y `ALL_ACTIONS`.

1. En una terminal (Linux, macOS o Unix) o el símbolo del sistema (Windows), ejecute el comando retry-stage-execution, tal y como se muestra en el siguiente ejemplo para una canalización denominada `MyPipeline`.

   ```
   aws codepipeline retry-stage-execution --pipeline-name MyPipeline --stage-name
    Deploy --pipeline-execution-id b59babff-5f34-EXAMPLE --retry-mode FAILED_ACTIONS
   ```

   La salida devuelve el ID de ejecución:

   ```
   {
       "pipelineExecutionId": "b59babff-5f34-EXAMPLE"
   }
   ```

2. También puede ejecutar el comando con un archivo de entrada JSON. Primero debe crear un archivo JSON que identifique la canalización, la etapa que incluye las acciones fallidas y la ejecución más reciente de la canalización en esa etapa. Ejecute el comando retry-stage-execution con el parámetro `--cli-input-json`. Para recuperar los detalles que necesita para el archivo JSON, es más fácil usar el comando get-pipeline-state.

   a. En un terminal (Linux, macOS o Unix) o símbolo del sistema (Windows), ejecute el comando get-pipeline-state en una canalización. Por ejemplo, para una canalización denominada MyFirstPipeline, escribirías algo similar a lo siguiente:

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

      La respuesta al comando incluye información de estado de la canalización para cada etapa. En el siguiente ejemplo, la respuesta indica que una o más acciones han fallado en la etapa de ensayo (Staging):

```
{
    "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. En un editor de texto sin formato, cree un archivo en el que registrará la siguiente información en formato JSON:

- El nombre de la canalización que incluye las acciones fallidas
- El nombre de la etapa que incluye las acciones fallidas
- El ID de la última ejecución de la canalización en la etapa
- El modo de volver a intentarlo.

En el MyFirstPipeline ejemplo anterior, el archivo tendría un aspecto similar al siguiente:

```
{
    "pipelineName": "MyFirstPipeline",
    "stageName": "Staging",
    "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
    "retryMode": "FAILED_ACTIONS"
```

```
}
```

c. Guarde el archivo con un nombre como **`retry-failed-actions.json`**.

d. Llame el archivo que creó al ejecutar el comando retry-stage-execution. Por ejemplo:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

```
aws codepipeline retry-stage-execution --cli-input-json file://retry-failed-
actions.json
```

e. Para ver los resultados del reintento, abre la CodePipeline consola y elige la canalización que contiene las acciones fallidas o vuelve a usar el get-pipeline-state comando. Para obtener más información, consulte Vea las canalizaciones y los detalles en CodePipeline.

## Configuración de una etapa para el reintento automático en caso de fallo

Puede configurar una etapa para el reintento automático en caso de fallo. La etapa realizará un reintento y mostrará el estado del reintento en la etapa fallida en la página Ver de la canalización.

Para configurar el modo de reintento, especifique que la etapa debe reintentar automáticamente todas las acciones en la etapa fallida, o bien solo las acciones fallidas en la etapa.

### Configuración de una etapa para el reintento automático en caso de fallo (consola)

Puede utilizar la consola para configurar un reintento automático en una etapa.

Configuración de una etapa para el reintento automático (consola)

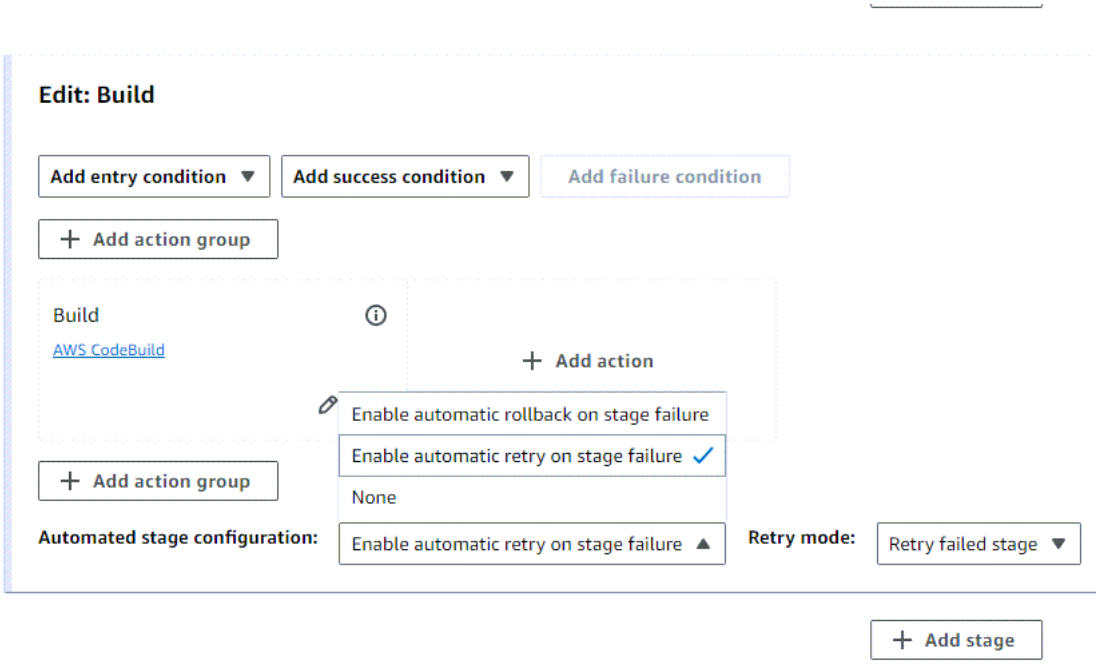
1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres y el estado de todas las canalizaciones asociadas a su AWS cuenta.
2. En Nombre, elija el nombre de la canalización que desea editar.
3. En la página de detalles de la canalización, elija Edit.
4. En la página Editar, para la acción que desee editar, elija Editar etapa.

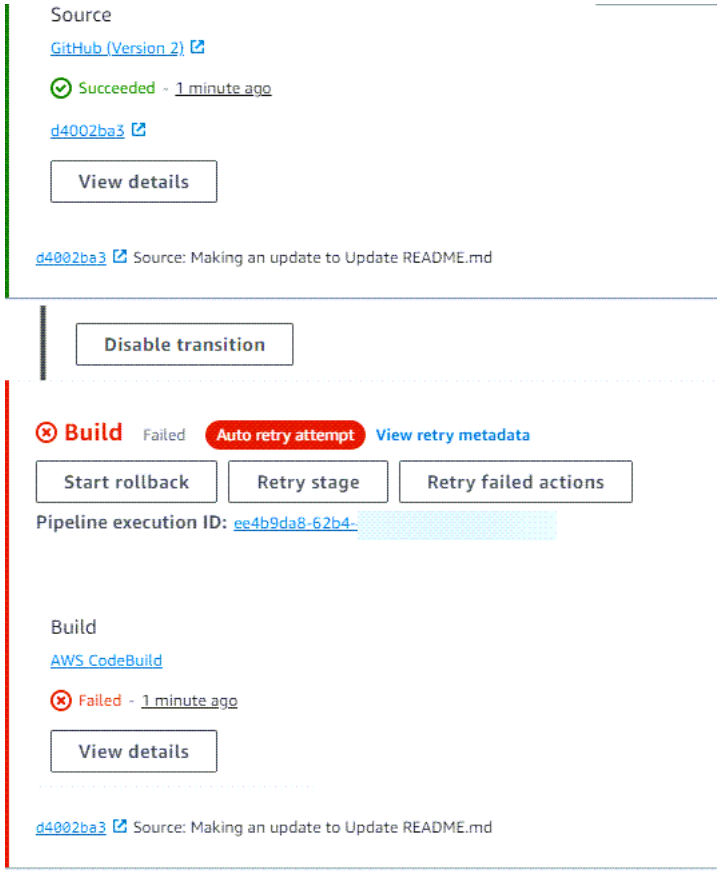
5. Seleccione Configuración automatizada de escenarios y, a continuación, seleccione Habilitar el reintento automático en caso de fallo de una etapa. Guarde los cambios en la canalización.
6. En Configuración automatizada de escenarios elija uno de los siguientes modos de reintento:

   - Para especificar que el modo reintentará todas las acciones en la etapa, seleccione Reintentar etapa fallida.
   - Para especificar que el modo reintentará solo las acciones fallidas en la etapa, seleccione Reintentar acciones fallidas.

   Guarde los cambios en la canalización.




7. Una vez ejecutada la canalización, si se produce un error en la etapa, se realizará el reintento automático. Los siguientes ejemplos muestran una etapa de compilación que se ha reintentado automáticamente.




8. Para ver los detalles del reintento, elija. Aparecerá la ventana.




## Configuración de una etapa para el reintento automático (CLI)

Para configurar una fase AWS CLI para que vuelva a intentarlo automáticamente en caso de error, utilice los comandos para crear o actualizar una canalización, tal y como se detalla en Creación de una canalización, etapas y acciones y. Editar una canalización en CodePipeline

- Abra un terminal (Linux, macOS o Unix) o el símbolo del sistema (Windows) y utilice la AWS CLI para ejecutar el comando `update-pipeline`, especificando la condición de fallo en la

estructura de canalización. En el siguiente ejemplo se configura el reintento automático para una etapa denominada `S3Deploy`:

```
{
                "name": "S3Deploy",
                "actions": [
                    {
                        "name": "s3deployaction",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "provider": "S3",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "BucketName": "static-website-bucket",
                            "Extract": "false",
                            "ObjectKey": "SampleApp.zip"
                        },
                        "outputArtifacts": [],
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "region": "us-east-1"
                    }
                ],
                 "onFailure": {
                    "result": "RETRY",
                    "retryConfiguration": {
                        "retryMode": "ALL_ACTIONS",
                    },
            }
```

## Configure una etapa para el reintento automático (CloudFormation)

Para configurar una fase CloudFormation para que se reintente automáticamente en caso de error, utilice el parámetro del ciclo de vida de la `OnFailure` etapa. Utilice el parámetro `RetryConfiguration` para configurar el modo de reintento.

```
OnFailure:
     Result: RETRY
     RetryConfiguration:
         RetryMode: ALL_ACTIONS
```

- Actualice la plantilla como se muestra en el fragmento de código siguiente. En el siguiente ejemplo se configura el reintento automático para una etapa denominada `Release`:

```
AppPipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
    RoleArn:
      Ref: CodePipelineServiceRole
    Stages:
      -
        Name: Source
        Actions:
          -
            Name: SourceAction
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
              -
                Name: SourceOutput
            Configuration:
              S3Bucket:
                Ref: SourceS3Bucket
              S3ObjectKey:
                Ref: SourceS3ObjectKey
            RunOrder: 1
      -
        Name: Release
        Actions:
          -
            Name: ReleaseAction
            InputArtifacts:
              -
                Name: SourceOutput
            ActionTypeId:
```

```
            Category: Deploy
            Owner: AWS
            Version: 1
            Provider: CodeDeploy
          Configuration:
            ApplicationName:
              Ref: ApplicationName
            DeploymentGroupName:
              Ref: DeploymentGroupName
          RunOrder: 1
     OnFailure:
         Result: RETRY
         RetryConfiguration:
            RetryMode: ALL_ACTIONS
  ArtifactStore:
    Type: S3
    Location:
      Ref: ArtifactStoreS3Location
    EncryptionKey:
      Id: arn:aws:kms:useast-1:ACCOUNT-ID:key/KEY-ID
      Type: KMS
  DisableInboundStageTransitions:
    -
      StageName: Release
      Reason: "Disabling the transition until integration tests are completed"
  Tags:
    - Key: Project
      Value: ProjectA
    - Key: IsContainerBased
      Value: 'true'
```

Para obtener más información sobre la configuración del reintento por etapas en caso de error, consulte la OnFailure sección siguiente de la `StageDeclaration` Guía del CloudFormation usuario.

## Configuración de la reversión de etapas

Puede revertir una etapa a una ejecución que se haya realizado correctamente en esa etapa. Puede preconfigurar una etapa para revertirla en caso de fallo, o bien puede revertir una etapa manualmente. La operación de reversión provocará una nueva ejecución. La ejecución de la

canalización de destino elegida para la reversión se utiliza para recuperar las variables y revisiones de origen.

El tipo de ejecución, ya sea estándar o de reversión, se muestra en el historial de canalizaciones, en el estado de la canalización y en los detalles de la ejecución de la canalización.

Temas


- Consideraciones sobre las reversiones
- Reversión manual de una etapa
- Configuración de una etapa para la reversión automática
- Visualización del estado de reversión en la lista de ejecuciones
- Visualización de los detalles del estado de reversión


## Consideraciones sobre las reversiones

Los aspectos sobre la reversión de etapas que se deben tener en cuenta son los siguientes:

- No se puede revertir una etapa de origen.
- La canalización solo puede revertirse a una ejecución anterior si dicha ejecución anterior se inició en la versión actual de la estructura de canalización.
- No se puede revertir a un identificador de ejecución de destino que sea de un tipo de ejecución de reversión.
- CodePipeline utilizará las variables y los artefactos de la ejecución a la que está retrocediendo.

## Reversión manual de una etapa

Puede revertir una etapa manualmente mediante la consola o la CLI. La canalización solo puede revertirse a una ejecución anterior si dicha ejecución anterior se inició en la versión actual de la estructura de canalización.

También puede configurar una etapa para que se revierta automáticamente en caso de fallo, como se detalla en Configuración de una etapa para la reversión automática.

# Reversión manual de una etapa (consola)

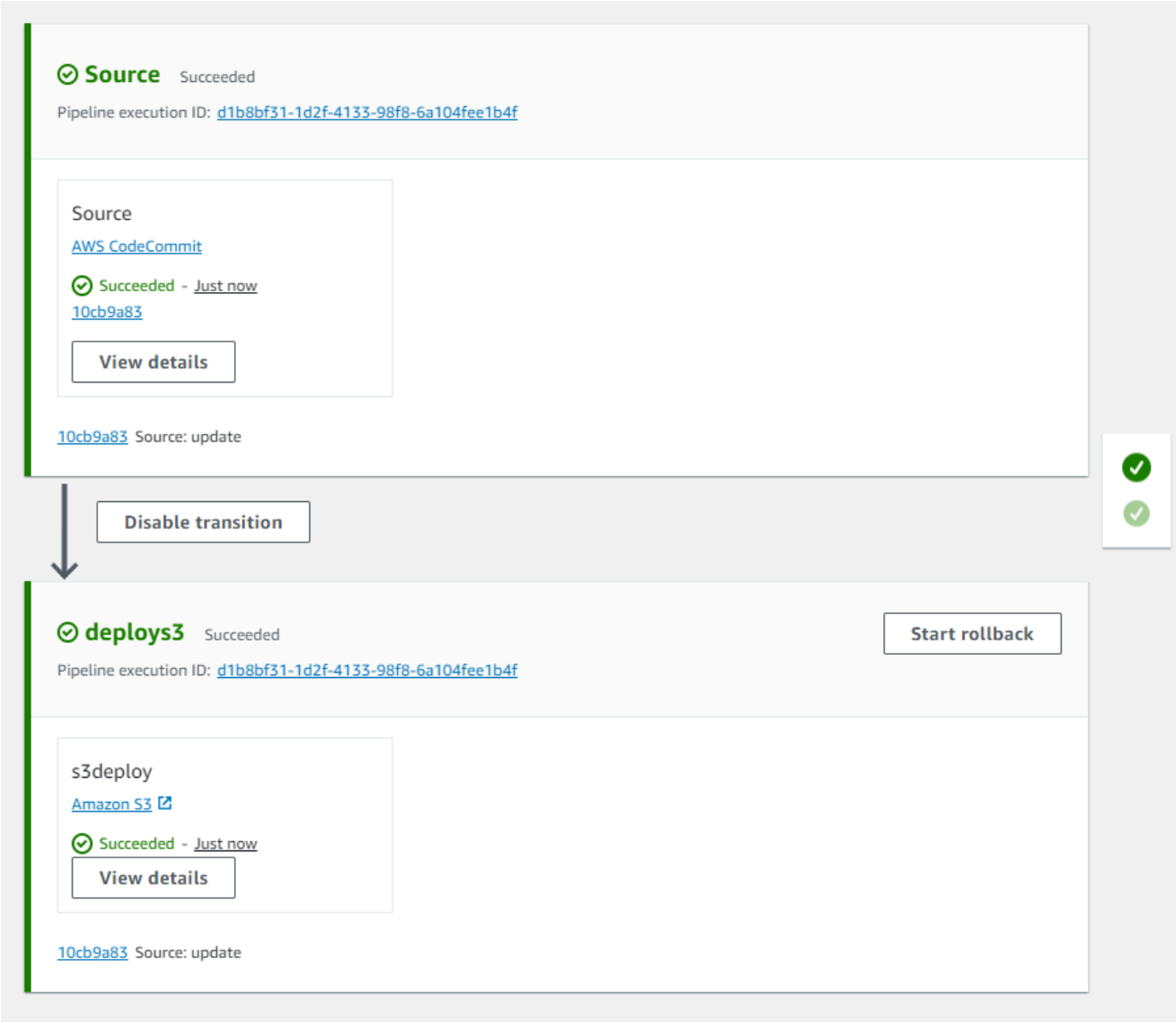
Puede utilizar la consola para revertir manualmente una etapa a una ejecución de canalización de destino. Cuando se revierte una etapa, se muestra una etiqueta Reversión en la visualización de la canalización de la consola.

Reversión manual de una etapa (consola)

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

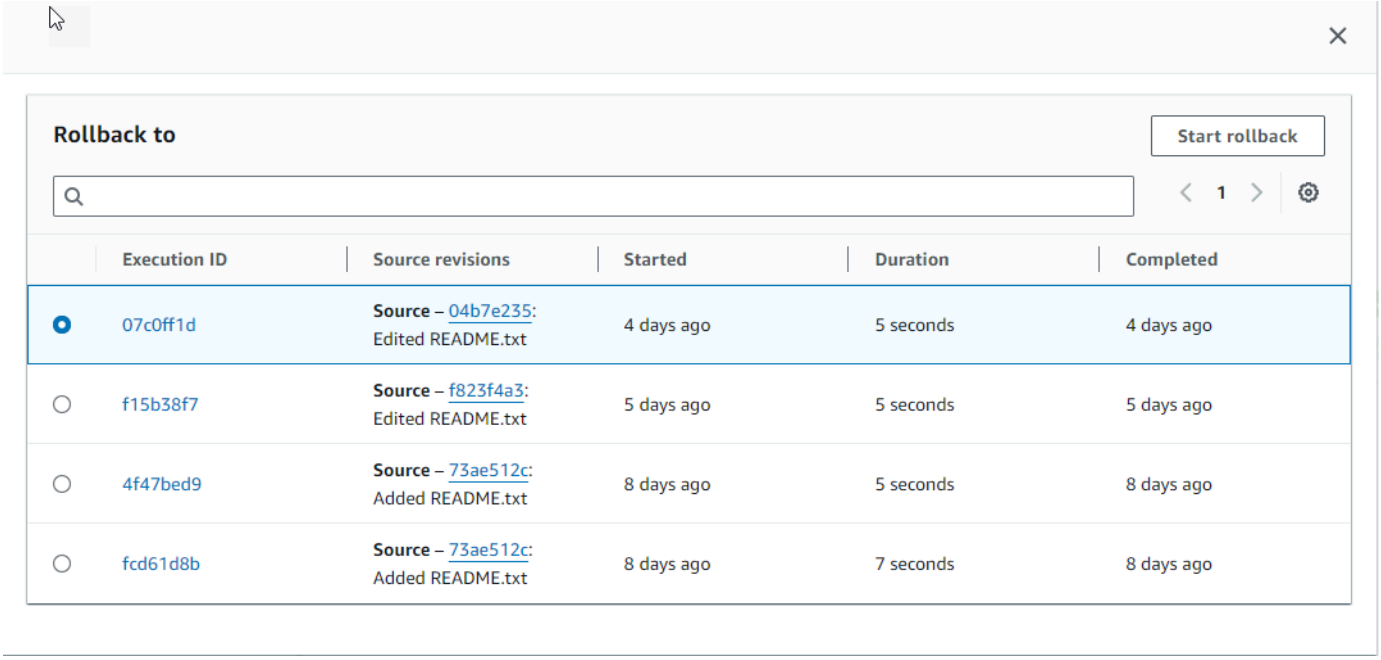
   Se muestran los nombres y el estado de todas las canalizaciones asociadas a su AWS cuenta.

2. En Nombre, elija el nombre de la canalización con la etapa que se va a revertir.

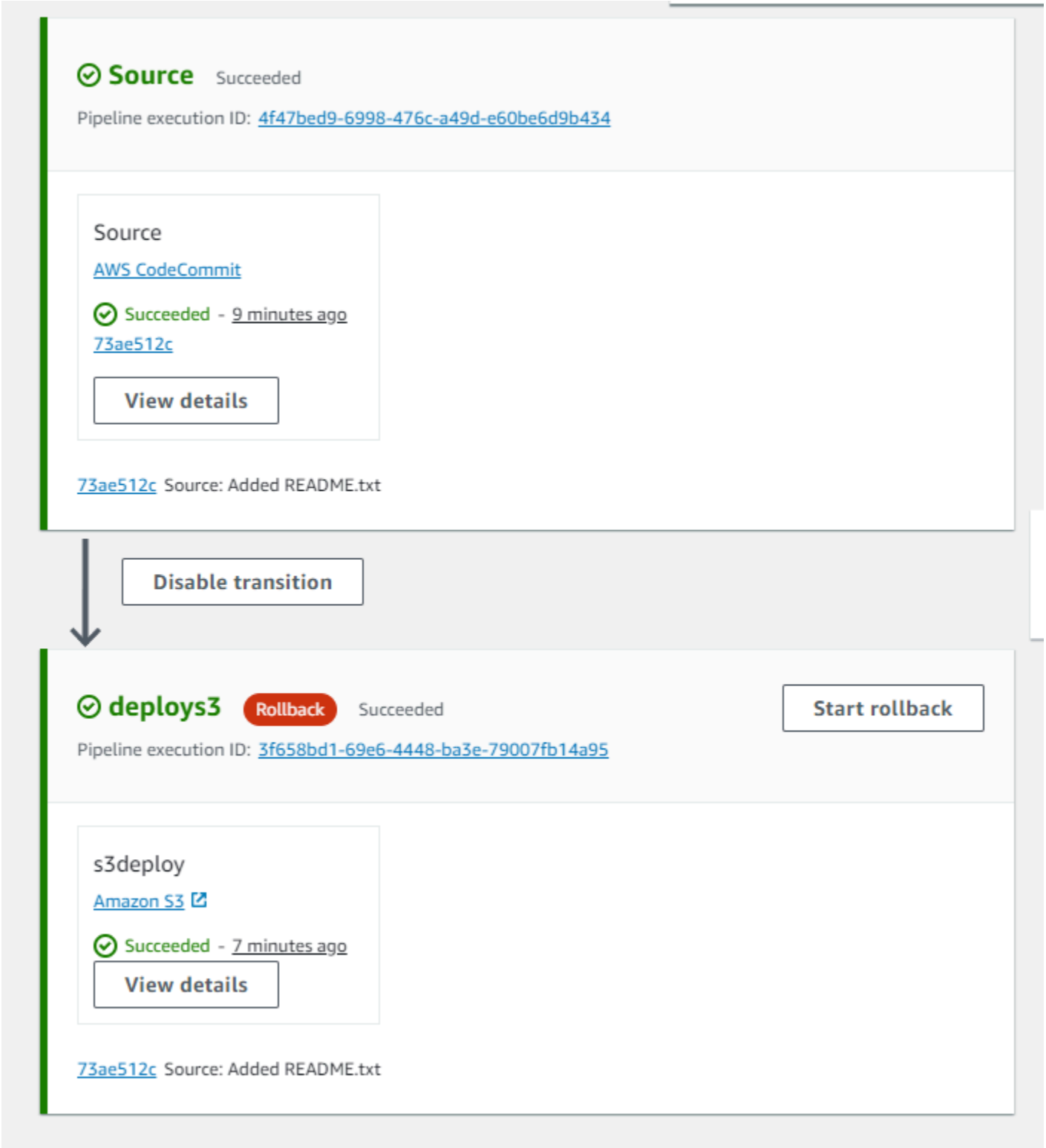



3. En la etapa, seleccione Iniciar la reversión. Aparecerá la página Reversión a.
4. Elija la ejecución de destino a la que desee revertir la etapa.

> **Note**
>
> En la lista de ejecuciones de canalización de destino disponibles se incluirán todas las ejecuciones de la versión actual de la canalización a partir del 1 de febrero de 2024.




En el siguiente diagrama, se muestra un ejemplo de la etapa de reversión con el nuevo identificador de ejecución.




## Reversión manual de una etapa (CLI)

Para utilizar el AWS CLI para hacer retroceder manualmente una fase, utilice el `rollback-stage` comando.

También puede revertir una etapa de forma manual como se detalla en Reversión manual de una etapa.

> **Note**
>
> En la lista de ejecuciones de canalización de destino disponibles se incluirán todas las ejecuciones de la versión actual de la canalización a partir del 1 de febrero de 2024.

Para revertir una etapa manualmente (CLI)

1. El comando de la CLI para la reversión manual requerirá el identificador de ejecución de una ejecución de canalización correcta realizada previamente en la etapa. Para obtener el identificador de ejecución de la canalización de destino que va a especificar, utilice el comando list-pipeline-executions con un filtro que devolverá las ejecuciones correctas en la etapa. Abra una terminal (Linux, macOS o Unix) o una línea de comandos (Windows) y utilícela AWS CLI para ejecutar el `list-pipeline-executions` comando, especificando el nombre de la canalización y el filtro para que se ejecute correctamente en la fase. En este ejemplo, la salida mostrará una lista de las ejecuciones en canalización de la canalización denominada MyFirstPipeline y de las ejecuciones satisfactorias en la etapa denominada`deploys3`.

   ```
   aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline --filter
    succeededInStage={stageName=deploys3}
   ```

   En el resultado, copie el identificador de ejecución de la ejecución correcta realizada previamente que desee especificar para la reversión. Lo utilizará en el siguiente paso como el identificador de ejecución de destino.

2. Abra un terminal (Linux, macOS o Unix) o el símbolo del sistema (Windows) y utilice la AWS CLI para ejecutar el comando `rollback-stage`, especificando el nombre de la canalización, el nombre de la etapa y la ejecución de destino a la que desee realizar la reversión. Por ejemplo, para revertir una etapa denominada Deploy para una canalización denominada*`MyFirstPipeline`*:

   ```
   aws codepipeline rollback-stage --pipeline-name MyFirstPipeline --stage-name Deploy
    --target-pipeline-execution-id bc022580-4193-491b-8923-9728dEXAMPLE
   ```

El resultado devuelve el identificador de ejecución de la nueva ejecución revertida. Se trata de un identificador independiente que utiliza las revisiones de origen y los parámetros de la ejecución de destino especificada.

## Configuración de una etapa para la reversión automática

Puede configurar las etapas de una canalización para revertirlas automáticamente en caso de fallo. Cuando la etapa falla, se revierte a la ejecución correcta más reciente. La canalización solo puede revertirse a una ejecución anterior si dicha ejecución anterior se inició en la versión actual de la estructura de canalización. Como la configuración de la reversión automática forma parte de la definición de la canalización, la etapa de canalización solo se revertirá automáticamente después de que se haya ejecutado correctamente la canalización en la etapa de canalización.

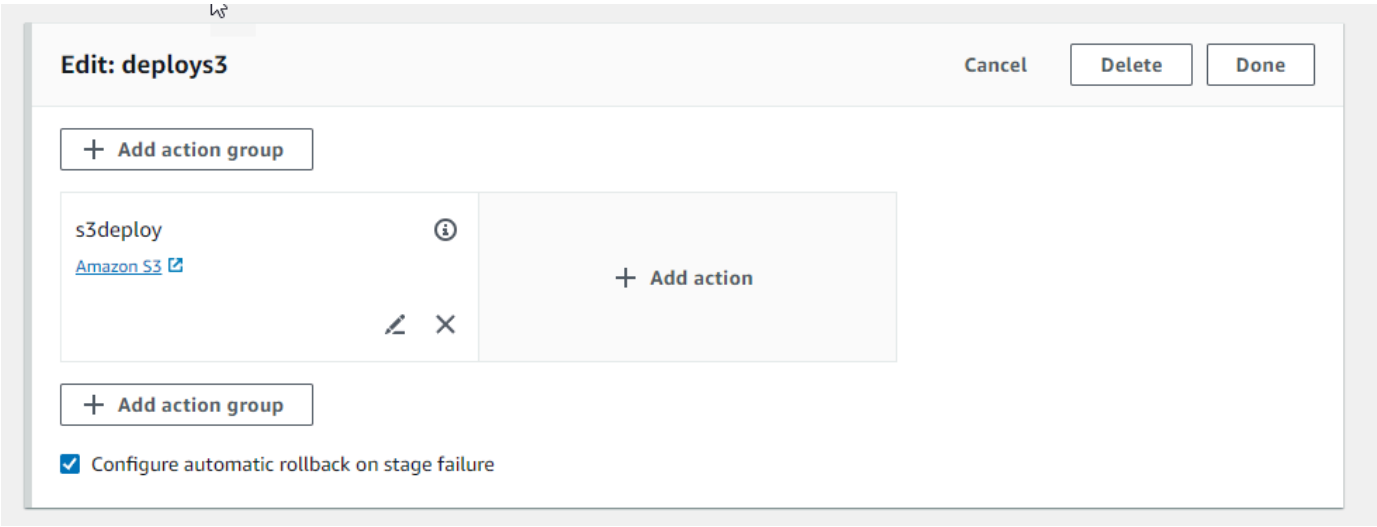
### Configuración de una etapa para la reversión automática (consola)

Puede revertir una etapa a una ejecución anterior que se realizara correctamente. Para obtener más información, consulta la Guía de RollbackStage API. CodePipeline

Configuración de una etapa para la reversión automática (consola)

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres y el estado de todas las canalizaciones asociadas a su AWS cuenta.
2. En Nombre, elija el nombre de la canalización que desea editar.
3. En la página de detalles de la canalización, elija Edit.
4. En la página Editar, para la acción que desee editar, elija Editar etapa.
5. Seleccione Configuración automatizada de escenarios y, a continuación, seleccione Configurar la reversión automática en caso de fallo en una etapa. Guarde los cambios en la canalización.




## Configuración de una etapa para la reversión automática (CLI)

AWS CLI Para configurar una fase fallida y volver automáticamente a la ejecución correcta más reciente, utilice los comandos para crear o actualizar una canalización, tal y como se detalla en Creación de una canalización, etapas y acciones yEditar una canalización en CodePipeline.

- Abra un terminal (Linux, macOS o Unix) o el símbolo del sistema (Windows) y utilice la AWS CLI para ejecutar el comando `update-pipeline`, especificando la condición de fallo en la estructura de canalización. El siguiente ejemplo configura la reversión automática para una etapa denominada: `S3Deploy`

```
{
                "name": "S3Deploy",
                "actions": [
                    {
                        "name": "s3deployaction",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "provider": "S3",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "BucketName": "static-website-bucket",
                            "Extract": "false",
                            "ObjectKey": "SampleApp.zip"
```

```
                    },
                    "outputArtifacts": [],
                    "inputArtifacts": [
                        {
                            "name": "SourceArtifact"
                        }
                    ],
                    "region": "us-east-1"
                }
            ],
            "onFailure": {
                "result": "ROLLBACK"
            }
        }
```

Para obtener más información sobre la configuración de las condiciones de error para la reversión por etapas, consulte la referencia de FailureConditions la CodePipeline API.

## Configure una etapa para la reversión automática (CloudFormation)

Para configurar una etapa CloudFormation para que retroceda automáticamente en caso de error, utilice el `OnFailure` parámetro. En caso de fallo, la etapa se revertirá automáticamente a la ejecución correcta más reciente.

```
OnFailure:
     Result: ROLLBACK
```

- Actualice la plantilla como se muestra en el fragmento de código siguiente. El siguiente ejemplo configura la reversión automática para una etapa denominada: `Release`

```
AppPipeline:
  Type: AWS::CodePipeline::Pipeline
  Properties:
    RoleArn:
      Ref: CodePipelineServiceRole
    Stages:
      -
        Name: Source
        Actions:
          -
            Name: SourceAction
```

```
            ActionTypeId:
              Category: Source
              Owner: AWS
              Version: 1
              Provider: S3
            OutputArtifacts:
              -
                Name: SourceOutput
            Configuration:
              S3Bucket:
                Ref: SourceS3Bucket
              S3ObjectKey:
                Ref: SourceS3ObjectKey
            RunOrder: 1
      -
        Name: Release
        Actions:
          -
            Name: ReleaseAction
            InputArtifacts:
              -
                Name: SourceOutput
            ActionTypeId:
              Category: Deploy
              Owner: AWS
              Version: 1
              Provider: CodeDeploy
            Configuration:
              ApplicationName:
                Ref: ApplicationName
              DeploymentGroupName:
                Ref: DeploymentGroupName
            RunOrder: 1
       OnFailure:
            Result: ROLLBACK
    ArtifactStore:
      Type: S3
      Location:
        Ref: ArtifactStoreS3Location
      EncryptionKey:
        Id: arn:aws:kms:useast-1:ACCOUNT-ID:key/KEY-ID
        Type: KMS
    DisableInboundStageTransitions:
      -
```

```
        StageName: Release
        Reason: "Disabling the transition until integration tests are completed"
    Tags:
      - Key: Project
        Value: ProjectA
      - Key: IsContainerBased
        Value: 'true'
```

Para obtener más información sobre la configuración de las condiciones de error para la reversión por etapas, consulte la sección de la OnFailure Guía del `StageDeclaration` CloudFormation usuario.

## Visualización del estado de reversión en la lista de ejecuciones

Puede ver el estado y el identificador de ejecución de destino de una ejecución de reversión.

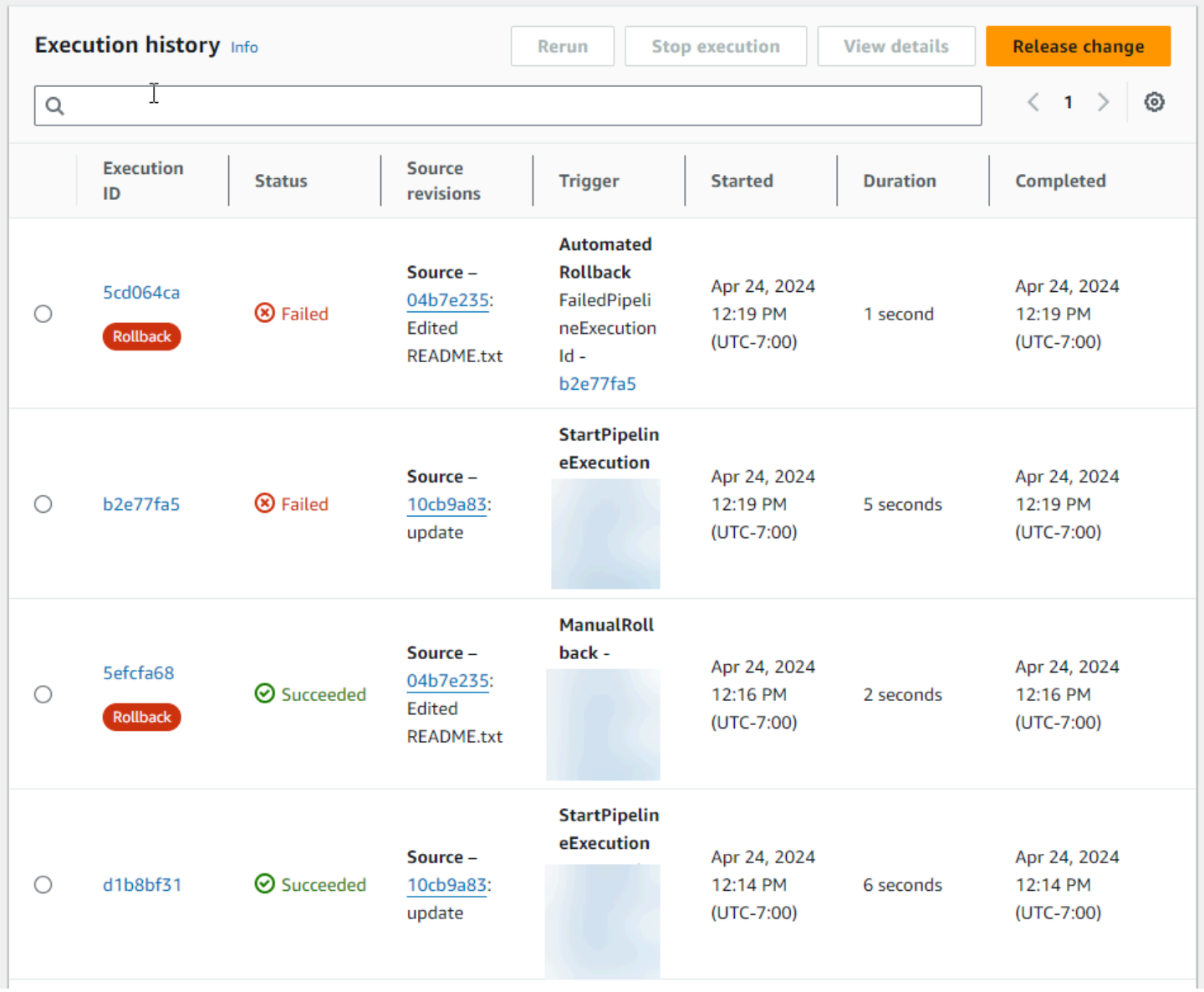
### Visualización del estado de reversión en la lista de ejecuciones (consola)

Puede utilizar la consola para ver el estado y el identificador de ejecución de destino de una ejecución de reversión en la lista de ejecuciones.

Visualización del estado de ejecución de reversión en la lista de ejecuciones (consola)

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

   Se muestran los nombres y estados de todas las canalizaciones asociadas con su Cuenta de AWS .
2. En Nombre, elija el nombre de la canalización que desee visualizar.
3. Elija Historial. La lista de ejecuciones muestra la etiqueta Reversión.




Elija el identificador de ejecución cuyos detalles desee ver.

## Consulta el estado de la reversión con list-pipeline-executions (CLI)

Puede utilizar la CLI para ver el estado y el identificador de ejecución de destino de una ejecución de reversión.

- Abra un terminal (Linux, macOS o Unix) o un símbolo del sistema (Windows) y utilice la AWS CLI para ejecutar el comando list-pipeline-executions.

```
aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline
```

Este comando devuelve una lista de todas las ejecuciones completadas asociadas con la canalización.

En el siguiente ejemplo, se muestran los datos devueltos de una canalización cuyo nombre es el *MyFirstPipeline* lugar en el que una ejecución de reversión inició la canalización.

```
{
    "pipelineExecutionSummaries": [
        {
            "pipelineExecutionId": "eb7ebd36-353a-4551-90dc-18ca5EXAMPLE",
            "status": "Succeeded",
            "startTime": "2024-04-16T09:00:28.185000+00:00",
            "lastUpdateTime": "2024-04-16T09:00:29.665000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "revision_ID",
                    "revisionSummary": "Added README.txt",
                    "revisionUrl": "console-URL"
                }
            ],
            "trigger": {
                "triggerType": "ManualRollback",
                "triggerDetail": "{arn:aws:sts::<account_ID>:assumed-role/<role>"}"
            },
            "executionMode": "SUPERSEDED",
            "executionType": "ROLLBACK",
            "rollbackMetadata": {
                "rollbackTargetPipelineExecutionId":
 "f15b38f7-20bf-4c9e-94ed-2535eEXAMPLE"
            }
        },
        {
            "pipelineExecutionId": "fcd61d8b-4532-4384-9da1-2aca1EXAMPLE",
            "status": "Succeeded",
            "startTime": "2024-04-16T08:58:56.601000+00:00",
            "lastUpdateTime": "2024-04-16T08:59:04.274000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "revision_ID",
                    "revisionSummary": "Added README.txt",
```

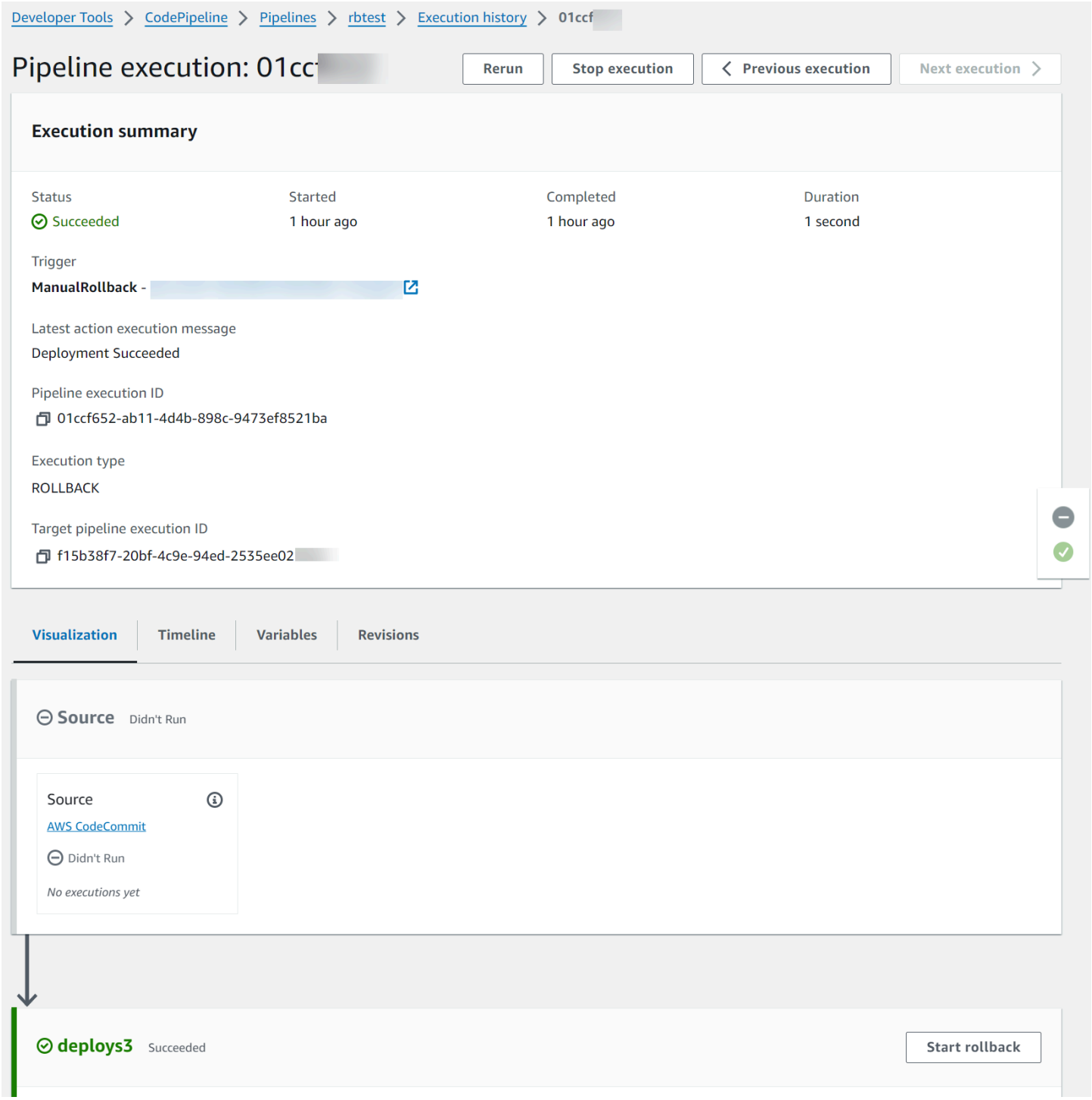
```
                    "revisionUrl": "console_URL"
                }
            ],
            "trigger": {
                "triggerType": "StartPipelineExecution",
                "triggerDetail": "arn:aws:sts::<account_ID>:assumed-role/<role>"
            },
            "executionMode": "SUPERSEDED"
        },
        {
            "pipelineExecutionId": "5cd064ca-bff7-425f-8653-f41d9EXAMPLE",
            "status": "Failed",
            "startTime": "2024-04-24T19:19:50.781000+00:00",
            "lastUpdateTime": "2024-04-24T19:19:52.119000+00:00",
            "sourceRevisions": [
                {
                    "actionName": "Source",
                    "revisionId": "<revision_ID>",
                    "revisionSummary": "Edited README.txt",
                    "revisionUrl": "<revision_URL>"
                }
            ],
            "trigger": {
                "triggerType": "AutomatedRollback",
                "triggerDetail": "{\"FailedPipelineExecutionId\":
\"b2e77fa5-9285-4dea-ae66-4389EXAMPLE\"}"
            },
            "executionMode": "SUPERSEDED",
            "executionType": "ROLLBACK",
            "rollbackMetadata": {
                "rollbackTargetPipelineExecutionId": "5efcfa68-d838-4ca7-
a63b-4a743EXAMPLE"
            }
         },
```

## Visualización de los detalles del estado de reversión

Puede ver el estado y el identificador de ejecución de destino de una ejecución de reversión.

# Visualización del estado de reversión en la página de detalles (consola)

Puede utilizar la consola para ver el estado y el identificador de ejecución de la canalización de destino de una ejecución de reversión.



## Consulta los detalles de la reversión con  get-pipeline-execution (CLI)

Las ejecuciones de canalización que se hayan revertido se mostrarán en el resultado para obtener la ejecución de la canalización.

- Para ver los detalles acerca de una canalización, ejecute el comando get-pipeline-execution especificando el nombre único de la canalización. Por ejemplo, para ver detalles acerca de una canalización denominada *MyFirstPipeline*, escriba lo siguiente:

```
aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
execution-id 3f658bd1-69e6-4448-ba3e-79007EXAMPLE
```

Este comando devuelve la estructura de la canalización.

En el siguiente ejemplo, se muestran los datos devueltos para una parte de una canalización denominada*MyFirstPipeline*, donde se muestran el identificador de ejecución de la reversión y los metadatos.

```
{
    "pipelineExecution": {
        "pipelineName": "MyFirstPipeline",
        "pipelineVersion": 6,
        "pipelineExecutionId": "2004a94e-8b46-4c34-a695-c8d20EXAMPLE",
        "status": "Succeeded",
        "artifactRevisions": [
            {
                "name": "SourceArtifact",
                "revisionId": "<ID>",
                "revisionSummary": "Added README.txt",
                "revisionUrl": "<console_URL>"
            }
        ],
        "trigger": {
            "triggerType": "ManualRollback",
            "triggerDetail": "arn:aws:sts::<account_ID>:assumed-role/<role>"
        },
        "executionMode": "SUPERSEDED",
        "executionType": "ROLLBACK",
        "rollbackMetadata": {
            "rollbackTargetPipelineExecutionId": "4f47bed9-6998-476c-a49d-
e60beEXAMPLE"
        }
```

```
    }
}
```

## Consulta el estado de la reversión con  get-pipeline-state (CLI)

Las ejecuciones de canalización que se hayan revertido se mostrarán en el resultado para obtener el estado de la canalización.

- Para ver los detalles acerca de una canalización, ejecute el comando get-pipeline-state especificando el nombre único de la canalización. Por ejemplo, para ver los detalles del estado de una canalización denominada*MyFirstPipeline*, introduce lo siguiente:

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

  En el siguiente ejemplo, se muestran los datos devueltos con el tipo de ejecución de reversión.

  ```
  {
      "pipelineName": "MyFirstPipeline",
      "pipelineVersion": 7,
      "stageStates": [
          {
              "stageName": "Source",
              "inboundExecutions": [],
              "inboundTransitionState": {
                  "enabled": true
              },
              "actionStates": [
                  {
                      "actionName": "Source",
                      "currentRevision": {
                          "revisionId": "<Revision_ID>"
                      },
                      "latestExecution": {
                          "actionExecutionId": "13bbd05d-
  b439-4e35-9c7e-887cb789b126",
                          "status": "Succeeded",
                          "summary": "update",
                          "lastStatusChange": "2024-04-24T20:13:45.799000+00:00",
                          "externalExecutionId": "10cbEXAMPLEID"
                      },
                      "entityUrl": "console-url",
  ```

```
                    "revisionUrl": "console-url"
                }
            ],
            "latestExecution": {
                "pipelineExecutionId": "cf95a8ca-0819-4279-ae31-03978EXAMPLE",
                "status": "Succeeded"
            }
        },
        {
            "stageName": "deploys3",
            "inboundExecutions": [],
            "inboundTransitionState": {
                "enabled": true
            },
            "actionStates": [
                {
                    "actionName": "s3deploy",
                    "latestExecution": {
                        "actionExecutionId":
 "3bc4e3eb-75eb-45b9-8574-8599aEXAMPLE",
                        "status": "Succeeded",
                        "summary": "Deployment Succeeded",
                        "lastStatusChange": "2024-04-24T20:14:07.577000+00:00",
                        "externalExecutionId": "mybucket/SampleApp.zip"
                    },
                    "entityUrl": "console-URL"
                }
            ],
            "latestExecution": {
                "pipelineExecutionId": "fdf6b2ae-1472-4b00-9a83-1624eEXAMPLE",
                "status": "Succeeded",
                "type": "ROLLBACK"
            }
        }
    ],
    "created": "2024-04-15T21:29:01.635000+00:00",
    "updated": "2024-04-24T20:12:24.480000+00:00"
}
```

# Configuración de las condiciones de una etapa

Puede especificar una condición para una etapa, como por ejemplo, comprobar si hay una variable específica en la ejecución de la canalización y, a continuación, obtener un resultado para la condición, como por ejemplo, omitir la etapa o fallar la etapa. Se puede configurar una canalización para comprobar las condiciones de una etapa durante la ejecución, en la que se deben especificar en primer lugar las comprobaciones de dicha etapa y, en segundo lugar, la forma en que debe continuar la etapa cuando se cumplen determinadas condiciones. Las condiciones contienen una o más reglas que están disponibles en una lista de reglas en CodePipeline. Si todas las reglas de una condición se realizan correctamente, se cumple la condición. Puede configurar las condiciones para que, cuando no se cumplan los criterios, se active el resultado especificado.

Cada condición tiene un conjunto de reglas, en el que las reglas se ordenan y se evalúan juntas. Por lo tanto, si una regla falla en la condición, la condición también fallará. Puede anular las condiciones de la regla en el tiempo de ejecución de la canalización.

Las condiciones se utilizan para tipos específicos de expresiones, y cada una tiene opciones específicas de resultados disponibles, de la siguiente manera:

- Entrada: las condiciones para realizar comprobaciones que, si se cumplen, permiten la entrada a una etapa. Las reglas se activan con las siguientes opciones de resultado: Fallar u Omitir.
- Fallo: las condiciones para realizar las comprobaciones de la etapa en caso de que se produzca un error. Las reglas se activan con la siguiente opción de resultado: Reversión.
- Éxito: las condiciones para realizar las comprobaciones de la etapa en caso de éxito. Las reglas se activan con las siguientes opciones de resultado: Reversión o Fallar.

Las condiciones están respaldadas por un conjunto de reglas para cada tipo de condición.

Para cada tipo de condición, la condición establece acciones específicas. La acción es el resultado de una comprobación de condición correcta o fallida. Por ejemplo, si la condición para la entrada (condición de entrada) genera una alarma (regla), la comprobación se realiza correctamente y el resultado (acción) es que se bloquea la entrada a la etapa.

También puede usar la AWS CodePipeline consola o la AWS CLI para deshacer manualmente o volver a intentar una etapa o las acciones de una etapa. Consulte Configuración de las condiciones de una etapa.

Temas


- Casos de uso de condiciones de etapa
- Consideraciones sobre la configuración de los resultados para las condiciones de etapa
- Consideraciones sobre las reglas configuradas para las condiciones de etapa
- Creación de condiciones de entrada
- Creación de condiciones de fallo
- Creación de condiciones de éxito
- Eliminación de condiciones de etapa
- Anulación de condiciones de etapa


# Casos de uso de condiciones de etapa

Las condiciones de etapa tienen varios casos de uso para configurar la seguridad de lanzamiento y de cambio en las canalizaciones. Los siguientes son ejemplos de casos de uso para las condiciones de etapa.

- Utilice una condición de entrada para definir una condición que compruebe el estado de la CloudWatch alarma y, a su vez, bloquee cualquier cambio si el entorno de producción no está en buen estado.
- Utilice una condición de entrada con un tiempo de espera de 60 para definir una condición que se evaluará cuando todas las acciones de una etapa se hayan completado correctamente y, a continuación, deshaga los cambios si una CloudWatch alarma pasa al estado de ALARMA en 60 minutos.
- Utilice la condición de éxito para definir una condición de modo que, cuando la etapa se complete correctamente, la regla compruebe si la hora actual se encuentra en la ventana de implementación y, a continuación, se implemente si la regla se ejecuta correctamente.

# Consideraciones sobre la configuración de los resultados para las condiciones de etapa

Los aspectos sobre las condiciones de etapa que se deben tener en cuenta son los siguientes:

- No puede utilizar el reintento automático de etapas con las condiciones onFailure.

- Al configurar una condición con un resultado de Reversión, la etapa solo puede revertirse a una ejecución anterior si está disponible en la versión actual de la estructura de canalización.
- Al configurar una condición con un resultado de Reversión, no se puede revertir a un identificador de ejecución de destino que sea un tipo de ejecución de reversión.
- En el caso de las condiciones de entrada que utilizan el resultado Omitir para saltarse la etapa si la condición falla, solo se admiten las reglas `LambdaInvoke` y `VariableCheck`.
- No se puede realizar un reintento manual de una etapa en una etapa que esté en estado Omitido.
- No se puede realizar una reversión manual a una etapa que esté en estado Omitido.
- No puede anular una condición si la condición está configurada con un resultado Omitir.
- A excepción de los resultados Omitir, puede anular la condición de una etapa al iniciar la ejecución de una canalización. En el caso de una condición de etapa en la que se produzca una anulación, la ejecución se realizará tal y como se detalla en la siguiente tabla.

| Tipo | Resultado configurado en caso de fallo en la condición | Estado de la etapa | Comportamiento de anulación |
|---|---|---|---|
| Entrada | Fail | In-progress | La etapa continúa. |
| Entrada | Omitir | Omitido | No se usa. |
| OnFailure | Reversión | Con error | La etapa ha fallado. |
| OnSuccess | Reversión | Realizado correctamente | La etapa continúa. |
| OnSuccess | Fail | Con error | La etapa continúa. |

## Consideraciones sobre las reglas configuradas para las condiciones de etapa

Los aspectos que se deben tener en cuenta sobre las reglas disponibles para las condiciones de etapa son los siguientes:

- Para la regla `LambdaInvoke`, primero debe configurar la función de Lambda que se utilizará en la regla. Tenga el ARN de la función de Lambda listo para proporcionarlo al configurar la regla.

- Para la `CloudWatchAlarm` regla, primero debe configurar el evento de CloudWatch eventos que se utilizará en la regla. Tenga preparado el ARN del evento que desee proporcionar al configurar la regla.

# Creación de condiciones de entrada

Puede configurar las condiciones de entrada de una etapa mediante la consola o la CLI. Configurará las reglas y los resultados correspondientes para cada condición. Para obtener un resultado de reversión, la canalización solo puede revertirse a una ejecución anterior si dicha ejecución anterior se inició en la versión actual de la estructura de canalización.

Los pasos proporcionan un ejemplo de condición de entrada que utiliza una regla de supervisión.

Para obtener más información, consulte Condición y RuleExecution en la Guía de CodePipeline API. RuleTypeId

## Creación de condiciones de entrada: ejemplo de CloudWatchAlarm regla (consola)

Puede configurar las condiciones de entrada para una etapa, junto con las reglas y los resultados que desee que realice la etapa cuando se cumplan las condiciones.

Configuración de una condición de entrada (consola)

1. Complete todos los requisitos previos, como crear el recurso y el ARN para una regla en la que se proporciona un recurso, como el. AWS CloudWatchAlarm
2. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en. http://console.aws.amazon.com/codesuite/codepipeline/home

   Se muestran los nombres y estados de todas las canalizaciones asociadas con su Cuenta de AWS .
3. En Nombre, elija el nombre de la canalización que desea editar.
4. En la página de detalles de la canalización, elija Edit.
5. En la página Editar, para la acción que desee editar, elija Editar etapa.
6. Seleccione Agregar condición de entrada. Aparecerá la tarjeta de Condición previa a la entrada a la fase con la opción de Error disponible para esta condición.
7. Elija Agregar regla y, a continuación, complete lo siguiente.

a. En Nombre de la regla, introduzca un nombre para la regla. En este ejemplo, escriba `MyAlarmRule`.

b. En Proveedor de reglas, elija el proveedor de reglas preconfigurado para agregarlo a la condición. Para este ejemplo AWS CloudWatchAlarm, elige y completa los pasos siguientes.

c. En Región, elija la región para su condición o deje la que aparece por defecto.

d. En Nombre de alarma, elija el CloudWatch recurso que desea usar para la regla. Debe haber creado previamente el recurso en su cuenta.

e. (Opcional) En Tiempo de espera, introduzca el tiempo de CodePipeline espera si la alarma está en estado de ALARMA cuando se evalúa por primera vez. Si la alarma está en estado Correcto cuando se comprueba la regla por primera vez, la regla se ejecutará correctamente de inmediato.

f. (Opcional) Introduzca los estados de alarma específicos que desee supervisar y el ARN del rol, si procede.

g. Cuando haya terminado de editar la etapa, elija Listo. En la página de edición de la canalización, elija Guardar.

8. Tras la ejecución, consulte el resultado.

## Creación de condiciones de entrada con el resultado Omitir y la regla **`VariableCheck`** (consola)

Puede configurar las condiciones de entrada para una etapa de modo que, si no se cumple la condición de entrada, se omita la etapa. Si la condición no se cumple, el resultado se activa y se omite la etapa. Cuando se omite una etapa, el estado de la etapa es Omitido y el estado de la acción es No se ejecutó. Para obtener información con respecto a las condiciones de etapa con los resultados Omitir, consulte Consideraciones sobre la configuración de los resultados para las condiciones de etapa.

En el siguiente ejemplo, la regla de verificación de variables determina que el valor no coincide, por lo que se omite la etapa de compilación.

Configure una condición de entrada con un Omitir resultado (consola)

1. Complete todos los requisitos previos, como crear el recurso y el ARN para una regla en la que se proporciona un recurso, como. AWS CloudWatchAlarm

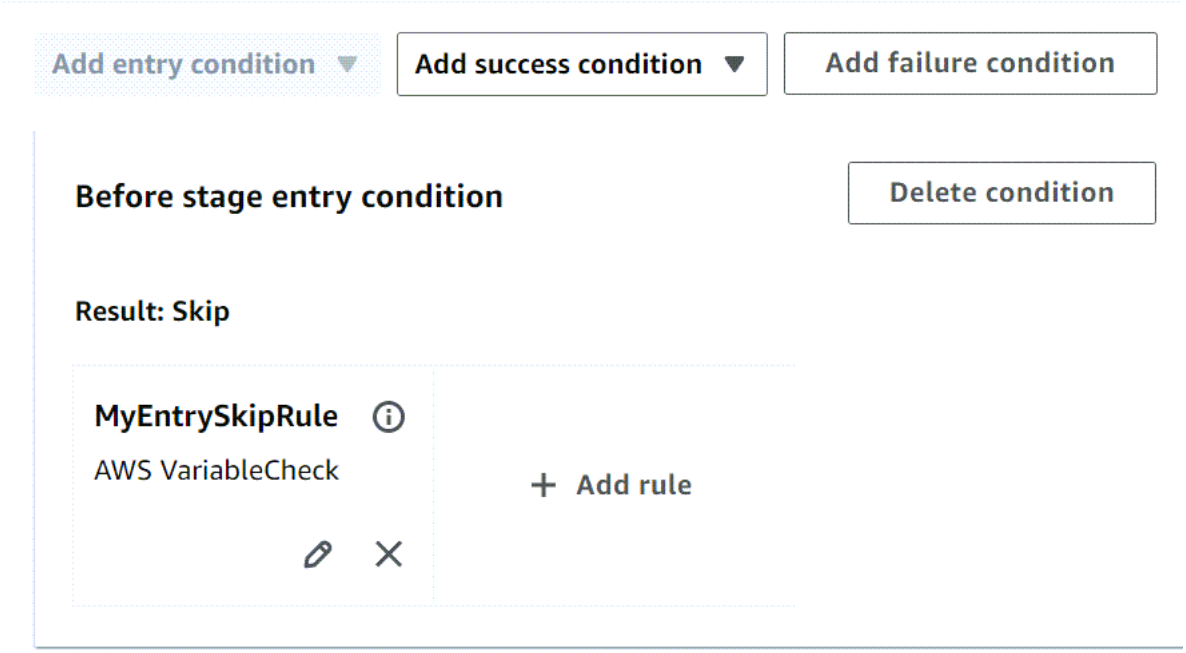
2. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en. http://console.aws.amazon.com/codesuite/codepipeline/home

   Se muestran los nombres y estados de todas las canalizaciones asociadas con su Cuenta de AWS .

3. En Nombre, elija el nombre de la canalización que desea editar.
4. En la página de detalles de la canalización, elija Edit.
5. En la página Editar, para la acción que desee editar, elija Editar etapa.
6. Seleccione Agregar condición de entrada y, a continuación, seleccione Omitir como el resultado.
7. Elija Agregar regla y, a continuación, complete lo siguiente.

   a. En Nombre de la regla, introduzca un nombre para la regla. En este ejemplo, escriba `MyAlarmRule`.

   b. En Proveedor de reglas, elija el proveedor de reglas preconfigurado para agregarlo a la condición. Para este ejemplo VariableCheck, elige y completa los pasos siguientes.

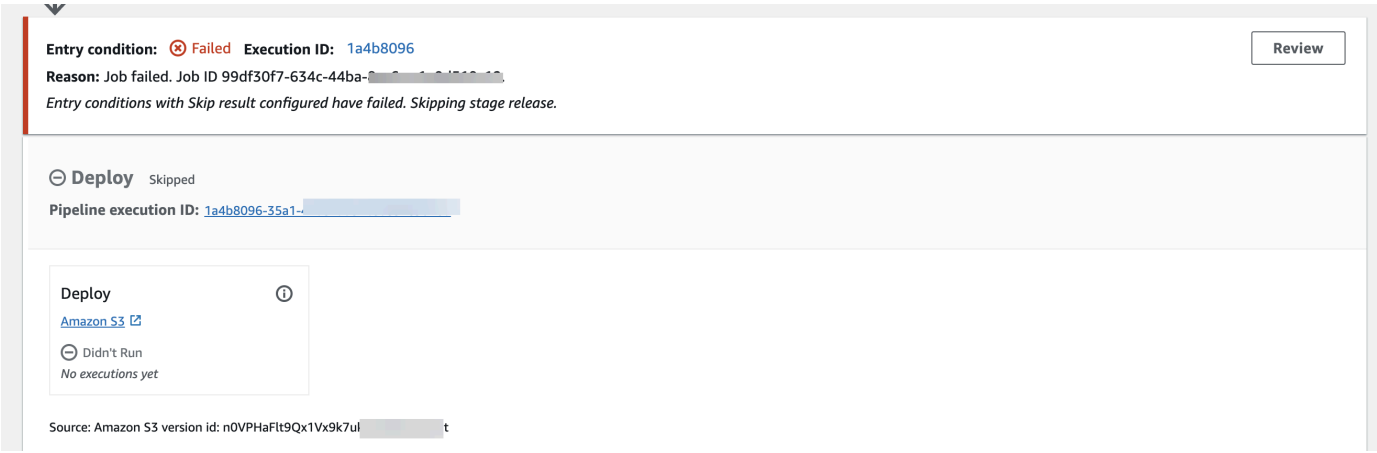



   c. En Región, elija la región para su condición o deje la que aparece por defecto.

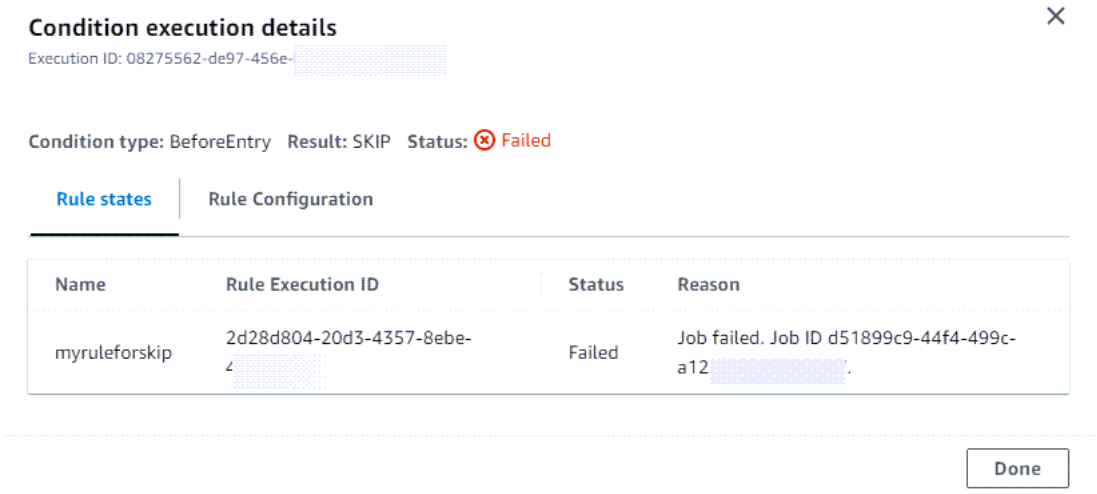
   d. En Variable, elige la variable con la que quieres comparar, `#{SourceVariables.FullRepositoryName}` por ejemplo, una canalización que tiene una acción de origen GitHub (a través de una GitHub aplicación). Introduzca el nombre del repositorio y elija el operador, como Igual a.

e. Cuando haya terminado de editar la etapa, elija Listo. En la página de edición de la canalización, elija Guardar.

8. Tras la ejecución, consulte el resultado.




9. Para revisar los detalles, seleccione Revisar. El detalle del siguiente ejemplo muestra que el resultado configurado para la condición es Omitir, que no se puede anular. El estado de la regla es Fallido porque no se cumple la condición.




## Creación de condiciones de entrada (CLI)

Para utilizar la condición de entrada AWS CLI para configurar una condición de entrada, utilice los comandos para crear o actualizar una canalización, tal y como se detalla en Creación de una canalización, etapas y acciones yEditar una canalización en CodePipeline.

Configuración de la condición y la regla o reglas (CLI)

- Abra un terminal (Linux, macOS o Unix) o el símbolo del sistema (Windows) y utilice la AWS CLI para ejecutar el comando `update-pipeline`, especificando la condición de fallo en la estructura de canalización. El siguiente ejemplo configura una condición de entrada para una etapa denominada`Deploy`:

```
{
    "name": "Deploy",
    "actions": [
        {
            "name": "Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "S3",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "BucketName": "MyBucket",
                "Extract": "false",
                "ObjectKey": "object.xml"
            },
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1",
            "namespace": "DeployVariables"
        }
    ],
    "beforeEntry": {
        "conditions": [
            {
                "result": "FAIL",
                "rules": [
                    {
                        "name": "MyAlarmRule",
                        "ruleTypeId": {
                            "category": "Rule",
```

```
                        "owner": "AWS",
                        "provider": "CloudWatchAlarm",
                        "version": "1"
                    },
                    "configuration": {
                        "AlarmName": "CWAlarm",
                        "WaitTime": "1"
                    },
                    "inputArtifacts": [],
                    "region": "us-east-1"
                }
            ]
        }
    ]
  }
}
```

Para obtener más información sobre cómo configurar las condiciones correctas para la reversión de etapas, consulte la referencia SuccessConditions de la CodePipeline API.

## Creación de condiciones de entrada (CFN)

CloudFormation Para configurar una condición de entrada, utilice el `beforeEntry` parámetro. En la entrada, la etapa ejecutará la regla y realizará el resultado.

```
beforeEntry:
     Result: FAIL
```

- Actualice la plantilla como se muestra en el fragmento de código siguiente. En el siguiente ejemplo se configura una condición de entrada con una regla denominada `MyMonitorRule`:

```
Name: Deploy
Actions:
- Name: Deploy
  ActionTypeId:
    Category: Deploy
    Owner: AWS
    Provider: S3
    Version: '1'
  RunOrder: 1
  Configuration:
```

```
    BucketName: MyBucket
    Extract: 'false'
    ObjectKey: object.xml
  OutputArtifacts: []
  InputArtifacts:
  - Name: SourceArtifact
  Region: us-east-1
  Namespace: DeployVariables
BeforeEntry:
  Conditions:
  - Result: FAIL
    Rules:
    - Name: MyMonitorRule
      RuleTypeId:
        Category: Rule
        Owner: AWS
        Provider: CloudWatchAlarm
        Version: '1'
      Configuration:
        AlarmName: CWAlarm
        WaitTime: '1'
      InputArtifacts: []
      Region: us-east-1
```

Para obtener más información sobre la configuración de las condiciones de BeforeEntry, consulte AWS::CodePipeline: :Pipeline BeforeEntryConditions `StageDeclaration` en la Guía del CloudFormation usuario.

# Creación de condiciones de fallo

Puede configurar las condiciones de fallo para una etapa mediante la consola o la CLI. Configurará las reglas y los resultados correspondientes para cada condición. Para obtener un resultado de reversión, la canalización solo puede revertirse a una ejecución anterior si dicha ejecución anterior se inició en la versión actual de la estructura de canalización.

## Creación de condiciones de fallo (consola)

Puede configurar las condiciones de fallo para una etapa, junto con las reglas y los resultados que desee que realice la etapa cuando se cumplan las condiciones.

Configuración de una condición de fallo (consola)

1. Complete todos los requisitos previos, como crear el recurso y el ARN para una regla en la que se proporcione un recurso, como la regla.  LambdaInvoke
2. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en. http://console.aws.amazon.com/codesuite/codepipeline/home

   Se muestran los nombres y estados de todas las canalizaciones asociadas con su Cuenta de AWS .
3. En Nombre, elija el nombre de la canalización que desea editar.
4. En la página de detalles de la canalización, elija Edit.
5. En la página Editar, para la acción que desee editar, elija Editar etapa.
6. Seleccione Agregar condición de fallo. Aparecerá la tarjeta de Condición de fallo con la opción de Reversión disponible para esta condición.
7. Elija Agregar regla y, a continuación, complete lo siguiente.

   a. En Nombre de la regla, introduzca un nombre para la regla. En este ejemplo, escriba `MyLambdaRule`.

   b. En Proveedor de reglas, elija el proveedor de reglas preconfigurado para agregarlo a la condición. Para este ejemplo AWS LambdaInvoke, elige y completa los pasos siguientes.

   c. En Región, elija la región para su condición o deje la que aparece por defecto.

   d. En Artefactos de entrada, seleccione el artefacto de origen.

   e. En Nombre de la función, elija el recurso de Lambda que se va a usar para la regla. Debe haber creado previamente el recurso en su cuenta.

   f. (Opcional) En Parámetros del usuario, introduzca cualquier par que represente parámetros para una configuración adicional.

   g. (Opcional) En el ARN del rol, introduzca el ARN del rol si está configurado.

   h. (Opcional) En Tiempo de espera en minutos, introduzca el tiempo en minutos que la regla debe esperar antes de que se agote el tiempo de espera.

   i. Cuando haya terminado de editar la etapa, elija Listo. En la página de edición de la canalización, elija Guardar.

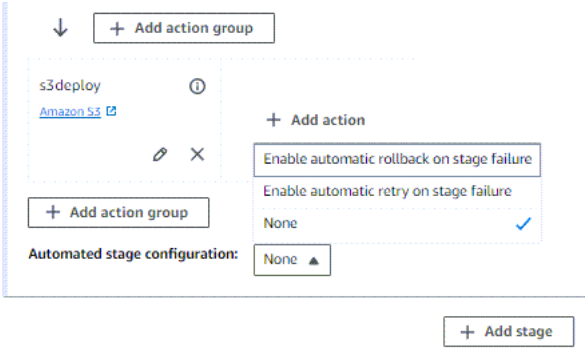
# Creación de condiciones onFailure con ejemplo de resultado de reintento (consola)

Puede configurar las condiciones onFailure para una etapa de modo que, si no se cumple la condición de entrada, se reintente la etapa. Como parte de este resultado, se configura el modo de reintento, especificando si se deben reintentar las acciones fallidas o si se debe reintentar la etapa fallida.

Configuración de una condición onFailure con un resultado de reintento (consola)

1. Complete todos los requisitos previos, como crear el recurso y el ARN para una regla en la que se proporciona un recurso, como el. AWS CloudWatchAlarm
2. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en. http://console.aws.amazon.com/codesuite/codepipeline/home

   Se muestran los nombres y estados de todas las canalizaciones asociadas con su Cuenta de AWS .
3. En Nombre, elija el nombre de la canalización que desea editar.
4. En la página de detalles de la canalización, elija Edit.
5. En la página Editar, para la acción que desee editar, elija Editar etapa.
6. En la parte inferior de la etapa, en Configuración automatizada de escenarios, seleccione Habilitar el reintento automático en caso de fallo de una etapa. En el Modo de reintento, seleccione Reintentar etapa fallida o Reintentar acciones fallidas.




7. Elija agregar una condición onFailure y, a continuación, elija Agregar regla e introduzca una regla para la condición.

   a. En Nombre de la regla, introduzca un nombre para la regla. En este ejemplo, escriba `MyAlarmRule`.

b. En Proveedor de reglas, elija el proveedor de reglas preconfigurado para agregarlo a la condición. Para este ejemplo  CloudWatchAlarm, elija y complete los pasos siguientes.

c. En Región, elija la región para su condición o deje la que aparece por defecto.

d. En Nombre de alarma, elija el recurso configurado para la alerta.

e. Cuando haya terminado de editar la etapa, elija Listo. En la página de edición de la canalización, elija Guardar.

8. Tras la ejecución, consulte el resultado.

## Creación de condiciones de fallo (CLI)

AWS CLI Para configurar una condición de fallo, utilice los comandos para crear o actualizar una canalización, tal y como se detalla en Creación de una canalización, etapas y acciones yEditar una canalización en CodePipeline.

Configuración de la condición y la regla o reglas (CLI)

- Abra un terminal (Linux, macOS o Unix) o el símbolo del sistema (Windows) y utilice la AWS CLI para ejecutar el comando `update-pipeline`, especificando la condición de fallo en la estructura de canalización. En el siguiente ejemplo, se configura una condición de fallo para una etapa denominada`Deploy`:

```
{
    "name": "Deploy",
    "actions": [
        {
            "name": "Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "S3",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "BucketName": "MyBucket",
                "Extract": "false",
                "ObjectKey": "object.xml"
            },
            "outputArtifacts": [],
```

```
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1",
            "namespace": "DeployVariables"
        }
    ],
    "onFailure": {
        "conditions": [
            {
                "result": "ROLLBACK",
                "rules": [
                    {
                        "name": "MyLambdaRule",
                        "ruleTypeId": {
                            "category": "Rule",
                            "owner": "AWS",
                            "provider": "LambdaInvoke",
                            "version": "1"
                        },
                        "configuration": {
                            "FunctionName": "my-function"
                        },
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "region": "us-east-1"
                    }
                ]
            }
        ]
    }
}
```

Para obtener más información sobre la configuración de las condiciones de error, consulte FailureConditions la referencia de la CodePipeline API.

# Creación de condiciones de fallo (CFN)

CloudFormation Para configurar una condición de fallo, utilice el `OnFailure` parámetro. En caso de error, el escenario ejecutará la regla y ejecutará el resultado.

```
OnFailure:
     Result: ROLLBACK
```

- Actualice la plantilla como se muestra en el fragmento de código siguiente. El siguiente ejemplo configura una OnFailure condición con una regla denominada`MyMonitorRule`:

```
name: Deploy
actions:
- name: Deploy
  actionTypeId:
    category: Deploy
    owner: AWS
    provider: S3
    version: '1'
  runOrder: 1
  configuration:
    BucketName: MyBucket
    Extract: 'false'
    ObjectKey: object.xml
  outputArtifacts: []
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
  namespace: DeployVariables
OnFailure:
  conditions:
  - result: ROLLBACK
    rules:
    - name: MyMonitorRule
      ruleTypeId:
        category: Rule
        owner: AWS
        provider: CloudWatchAlarm
        version: '1'
      configuration:
        AlarmName: AlarmOnHelloWorldInvocation
        AlarmStates: ALARM
```

```
        WaitTime: '1'
      inputArtifacts: []
      region: us-east-1
```

Para obtener más información sobre la configuración de las condiciones de error, consulte OnFailure `StageDeclaration` la Guía del CloudFormation usuario.

# Creación de condiciones de éxito

Puede configurar las condiciones de éxito para una etapa mediante la consola o la CLI. Configurará las reglas y los resultados correspondientes para cada condición. Para obtener un resultado de reversión, la canalización solo puede revertirse a una ejecución anterior si dicha ejecución anterior se inició en la versión actual de la estructura de canalización.

Los pasos proporcionan un ejemplo de condición de éxito que utiliza una regla de ventana de implementación.

Para obtener más información, consulte Condición y RuleExecution en la Guía de la CodePipeline API. RuleTypeId

## Creación de condiciones de éxito (consola)

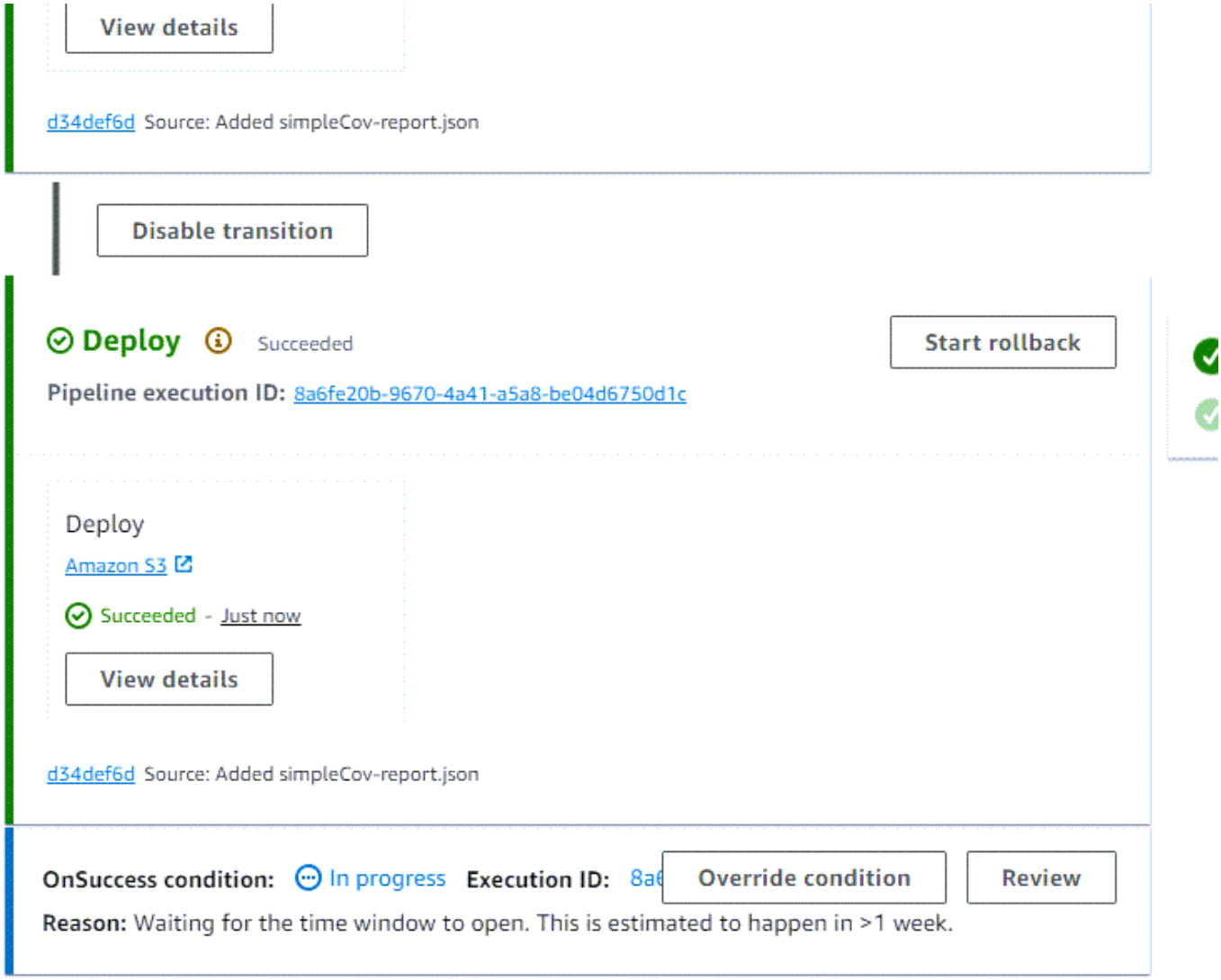
Puede configurar las condiciones de éxito para una etapa, junto con las reglas y los resultados que desee que realice la etapa cuando se cumplan las condiciones.

Configuración de una condición de éxito (consola)

1. Complete todos los requisitos previos, como crear el recurso y el ARN para una regla en la que se proporciona un recurso, como el. AWS LambdaRule
2. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en. http://console.aws.amazon.com/codesuite/codepipeline/home

   Se muestran los nombres y estados de todas las canalizaciones asociadas con su Cuenta de AWS .
3. En Nombre, elija el nombre de la canalización que desea editar.
4. En la página de detalles de la canalización, elija Edit.
5. En la página Editar, para la acción que desee editar, elija Editar etapa.

6. Seleccione Agregar condición de éxito. Aparecerá la tarjeta Condición de éxito en la fase. Seleccione la opción Reversión o Fallar que se muestra entre los resultados disponibles para este tipo de condición.
7. Elija Agregar regla y, a continuación, complete lo siguiente.
   a. En Nombre de la regla, introduzca un nombre para la regla. En este ejemplo, escriba `MyDeploymentRule`.
   b. En Proveedor de reglas, elija la regla preconfigurada para agregarla a la condición. Para este ejemplo AWS DeploymentWindow, elige y completa los pasos siguientes.
   c. En Región, elija la región para su condición o deje la que aparece por defecto.
   d. En Cron, introduzca la expresión cron para la ventana de implementación. La expresión cron define los días y las horas en los que se debe permitir la implementación. Para obtener información de referencia sobre las expresiones cron, consulte Uso de expresiones cron y de frecuencia para programar reglas.
   e. (Opcional) En TimeZone, introduzca la zona horaria de la ventana de implementación.
8. Tras la ejecución, consulte el resultado.




## Creación de condiciones de éxito (CLI)

AWS CLI Para configurar una condición de éxito, utilice los comandos para crear o actualizar una canalización, tal y como se detalla en Creación de una canalización, etapas y acciones yEditar una canalización en CodePipeline.

Configuración de la condición y la regla o reglas (CLI)

- Abra un terminal (Linux, macOS o Unix) o el símbolo del sistema (Windows) y utilice la AWS CLI para ejecutar el comando `update-pipeline`, especificando la condición de fallo en la

estructura de canalización. En el siguiente ejemplo, se configura una condición On Success para una etapa denominada`Deploy`, donde la regla se denomina`MyDeploymentRule`:

```
{
    "name": "Deploy",
    "actions": [
        {
            "name": "Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "S3",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "BucketName": "MyBucket",
                "Extract": "false",
                "ObjectKey": "object.xml"
            },
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1",
            "namespace": "DeployVariables"
        }
    ],
    "onSuccess": {
       "conditions": [
            {
                "result": "FAIL",
                "rules": [
                    {
                        "name": "MyAlarmRule",
                        "ruleTypeId": {
                            "category": "Rule",
                            "owner": "AWS",
                            "provider": "CloudWatchAlarm",
                            "version": "1"
                        },
```

```
                        "configuration": {
                            "AlarmName": "CWAlarm",
                            "WaitTime": "1"
                        },
                        "inputArtifacts": [],
                        "region": "us-east-1"
                    }
                ]
            }
        ]
    }
}
```

Para obtener más información sobre la configuración de las condiciones de éxito, consulte SuccessConditions la referencia de la CodePipeline API.

## Creación de una condición de éxito (CFN)

CloudFormation Para configurar una condición en caso de éxito, utilice el `OnSuccess` parámetro. En caso de éxito, la etapa ejecutará la regla y realizará el resultado.

```
OnSuccess:
     Result: ROLLBACK
```

- Actualice la plantilla como se muestra en el fragmento de código siguiente. El siguiente ejemplo configura una OnSuccess condición con una regla denominada`MyDeploymentWindowRule`:

```
name: Deploy
actions:
- name: Deploy
  actionTypeId:
    category: Deploy
    owner: AWS
    provider: S3
    version: '1'
  runOrder: 1
  configuration:
    BucketName: MyBucket
    Extract: 'false'
    ObjectKey: object.xml
  outputArtifacts: []
```

```
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
  namespace: DeployVariables
onSuccess:
   conditions:
  - result: FAIL
    rules:
    - name: MyMonitorRule
      ruleTypeId:
        category: Rule
        owner: AWS
        provider: CloudWatchAlarm
        version: '1'
      configuration:
        AlarmName: CWAlarm
        WaitTime: '1'
      inputArtifacts: []
      region: us-east-1
```

Para obtener más información sobre la configuración de las condiciones de error para la reversión por etapas, consulte la OnFailure sección siguiente `StageDeclaration` en la Guía del CloudFormation usuario.

## Eliminación de condiciones de etapa

Puede eliminar las condiciones de etapa que se hayan configurado para la canalización.

Para eliminar una condición de etapa

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

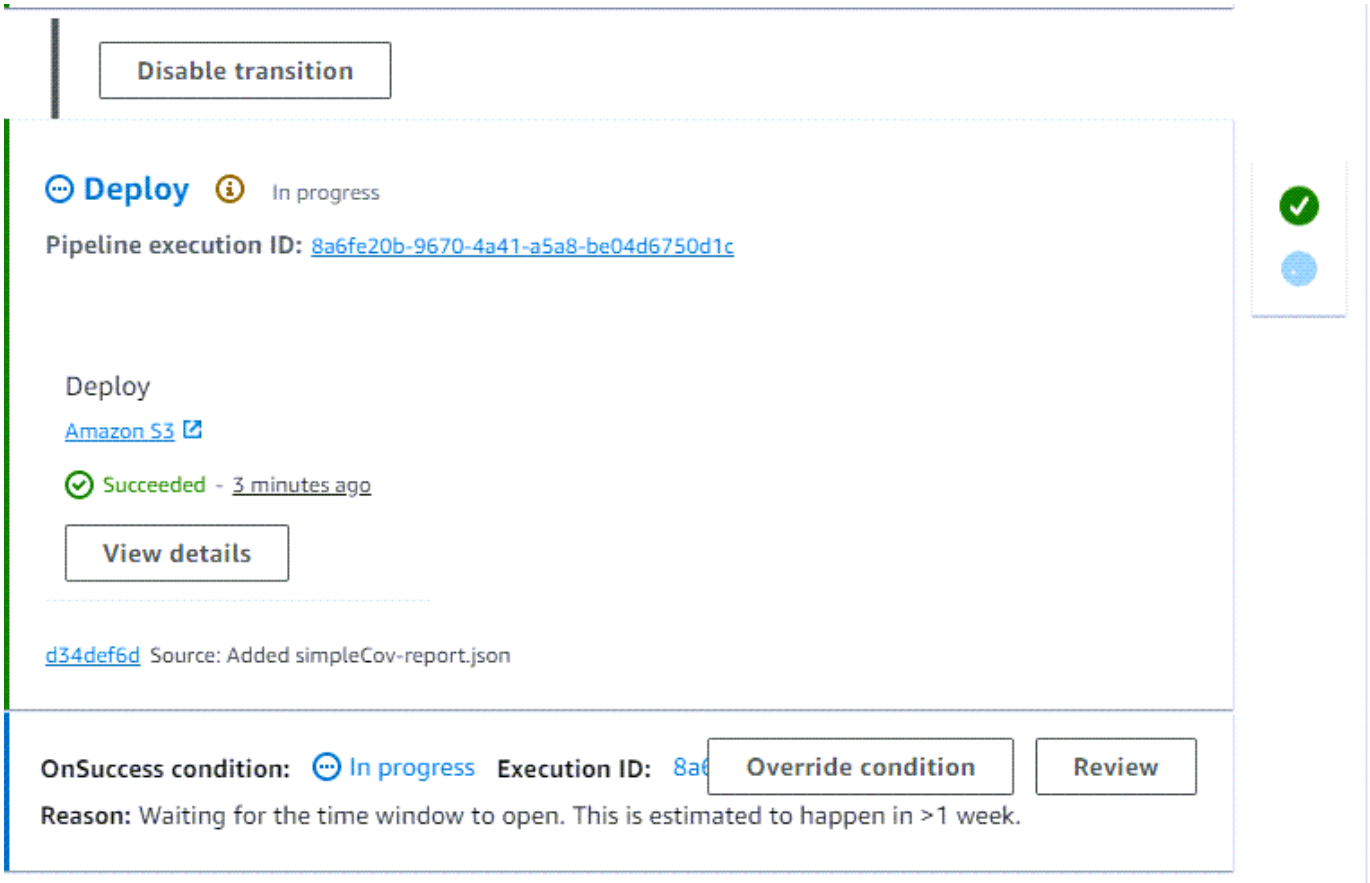
   Cuenta de AWS Se muestran los nombres y el estado de todas las canalizaciones asociadas a la suya.
2. En Nombre, elija el nombre de la canalización que desea editar.
3. En la página de detalles de la canalización, elija Edit.
4. En la página Editar, para la condición que desee editar, elija Editar etapa.
5. Junto a la condición que desee eliminar, elija Eliminar condición.

# Anulación de condiciones de etapa

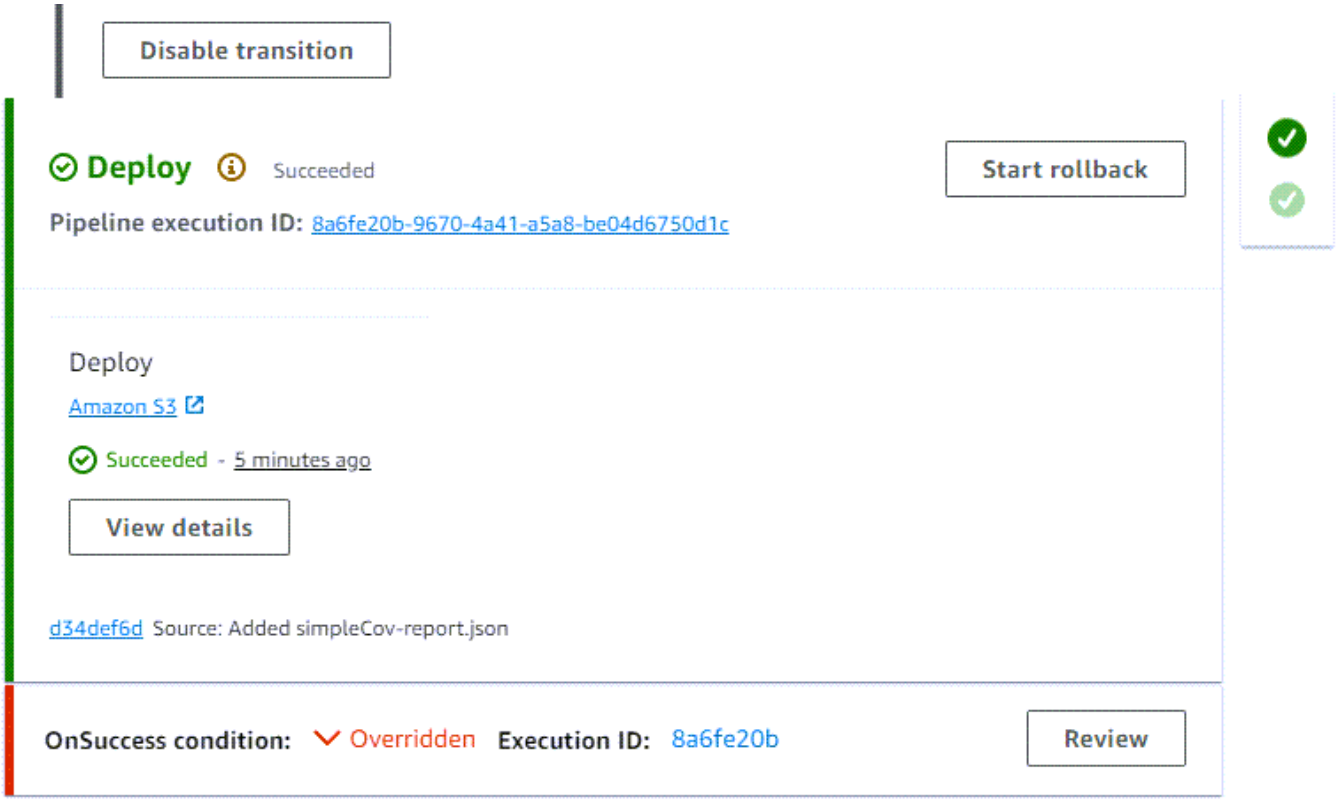
Puede anular las condiciones de etapa que se hayan configurado para la canalización. En la consola, cuando la etapa y la regla estén en ejecución, puede optar por anular la condición de la etapa. Esto da como resultado la puesta en marcha de la etapa.

Para anular una condición de etapa

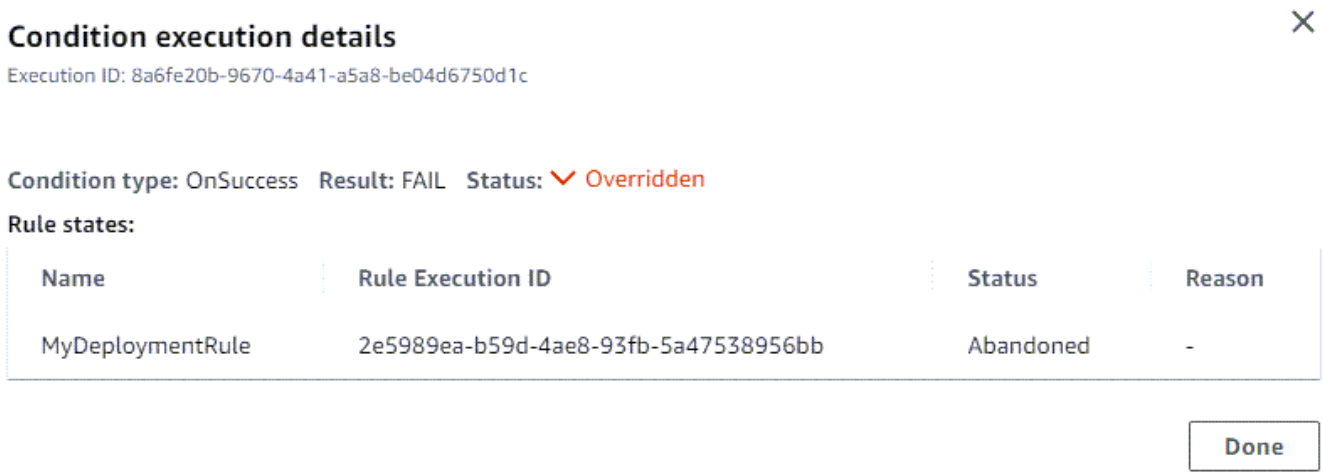
1. En este ejemplo, la etapa de canalización se ejecuta con una condición. El botón Anular está habilitado.




2. Junto a la condición que desee anular, seleccione Anular.




3. Para revisar los detalles, seleccione Revisar. El detalle del siguiente ejemplo muestra que el resultado configurado para la condición es Fallar, que no se ha anulado. El estado de la regla es Abandonado debido a la anulación.



# Uso de tipos de acciones, acciones personalizadas y acciones de aprobación

En AWS CodePipeline, una acción es parte de la secuencia en una etapa de una canalización. Es una tarea que se realiza en el artefacto en esa etapa. Las acciones de canalización se producen en un orden especificado, en serie o en paralelo, según se determina en la configuración de la etapa.

CodePipeline proporciona soporte para seis tipos de acciones:

- origen
- Build
- Prueba
- Implementación
- Aprobación
- Invocación

Para obtener información sobre los productos Servicio de AWS y servicios de sus socios que puedes integrar en tu cartera en función del tipo de acción, consultaIntegraciones con tipos de CodePipeline acciones.

Temas


- Trabajar con tipos de acciones
- Crea y añade una acción personalizada en CodePipeline
- Etiquetar una acción personalizada en CodePipeline
- Invoca un AWS Lambda función en una canalización en CodePipeline
- Incorporación de una acción de aprobación manual a una etapa
- Agregue una acción interregional en CodePipeline
- Trabajar con variables


## Trabajar con tipos de acciones

Los tipos de acciones son acciones preconfiguradas que usted, como proveedor, crea para los clientes mediante uno de los modelos de integración compatibles en AWS CodePipeline.

Puede solicitar, ver y actualizar los tipos de acciones. Si el tipo de acción se crea para tu cuenta como propietario, puedes usarlo AWS CLI para ver o actualizar las propiedades y la estructura del tipo de acción. Si eres el proveedor o el propietario del tipo de acción, tus clientes pueden elegir la acción y añadirla a sus canalizaciones cuando esté disponible en CodePipeline.

> **Note**
>
> Las acciones se crean con `custom` en el campo `owner` para ejecutarlas con un proceso de trabajo. No se crean con un modelo de integración. Para obtener información acerca de las acciones personalizadas, consulte Crea y añade una acción personalizada en CodePipeline.

### Componentes de tipo acción

Los siguientes componentes componen un tipo de acción.

- ID de tipo de acción: el ID consta de la categoría, el propietario, el proveedor y la versión. En el siguiente ejemplo se muestra un ID de tipo de acción con el propietario de `ThirdParty`, una categoría de `Test`, el nombre de proveedor `TestProvider` y una versión de `1`.

```
{
    "Category": "Test",
    "Owner": "ThirdParty",
    "Provider": "TestProvider",
    "Version": "1"
},
```

- Configuración del ejecutor: el modelo de integración, o motor de acción, especificado al crear la acción. Al especificar el ejecutor de un tipo de acción, se elige uno de los dos tipos siguientes:
  - Lambda: el propietario del tipo de acción escribe la integración como una función de Lambda, que se invoca CodePipeline siempre que hay un trabajo disponible para la acción.
  - JobWorker: El propietario del tipo de acción escribe la integración como un trabajador que sondea los trabajos disponibles en las canalizaciones de clientes. A continuación, el trabajador ejecuta el trabajo y envía el resultado del trabajo a través CodePipeline de las CodePipeline API.

> **Note**
>
> El modelo de integración de los procesos de trabajo no es el modelo de integración preferido.

- Artefactos de entrada y salida: límites de los artefactos que el propietario del tipo de acción designa para los clientes de la acción.
- Permisos: la estrategia de permisos que designa a los clientes que pueden acceder al tipo de acción de terceros. Las estrategias de permisos disponibles dependen del modelo de integración elegido para el tipo de acción.
- URL: enlaces profundos a recursos con los que el cliente puede interactuar, como la página de configuración del propietario del tipo de acción.

Temas


- Solicitar un tipo de acción
- Añadir un tipo de acción disponible a una canalización (consola)
- Ver un tipo de acción
- Actualizar un tipo de acción


## Solicitar un tipo de acción

Cuando un proveedor externo solicita un nuevo tipo de CodePipeline acción, el tipo de acción se crea para el propietario del tipo de acción CodePipeline, y este puede administrar y ver el tipo de acción.

Un tipo de acción puede ser público o privado. Cuando se crea el tipo de acción, es privado. Para solicitar que un tipo de acción se cambie por una acción pública, ponte en contacto con el equipo CodePipeline de servicio.

Antes de crear el archivo de definición de la acción, los recursos de ejecución y la solicitud de tipo de acción para el CodePipeline equipo, debes elegir un modelo de integración.

## Paso 1: Elegir modelo de integración

Elija su modelo de integración y, a continuación, cree la configuración para ese modelo. Tras elegir el modelo de integración, debe configurar los recursos de integración.

- Para el modelo de integración de Lambda, cree una función de Lambda y agregue permisos. Agregue permisos a la función Lambda de su integrador para proporcionar al CodePipeline servicio permisos para invocarlo mediante el principal del CodePipeline servicio:. `codepipeline.amazonaws.com` Los permisos se pueden agregar mediante CloudFormation o la línea de comandos.
  - Ejemplo de adición de permisos mediante 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
```

  - Documentación para la línea de comandos
- Para el modelo de integración de trabajadores, se crea una integración con una lista de cuentas permitidas en la que el trabajador busca trabajos con las CodePipeline API.

## Paso 2: Crear un archivo de definición de tipo de acción

El tipo de acción se define en un archivo de definición de tipo de acción mediante JSON. En el archivo, debe incluir la categoría de acción, el modelo de integración utilizado para administrar el tipo de acción y las propiedades de configuración.

> **Note**
>
> Después de crear una acción pública, no puede cambiar la propiedad del tipo de acción en `properties` de `optional` a `required`. No se puede cambiar el `owner`.

Para obtener más información sobre los parámetros del archivo de definición del tipo de acción, consulte ActionTypeDeclaration y UpdateActionType en la referencia de la CodePipeline API.

El archivo de definición de tipos de acción consta de ocho secciones:

- `description`: la descripción del tipo de acción que se va a actualizar.
- `executor`: información sobre el ejecutor de un tipo de acción que se creó con un modelo de integración compatible, ya sea `Lambda` o `job worker`. Solo puede proporcionar `jobWorkerExecutorConfiguration` o `lambdaExecutorConfiguration`, según el tipo de ejecutor.
  - `configuration`: recursos para la configuración del tipo de acción, en función del modelo de integración elegido. Para el modelo de integración de Lambda, utilice el ARN de función de Lambda. Para el modelo de integración de procesos de trabajo, utilice la cuenta o la lista de cuentas desde donde se ejecuta el proceso de trabajo.
  - `jobTimeout`: el tiempo de espera en segundos para el trabajo. La ejecución de una acción puede constar de varios trabajos. Este es el tiempo de espera de una sola tarea y no de toda la ejecución de la acción.

    > **Note**
    >
    > Para el modelo de integración de Lambda, el tiempo de espera máximo es de 15 minutos.

  - `policyStatementsTemplate`: la declaración de política que especifica los permisos de la cuenta del CodePipeline cliente que se necesitan para ejecutar correctamente una acción.
  - `type`: el modelo de integración utilizado para crear y actualizar el tipo de acción, ya sea `Lambda` o `JobWorker`.
- `id`: la categoría, el propietario, el proveedor y el ID de versión del tipo de acción:
  - `category`: el tipo de acción puede realizarse en la etapa: fuente, compilación, implementación, prueba, invocación o aprobación.
  - `provider`: el proveedor del tipo de acción al que se llama, como la empresa proveedora o el nombre del producto. El nombre del proveedor se proporciona al crear el tipo de acción.
  - `owner`: el creador del tipo de acción que se llama: `AWS` o `ThirdParty`.
  - `version`: una cadena utilizada para versionar el tipo de acción. Para la primera versión, establezca el número de versión en 1.
- `inputArtifactDetails`: el número de artefactos que cabe esperar de la fase anterior de la canalización.

- `outputArtifactDetails`: el número de artefactos que cabe esperar del resultado de la fase de tipo acción.
- `permissions`: detalles que identifican las cuentas con permisos para usar el tipo de acción.
- `properties`: los parámetros necesarios para completar las tareas del proyecto.
  - `description`: la descripción de la propiedad de configuración de la acción que se muestra a los usuarios.
  - `optional`: si la propiedad de configuración es opcional.
  - `noEcho`: si el valor del campo introducido por el cliente se omite del registro. Si es `true` así, el valor se borra cuando se devuelve con una solicitud de GetPipeline API.
  - `key`: si la propiedad de configuración es una clave.
  - `queryable`: si la propiedad se usa con el sondeo. Un tipo de acción puede tener hasta una propiedad consultable. Si tiene una, dicha propiedad debe ser obligatoria y no ser secreta.
  - `name`: el nombre de la propiedad que se muestra a los usuarios.
- `urls`: Los usuarios CodePipeline muestran una lista de las URL.
  - `entityUrlTemplate`: URL de los recursos externos del tipo de acción, como una página de configuración.
  - `executionUrlTemplate`: URL con los detalles de la última ejecución de la acción.
  - `revisionUrlTemplate`: La URL que se muestra en la CodePipeline consola lleva a la página en la que los clientes pueden actualizar o cambiar la configuración de la acción externa.
  - `thirdPartyConfigurationUrl`: la dirección URL de una página en la que los usuarios pueden registrarse en un servicio externo y realizar la configuración inicial de la acción proporcionada por ese servicio.

En el código siguiente se muestra un ejemplo de archivo de definición del tipo de acción.

```
{
   "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"
      }
   }
}
```

## Paso 3: Registra tu integración con CodePipeline

Para registrar tu tipo de acción con CodePipeline, ponte en contacto con el equipo CodePipeline de servicio y envíales tu solicitud.

El equipo CodePipeline de servicio registra la nueva integración de tipos de acción realizando cambios en la base de código del servicio. CodePipeline registra dos nuevas acciones: una acción pública y una acción privada. Utiliza la acción privada para realizar las pruebas y, cuando esté lista, activa la acción pública para atender el tráfico de clientes.

Para registrar una solicitud de una integración de Lambda

- Envía una solicitud al equipo CodePipeline de servicio mediante el siguiente formulario.

```
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 Regiones de AWS where your action will be available

5. Will this be available as a public action?
```

Para registrar una solicitud de una integración de proceso de trabajo

- Envíe una solicitud al equipo CodePipeline de servicio mediante el siguiente formulario.

```
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 Regiones de AWS where your action will be available

5. Will this be available as a public action?
```

### Paso 4: Activar la nueva integración

Ponte en contacto con el equipo de CodePipeline servicio cuando estés listo para usar la nueva integración de forma pública.

## Añadir un tipo de acción disponible a una canalización (consola)

Añada su tipo de acción a una canalización para poder probarla. Puede hacerlo creando una nueva canalización o editando una existente.

> **Note**
>
> Si su tipo de acción es una acción de categoría de origen, compilación o implementación, puede añadirla creando una canalización. Si el tipo de acción figura en la categoría de pruebas, debe añadirla modificando una canalización existente.

Para añadir tu tipo de acción a una canalización existente desde la CodePipeline consola

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En la lista de canalizaciones, elija la canalización a la que quieres añadir el tipo de acción.
3. En la página de vista de resumen de la canalización, elija Editar.
4. Elija editar la etapa. En la etapa en la que desea añadir la acción de aprobación, elija + Añadir grupo de acciones. Aparece la página Editar acción.
5. En la página Editar acción, en Nombre de la acción, introduzca un nombre para la acción. Este es el nombre que se muestra para la etapa de su proceso.
6. En el Proveedor de acciones, elige el tipo de acción de la lista.

   Tenga en cuenta que el valor de la lista se basa en el `provider` especificado en el archivo de definición del tipo de acción.
7. En Artefactos de entrada, introduzca el nombre del artefacto en este formato:

   *`Artifactname`*`::`*`FileName`*

   Tenga en cuenta que las cantidades mínimas y máximas permitidas se definen en función de los `inputArtifactDetails` especificados en el archivo de definición del tipo de acción.
8. Selecciona  Conectar a < Action_Name >.

   Se abre una ventana del navegador que conecta con el sitio web que ha creado para su tipo de acción.
9. Inicie sesión en su sitio web como cliente y complete los pasos que sigue un cliente para usar su tipo de acción. Los pasos varían en función de la categoría de la acción, el sitio web y la configuración, pero normalmente incluyen una acción de finalización que devuelve al cliente a la página de  edición de la acción.

10. En la página CodePipeline Editar acción, aparecen los campos de configuración adicionales para la acción. Los campos que se muestran son las propiedades de configuración que especificó en el archivo de definición de acciones. Introduzca la información en los campos personalizados para su tipo de acción.

    Por ejemplo, si el archivo de definición de acciones especificó un nombre de propiedad `Host`, y un campo con la etiqueta Host con la etiqueta para la acción en la página Editar acción.

11. En Artefactos de salida, introduzca el nombre del artefacto en este formato:

    *`Artifactname`*`::`*`FileName`*

    Tenga en cuenta que las cantidades mínimas y máximas permitidas se definen en función de los `outputArtifactDetails` especificados en el archivo de definición del tipo de acción.

12. Elija Listo para volver a la página de detalles de la canalización.

    > **Note**
    >
    > Si lo desea, sus clientes pueden utilizar la CLI para añadir el tipo de acción a su canalización.

13. Para probar su acción, confirme un cambio en el origen especificado en la etapa de origen de la canalización o siga los pasos indicados en Iniciar manualmente una canalización.

Para crear una canalización con su tipo de acción, siga los pasos en Creación de una canalización, etapas y acciones y elija su tipo de acción entre tantas etapas como vaya a probar.

## Ver un tipo de acción

Puede usar la CLI para ver el tipo de acción. Utilice el comando get-action-type para ver los tipos de acciones que se han creado mediante un modelo de integración.

Para ver un tipo de acción

1. Cree un archivo JSON de entrada con el nombre `file.json`. Agregue su ID de tipo de acción en formato JSON, tal como se muestra en el siguiente ejemplo.

```
{
    "category": "Test",
    "owner": "ThirdParty",
```

```
    "provider": "TestProvider",
    "version": "1"
}
```

2. En una ventana de terminal o en la línea de comandos, ejecute el comando get-action-type.

```
aws codepipeline get-action-type --cli-input-json file://file.json
```

Este comando devuelve el resultado de la definición de acción para un tipo de acción. En este ejemplo, se muestra un tipo de acción que se creó con el modelo de integración de Lambda.

```
{
    "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": []
    }
```

```
}
```

## Actualizar un tipo de acción

Puede usar la CLI para editar los tipos de acciones que se crean con un modelo de integración.

En el caso de un tipo de acción pública, no puede actualizar el propietario, no puede cambiar las propiedades opcionales por obligatorias y solo puede añadir nuevas propiedades opcionales.

1. Use el comando `get-action-type` para obtener la estructura del tipo de acción. Copie la estructura.
2. Cree un archivo de entrada JSON con el nombre `action.json`. Pegue en ella la estructura del tipo de acción que copió en el paso anterior. Actualice los parámetros que desee cambiar. También puede añadir parámetros opcionales.

   Para obtener más información sobre los parámetros del archivo de entrada, consulte la descripción del archivo de definición de acciones en Paso 2: Crear un archivo de definición de tipo de acción.

   En el siguiente ejemplo, se muestra cómo actualizar un tipo de acción de ejemplo creado con el modelo de integración de Lambda. En este ejemplo, se realizan los siguientes cambios:

   - Cambie el nombre de `provider` a `TestProvider1`.
   - Añada un límite de tiempo de espera del trabajo de 900 segundos.
   - Añade una propiedad de configuración de acciones llamada `Host` que se muestra al cliente que utiliza la acción.

   ```
   {
       "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. En el terminal o la línea de comandos, ejecute el comando update-action-type.

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

Este comando devuelve el resultado del tipo de acción para que coincida con los parámetros actualizados.

# Crea y añade una acción personalizada en CodePipeline

AWS CodePipeline incluye una serie de acciones que le ayudan a configurar los recursos de creación, prueba e implementación para su proceso de lanzamiento automatizado. Si su proceso

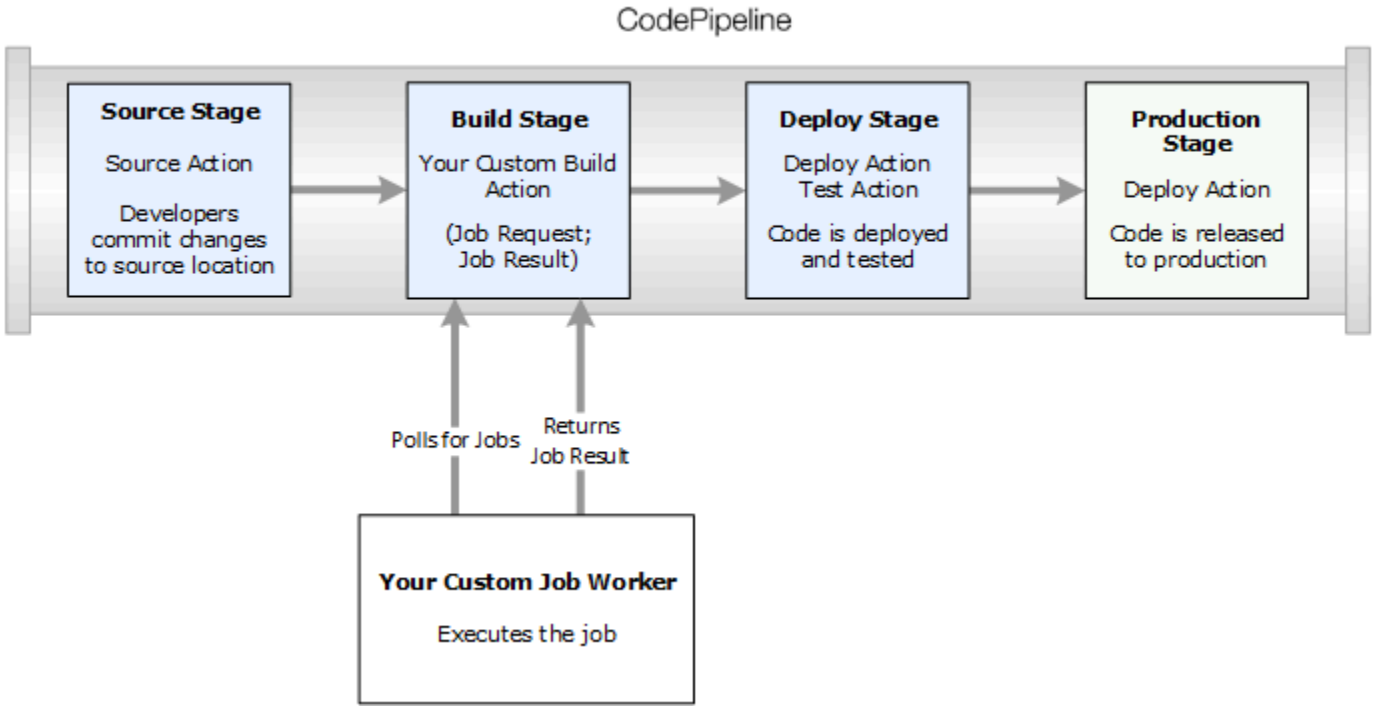
de lanzamiento incluye actividades que no figuran en las acciones predeterminadas, como un proceso de creación desarrollado internamente o un conjunto de pruebas, puede crear una acción personalizada para esas actividades e incluirla en su canalización. Puedes usarlo AWS CLI para crear acciones personalizadas en las canalizaciones asociadas a tu AWS cuenta.

Puedes crear acciones personalizadas para las siguientes categorías de AWS CodePipeline acciones:

- Una acción de creación personalizada que crea o transforma los elementos
- Una acción de implementación personalizada que implementa elementos en uno o varios servidores, sitios web o repositorios
- Una acción de prueba personalizada que configura y ejecuta pruebas automáticas
- Una acción de invocación personalizada que ejecuta funciones

Al crear una acción personalizada, también debe crear un trabajador que sondee las solicitudes de trabajo CodePipeline para esta acción personalizada, ejecute el trabajo y devuelva el resultado del estado a CodePipeline. Este proceso de trabajo puede estar ubicado en cualquier equipo o recurso que tenga acceso al punto de enlace público de CodePipeline. Para administrar el acceso y la seguridad fácilmente, sopese alojar su proceso de trabajo en una instancia de Amazon EC2.

Este diagrama ofrece una perspectiva de una canalización con una acción personalizada de creación:




Si una canalización incluye una acción personalizada en una etapa, la canalización creará una solicitud de trabajo. Un proceso de trabajo personalizado detectará esa solicitud y realizará el trabajo (en este ejemplo, un proceso personalizado que utiliza software de creación de terceros). Una vez realizada la acción, el proceso de trabajo devuelve un resultado que indica si el proceso se ha completado correctamente. Si se ha completado correctamente, la canalización proporcionará la revisión y sus artefactos a la siguiente acción. Si no se ha completado correctamente, la canalización no proporcionará a la revisión a la siguiente acción en la canalización.

> **Note**
>
> En estas instrucciones, se presupone que ya ha completado los pasos que se detallan en Empezar con CodePipeline.

Temas

- Crear una acción personalizada
- Crear un proceso de trabajo para la acción personalizada
- Agregar una acción personalizada a una canalización

# Crear una acción personalizada

Para crear una acción personalizada con AWS CLI

1. Abra un editor de texto y cree un archivo JSON para la acción personalizada que incluye la categoría de acción, el proveedor de la acción y cualquier configuración que requiera la acción del cliente. Por ejemplo, para crear una acción de compilación personalizada que requiera solo una propiedad, el archivo JSON podría tener este aspecto:

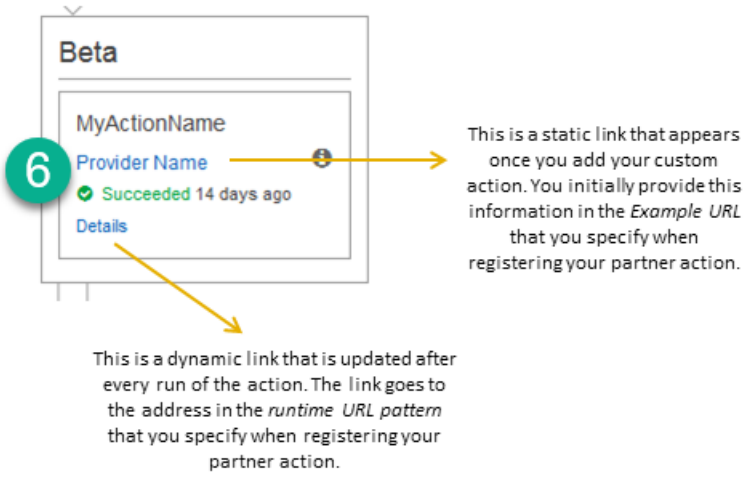
```
{
    "category": "Build",
    "provider": "My-Build-Provider-Name",
    "version": "1",
    "settings": {
        "entityUrlTemplate": "https://my-build-instance/job/{Config:ProjectName}/",
        "executionUrlTemplate": "https://my-build-instance/job/
{Config:ProjectName}/lastSuccessfulBuild/{ExternalExecutionId}/"
    },
    "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"
    }]
}
```

Este ejemplo añade el etiquetado a la acción personalizada incluyendo la clave de etiqueta `Project` y el valor `ProjectA` a la acción personalizada. Para obtener más información sobre cómo etiquetar los recursos en CodePipeline, consulteEtiquetado de recursos.

El archivo JSON incluye dos propiedades, `entityUrlTemplate` y `executionUrlTemplate`. Puede consultar un nombre en las propiedades de configuración de la acción personalizada dentro de las plantillas URL si sigue el formato de `{Config:`*`name`*`}`, siempre y cuando la propiedad de configuración sea obligatoria y no sea secreta. Por ejemplo, en el ejemplo anterior, el `entityUrlTemplate` valor hace referencia a la propiedad *`ProjectName`* de configuración.

- `entityUrlTemplate`: el enlace estático que proporciona información acerca del proveedor de servicios de la acción. En el ejemplo, el sistema de compilación incluye un enlace estático a cada proyecto de compilación. El formato del enlace variará según su proveedor de compilación (o bien, si crea un tipo de acción diferente, como una prueba, otro proveedor de servicios). Debe proporcionar este formato de enlace para que cuando se añada la acción del cliente, el usuario pueda elegir este enlace para abrir un explorador en una página de su sitio web que proporcione la información específica del proyecto de compilación (o entorno de prueba).
- `executionUrlTemplate`: el enlace dinámico que se actualizará con información acerca de la ejecución actual o más reciente de la acción. Cuando el empleado de trabajo personalizado actualiza el estado de este (por ejemplo, se ha realizado correctamente, error o en progreso), también proporcionará un `externalExecutionId` que se usará para completar el enlace. Este enlace se puede usar para proporcionar detalles acerca de la ejecución de una acción.

Por ejemplo, al ver la acción en la canalización, verá los siguientes dos enlaces:




1

Este enlace estático aparece después de añadir su acción personalizada y apunta a la dirección en `entityUrlTemplate`, la que especifica al crear su acción personalizada.

2

Este enlace dinámico se actualiza después de cada ejecución de la acción y apunta a la dirección en `executionUrlTemplate`, la que especifica al crear su acción personalizada.

Para obtener más información sobre estos tipos de enlaces, así como sobre `RevisionURLTemplate` y`ThirdPartyURL`, consulte ActionTypeSettings y CreateCustomActionType en la referencia de la CodePipeline API. Para obtener más información acerca de los requisitos de estructura de acción y cómo crear una acción, consulte CodePipeline referencia de estructura de tuberías.

2. Guarda el archivo JSON y asígnale un nombre que puedas recordar fácilmente (por ejemplo, *`MyCustomAction`* .json).
3. Abra una sesión de terminal (Linux, OS X, Unix) o el símbolo del sistema (Windows) en un equipo en el que haya instalado la AWS CLI.
4. Usa el AWS CLI para ejecutar el aws codepipeline create-custom-action-type comando y especifica el nombre del archivo JSON que acabas de crear.

   Por ejemplo, para crear una acción personalizada de compilación:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

```
aws codepipeline create-custom-action-type --cli-input-json
 file://MyCustomAction.json
```

5. Este comando devuelve toda la estructura de la acción personalizada que ha creado, así como la propiedad de configuración de la acción `JobList`, que ya se ha añadido. Al añadir una acción personalizada a una canalización, puede usar `JobList` para especificar los proyectos del proveedor que puede sondear en búsqueda de trabajos. Si no configura esto, todos los trabajos disponibles se devolverán cuando el empleado de trabajo personalizado sondee trabajos.

   Por ejemplo, el comando anterior podría devolver una estructura similar a la siguiente:

```
{
    "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**
>
> Como parte del resultado del create-custom-action-type comando, la `id` sección incluye`"owner": "Custom"`. CodePipeline se asigna automáticamente `Custom` como propietario de los tipos de acción personalizados. Este valor no se puede asignar ni cambiar cuando usa el comando create-custom-action-type o el comando update-pipeline.

## Crear un proceso de trabajo para la acción personalizada

Las acciones personalizadas requieren un trabajador que sondee las solicitudes de trabajo CodePipeline para la acción personalizada, ejecute el trabajo y devuelva el resultado del estado a CodePipeline. El proceso de trabajo puede estar ubicado en cualquier equipo o recurso que tenga acceso al punto de enlace público de CodePipeline.

El proceso de trabajo se puede diseñar de muchas formas. En las secciones siguientes se ofrecen orientaciones prácticas para desarrollar un proceso de trabajo personalizado para CodePipeline.

Temas

- Elegir y configurar una estrategia de administración de permisos para el proceso de trabajo
- Desarrollar un proceso de trabajo para una acción personalizada
- Arquitectura y ejemplos de procesos de trabajo personalizados

## Elegir y configurar una estrategia de administración de permisos para el proceso de trabajo

Para desarrollar un trabajador personalizado para su acción personalizada CodePipeline, necesitará una estrategia para integrar la administración de usuarios y permisos.

La estrategia más sencilla es añadir la infraestructura que necesita para el proceso de trabajo personalizado creando instancias de Amazon EC2 con un rol de instancia de Amazon EC2. Esto le permite ampliar fácilmente los recursos que necesite para la integración. Puedes utilizar la integración integrada con AWS para simplificar la interacción entre tu trabajador personalizado y CodePipeline.

Para configurar instancias de Amazon EC2

1. Obtenga más información acerca de Amazon EC2 y determine si es la opción correcta para su integración. Para más información, consulte Amazon EC2 - Alojamiento de servidores virtuales.
2. Comience a crear sus instancias de Amazon EC2. Para obtener información, consulte Introducción a instancias de Linux con Amazon EC2.

Otra estrategia que puede considerar es utilizar federación de identidades con IAM para integrar los recursos y el sistema de proveedor de identidad que ya tiene. Esta estrategia resulta particularmente útil si ya tiene un proveedor de identidad corporativo o una configuración que admita usuarios que utilicen proveedores de identidad web. La federación de identidades le permite conceder un acceso seguro a AWS los recursos CodePipeline, incluso sin tener que crear o administrar usuarios de IAM. Puede utilizar características y políticas sobre los requisitos de seguridad de contraseñas y la rotación de credenciales. Puede utilizar aplicaciones de ejemplo como plantillas para su propio diseño.

Para configurar la identidad federada

1. Obtenga más información acerca de la federación de identidades de IAM. Para obtener información, consulte Administrar federación.
2. Revise los ejemplos en Escenarios para conceder acceso temporal para identificar el escenario de acceso temporal que mejor se adapte a las necesidades de su acción personalizada.
3. Revise los ejemplos de códigos de identidad federada pertinentes a su infraestructura, como:
   - Aplicación de la muestra de identidad federada para un caso de uso de Active Directory

4. Comience a configurar una identidad federada. Para obtener información, consulte Federación y proveedores de identidades en la Guía de usuarios de IAM.

Crea una de las siguientes opciones para usarla Cuenta de AWS cuando ejecutes tu trabajador de trabajo y acción personalizado.

Los usuarios necesitan acceso programático si quieren interactuar con personas AWS ajenas a. Consola de administración de AWS La forma de conceder el acceso programático depende del tipo de usuario al que acceda. AWS

Para conceder acceso programático a los usuarios, elija una de las siguientes opciones.

| ¿Qué usuario necesita acceso programático? | Para | Mediante |
| --- | --- | --- |
| IAM | (Recomendado) Usa las credenciales de la consola como credenciales temporales para firmar las solicitudes programáticas a los AWS CLI AWS SDK o las API. AWS | Siga las instrucciones de la interfaz que desea utilizar:<br>• Para ello AWS CLI, consulte Inicio de sesión para el desarrollo AWS local en la Guía del AWS Command Line Interface usuario.<br>• Para AWS los SDK, consulte Iniciar sesión para el desarrollo AWS local en la Guía de referencia AWS sobre los SDK y las herramientas. |
| Identidad del personal<br><br>(Usuarios administrados en el IAM Identity Center) | Usa credenciales temporales para firmar las solicitudes programáticas a los AWS SDK o AWS CLI las API. AWS | Siga las instrucciones de la interfaz que desea utilizar:<br>• Para ello AWS CLI, consulte Configuración AWS CLI para su uso AWS IAM Identity Center en la Guía |

| ¿Qué usuario necesita acceso programático? | Para | Mediante |
| --- | --- | --- |
| | | del AWS Command Line Interface usuario.<br>• Para ver AWS los SDK, las herramientas y las AWS API, consulte la autentica ción del IAM Identity Center en la guía de referencia de AWS SDK y herramientas. |
| IAM | Utilice credenciales temporale s para firmar las solicitudes programáticas a los AWS SDK o las AWS CLI API. AWS | Siga las instrucciones que aparecen en la Guía del usuario de IAM, que se indican en la sección Uso de credenciales temporales con los AWS recursos. |

| ¿Qué usuario necesita acceso programático? | Para | Mediante |
| --- | --- | --- |
| IAM | (No recomendado)<br>Utilice credenciales a largo plazo para firmar las solicitud es programáticas a los AWS CLI AWS SDK o las API. AWS | Siga las instrucciones de la interfaz que desea utilizar:<br>• Para ello AWS CLI, consulte Autenticación mediante credenciales de usuario de IAM en la Guía del usuario. AWS Command Line Interface<br>• Para ver AWS los SDK y las herramientas, consulte Autenticar con credencia les de larga duración en la Guía de referencia sobre los AWS SDK y las herramien tas.<br>• Para ver AWS las API, consulte Administrar las claves de acceso para los usuarios de IAM en la Guía del usuario de IAM. |

A continuación se muestra una política de ejemplo que puede crear para utilizarla con un proceso de trabajo personalizado. Esta política es solo un ejemplo y se ofrece "tal cual".

JSON

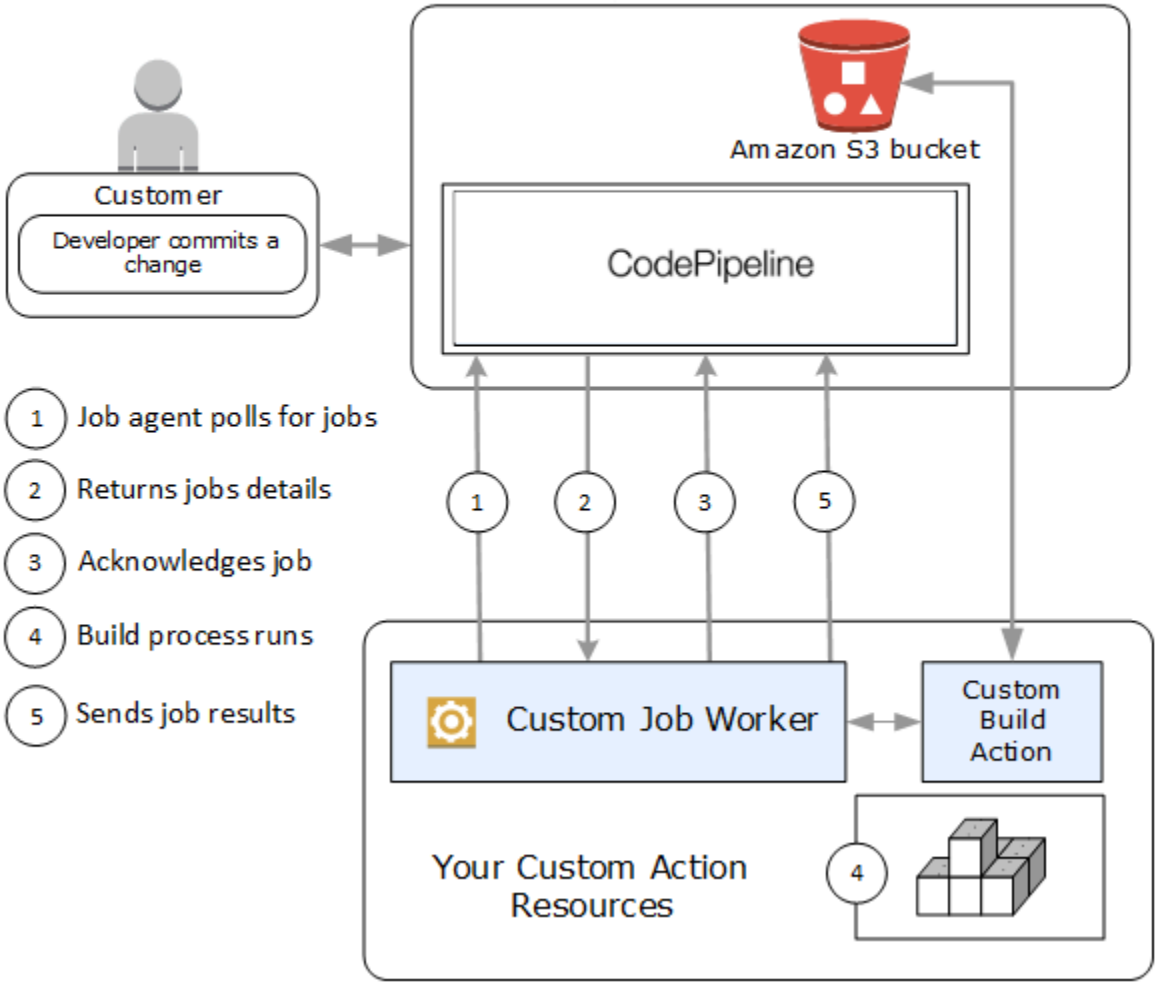
```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:PollForJobs",
```

```
        "codepipeline:AcknowledgeJob",
        "codepipeline:GetJobDetails",
        "codepipeline:PutJobSuccessResult",
        "codepipeline:PutJobFailureResult"
      ],
      "Resource": [
        "arn:aws:codepipeline:us-east-2::actionType:custom/
Build/MyBuildProject/1/"
      ]
    }
  ]
}
```

> **Note**
>
> Plantéese utilizar la política administrada `AWSCodePipelineCustomActionAccess`.

## Desarrollar un proceso de trabajo para una acción personalizada

Una vez que haya elegido su estrategia de administración de permisos, debe considerar la forma en que interactuará con su trabajador laboral. CodePipeline El siguiente diagrama muestra un flujo de trabajo de una acción personalizada y el proceso de trabajo de un proceso de creación.




1. Su trabajador CodePipeline busca trabajos en uso`PollForJobs`.
2. Cuando se activa una canalización debido a un cambio en su etapa de código fuente (por ejemplo, cuando un desarrollador confirma un cambio), se inicia el proceso de lanzamiento automático. Este proceso sigue hasta la etapa en la que se ha configurado la acción personalizada. Cuando llegue a tu acción en esta etapa, pondrá un trabajo en CodePipeline cola. El trabajo aparecerá si el proceso de trabajo invoca realiza una llamada `PollForJobs` de nuevo para obtener el estado. Tome los detalles del trabajo de `PollForJobs` y páselos de nuevo al proceso de trabajo.
3. El trabajador llama `AcknowledgeJob` para enviar CodePipeline un acuse de recibo del trabajo. CodePipeline devuelve un acuse de recibo que indica que el trabajador debe continuar con el trabajo (`InProgress`) o, si hay más de un trabajador buscando trabajo y otro trabajador ya lo ha reclamado, recibirá una respuesta de `InvalidNonceException` error. Tras la `InProgress` confirmación, CodePipeline espera a que se devuelvan los resultados.

4. El proceso de trabajo inicia la acción personalizada en la revisión y después se ejecuta la acción. Junto con otras acciones, la acción personalizada devuelve un resultado al proceso de trabajo. En el ejemplo de una acción personalizada de creación, la acción obtiene artefactos del bucket de Amazon S3, los compila y transmite artefactos compilados correctamente al bucket de Amazon S3.
5. Mientras se ejecuta la acción, el proceso de trabajo puede llamar a `PutJobSuccessResult` con un token de continuación (la serialización del estado del trabajo generada por el proceso de trabajo; por ejemplo, un identificador de compilación en formato JSON o una clave de objeto de Amazon S3) y también a la información de `ExternalExecutionId`, que se usará para rellenar el enlace de `executionUrlTemplate`. Esto añadirá en la vista de la canalización en la consola un enlace de trabajo a los detalles de una acción mientras se está realizando. Aunque no es obligatorio, resulta conveniente hacerlo porque de este modo los usuarios ven el estado de la acción personalizada mientras se ejecuta.

   En cuanto se ha llamado a `PutJobSuccessResult`, se considera que el trabajo se ha completado. Se crea un nuevo trabajo CodePipeline que incluye el token de continuación. Este trabajo aparecerá si el proceso de trabajo realiza de nuevo una llamada `PollForJobs`. El nuevo trabajo se puede usar para comprobar el estado de la acción: devolverá un token de continuación o, si la acción está completada, no lo devolverá.

   > **Note**
   >
   > Si el proceso de trabajo realiza todo el trabajo de la acción personalizada, debería plantearse dividirlo en por lo menos dos pasos. En el primer paso se establece la página de detalles de la acción. En cuanto haya creado la página de detalles, puede serializar el estado del proceso de trabajo y devolverlo como un token de continuación, cumpliendo los límites de tamaño (consulte Cuotas en AWS CodePipeline). Por ejemplo, podría escribir el estado de la acción en la cadena que utiliza como token de continuación. En el segundo paso (y los subsiguientes) del proceso de trabajo serían en los que se realizaría la tarea de la acción. El paso final devuelve el resultado exitoso o CodePipeline fallido, sin ningún símbolo de continuación en el paso final.

   Para obtener más información sobre el token de continuación, consulte las especificaciones de `PutJobSuccessResult` en la CodePipeline API Reference.
6. Una vez finalizada la acción personalizada, el trabajador devuelve el resultado de la acción personalizada CodePipeline llamando a una de las dos API siguientes:

- `PutJobSuccessResult` sin token de continuación, que indica que la acción personalizada se realizó correctamente
- `PutJobFailureResult`, que indica que la acción personalizada no se realizó correctamente

En función del resultado, la canalización proseguirá con la siguiente acción (si se ha realizado correctamente) o se detendrá (si no se ha realizado correctamente).

## Arquitectura y ejemplos de procesos de trabajo personalizados

Una vez planificado el flujo general de trabajo, puede crear el proceso de trabajo. Los detalles de la acción personalizada determinan en última instancia lo que va a necesitar para el proceso de trabajo. La mayoría de los procesos para acciones personalizadas incluyen las funcionalidades siguientes:

- Encuesta para ver si los trabajos están en CodePipeline uso`PollForJobs`.
- Reconocer los trabajos y devolver los resultados al CodePipeline uso `AcknowledgeJobPutJobSuccessResult`,, y`PutJobFailureResult`.
- Recuperar artefactos and/or al colocarlos en el bucket de Amazon S3 para la canalización. Para descargar artefactos del bucket de Amazon S3 debe crear un cliente de Amazon S3 que utilice la versión 4 de Signature (Sig V4). Se requiere el Sig V4 para AWS KMS.

  Para cargar artefactos al bucket de Amazon S3 deberá configurar la solicitud `PutObject` de Amazon S3 para que use el cifrado. Actualmente, solo se admite el servicio de administración de AWS claves (AWS KMS) para el cifrado. AWS KMS usos AWS KMS keys. Para saber si debe utilizar una clave gestionada por el cliente Clave administrada de AWS o una clave gestionada por el cliente para cargar artefactos, el trabajador personalizado debe examinar los datos de la tarea y comprobar la propiedad de la clave de cifrado. Si la propiedad está configurada, debe usar esa ID de clave gestionada por el cliente al realizar la configuración AWS KMS. Si la propiedad de la clave es nula, utilice la Clave administrada de AWS. CodePipeline usa el Clave administrada de AWS a menos que se configure lo contrario.

  Para ver un ejemplo que muestre cómo crear los AWS KMS parámetros en Java o.NET, consulte Especificar los AWS Key Management Service en Amazon S3 mediante los AWS SDK. Para obtener más información sobre el bucket de Amazon S3 para CodePipeline, consulteCodePipeline conceptos.

Un ejemplo más complejo de un trabajador personalizado está disponible en GitHub. El ejemplo es de código abierto y se proporciona "tal cual".

- Ejemplo de Job Worker para CodePipeline: descargue el ejemplo del GitHub repositorio.

## Agregar una acción personalizada a una canalización

Cuando tengas un trabajador, puedes añadir tu acción personalizada a una canalización creando una nueva y seleccionándola cuando utilices el asistente de creación de canalizaciones, editando una canalización existente y añadiendo la acción personalizada, o bien utilizando los AWS CLI SDK o las API.

> **Note**
>
> Puede crear una canalización con el asistente Crear canalización que incluya una acción personalizada si es una acción de compilación o implementación. Si la acción personalizada figura en la categoría de pruebas, debe añadirla modificando una canalización existente.

Temas

- Agregar una acción personalizada a una canalización existente (CLI)

### Agregar una acción personalizada a una canalización existente (CLI)

Puedes utilizarla AWS CLI para añadir una acción personalizada a una canalización existente.

1. Abra una sesión de terminal (Linux, macOS, or Unix) o el símbolo del sistema (Windows) y ejecute el comando get-pipeline para copiar la estructura de canalización que desee editar en un archivo JSON. Por ejemplo, para una canalización denominada **`MyFirstPipeline`**, debería escribir el siguiente comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando no devuelve nada, pero el archivo creado debería aparecer en el directorio en el que se ejecutó el comando.

2. Abra el archivo JSON en cualquier editor de texto y modifique la estructura del archivo para añadir su acción personalizada a la etapa existente.

> **Note**
>
> Si desea que su acción se ejecute en paralelo con otra acción en esa etapa, asegúrese de asignarla al mismo valor de `runOrder` que esa acción.

Por ejemplo, para modificar la estructura de una canalización que permita añadir una etapa denominada Build y añadir una acción personalizada de compilación a esa etapa, es posible modificar el JSON para añadir la etapa Build antes de una etapa de implementación de la siguiente manera:

```
,
    {
      "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. Para aplicar los cambios, ejecute el comando update-pipeline, especificando el archivo JSON de la canalización, de forma similar a como se muestra a continuación:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

Este comando devuelve la estructura completa de la canalización editada.

4. Abre la CodePipeline consola y elige el nombre de la canalización que acabas de editar.

La canalización muestra los cambios. La próxima vez que haga un cambio en la ubicación de código fuente, la canalización ejecutará dicha revisión a través de la estructura revisada de la canalización.

# Etiquetar una acción personalizada en CodePipeline

Las etiquetas son pares clave-valor asociados AWS a los recursos. Puede utilizar la consola o la CLI para aplicar etiquetas a las acciones personalizadas de CodePipeline. Para obtener información sobre el etiquetado de CodePipeline recursos, los casos de uso, las restricciones de clave y valor de las etiquetas y los tipos de recursos compatibles, consulte. Etiquetado de recursos

Puede agregar, eliminar y actualizar los valores de las etiquetas de una acción personalizada. Puede añadir hasta 50 etiquetas a cada acción personalizada.

Temas


- Incorporación de etiquetas en una acción personalizada
- Visualización de las etiquetas de una acción personalizada
- Edición de las etiquetas de una acción personalizada
- Eliminación de etiquetas de una acción personalizada


## Incorporación de etiquetas en una acción personalizada

Siga estos pasos para usar el AWS CLI para agregar una etiqueta a una acción personalizada. Para añadir una etiqueta a una acción personalizada al crearla, consulte Crea y añade una acción personalizada en CodePipeline.

En estos pasos, se presupone que ya ha instalado una versión reciente de la AWS CLI o que la ha actualizado a la versión actual. Para obtener más información, consulte Instalación de la AWS Command Line Interface.

En el terminal o la línea de comandos, ejecute el comando tag-resource, especificando el nombre de recurso de Amazon (ARN) de la acción personalizada a la que desea añadir la etiqueta, y la clave y el valor de la etiqueta que desea añadir. Puede añadir más de una etiqueta a una acción personalizada. Por ejemplo, para etiquetar una acción personalizada con dos etiquetas, una clave de etiqueta `TestActionType` con el valor de etiqueta de `UnitTest` y una clave de etiqueta `ApplicationName` con el valor de etiqueta de`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
```

Si se ejecuta correctamente, este comando no devuelve nada.

## Visualización de las etiquetas de una acción personalizada

Siga estos pasos para usar el AWS CLI para ver las AWS etiquetas de una acción personalizada. Si no se han añadido etiquetas, la lista obtenida está vacía.

En el terminal o la línea de comandos, ejecute el comando list-tags-for-resource. Por ejemplo, para ver una lista de las claves y los valores de las etiquetas para una acción personalizada con el ARN `arn:aws:codepipeline:us-west-2:account-id:actiontype:Owner/Category/Provider/Version`:

```
aws codepipeline list-tags-for-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:actiontype:Owner/Category/Provider/Version
```

Si se ejecuta correctamente, este comando proporciona información similar a la siguiente:

```
{
    "tags": {
        "TestActionType": "UnitTest",
        "ApplicationName": "MyApplication"
    }
}
```

## Edición de las etiquetas de una acción personalizada

Siga estos pasos para usar el AWS CLI para editar una etiqueta para una acción personalizada. Puede cambiar el valor de una clave existente o añadir otra clave. También puede eliminar etiquetas de una acción personalizada, tal y como se muestra en la sección siguiente.

En el terminal o la línea de comandos, ejecute el comando tag-resource, especificando el nombre de recurso de Amazon (ARN) de la acción personalizada en la que desea actualizar una etiqueta, y especifique la clave y el valor de la etiqueta:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:actiontype:Owner/Category/Provider/Version --tags
 key=TestActionType,value=IntegrationTest
```

## Eliminación de etiquetas de una acción personalizada

Siga estos pasos para usar el AWS CLI para eliminar una etiqueta de una acción personalizada. Cuando se quitan etiquetas del recurso asociado, las etiquetas se eliminan.

> **Note**
>
> Si se elimina una acción personalizada, todas las asociaciones de etiquetas se quitan de la acción personalizada eliminada. No es necesario eliminar las etiquetas antes de eliminar una acción personalizada.

En el terminal o la línea de comandos, ejecute el comando untag-resource, especificando el ARN de la acción personalizada en la que desea eliminar las etiquetas y la clave de etiqueta de la etiqueta que desea eliminar. Por ejemplo, para eliminar una etiqueta de una acción personalizada con la clave de etiqueta*TestActionType*:

```
aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-
id:actiontype:Owner/Category/Provider/Version --tag-keys TestActionType
```

Si se ejecuta correctamente, este comando no devuelve nada. Para verificar las etiquetas asociadas a la acción personalizada, ejecute el comando list-tags-for-resource.

# Invoca un AWS Lambda función en una canalización en CodePipeline

AWS Lambda es un servicio automático que permite ejecutar código sin aprovisionar ni administrar servidores. Puede crear funciones de Lambda y añadirlas como acciones en las canalizaciones. Puesto que Lambda le permite escribir funciones para realizar casi cualquier tarea, puede personalizar el funcionamiento de la canalización.

> **Important**
>
> No registres el evento JSON que se CodePipeline envía a Lambda, ya que esto puede provocar que las credenciales de usuario se registren en CloudWatch los registros. El CodePipeline rol usa un evento JSON para pasar credenciales temporales a Lambda en el `artifactCredentials` campo. Para ver un ejemplo de evento, consulte Ejemplo de evento JSON.

Estos son modos posibles de usar las funciones de Lambda en las canalizaciones:

- Crear recursos bajo demanda en una etapa de una canalización CloudFormation y eliminarlos en otra etapa.
- Para implementar versiones de aplicaciones sin tiempo de inactividad, utilice una función de Lambda que intercambie los valores de CNAME. AWS Elastic Beanstalk
- Para implementar instancias Docker de Amazon ECS.
- Para hacer una copia de seguridad (es decir, crear un snapshot de AMI) antes de proceder a la creación o la implementación.
- Para hacer posible la integración con productos de terceros en su canalización, por ejemplo para publicar mensajes en un cliente IRC.

> **Note**
>
> La creación y ejecución de funciones de Lambda pueden generar cargos en su cuenta. AWS Para obtener más información, consulte Precios.

En este tema se presupone que está familiarizado AWS CodePipeline AWS Lambda y sabe cómo crear canalizaciones, funciones y las políticas y funciones de IAM de las que dependen. En este tema puede ver cómo:

- Crear una función de Lambda que pruebe si se ha implementado correctamente una página web.
- Configure los CodePipeline roles de ejecución de Lambda y los permisos necesarios para ejecutar la función como parte de la canalización.
- Editar una canalización para añadir la función de Lambda como una acción.
- Probar la acción lanzando un cambio manualmente.

> **Note**
>
> Al utilizar la acción de invocación de Lambda entre regiones CodePipeline, el estado de la ejecución de Lambda con el signo PutJobSuccessResult y PutJobFailureResult debe enviarse a la AWS región en la que esté presente la función de Lambda y no a la región en la que exista. CodePipeline

En este tema se incluyen ejemplos de funciones para demostrar la flexibilidad de trabajar con funciones de Lambda en: CodePipeline

- Basic Lambda function
  - Crear una función Lambda básica con la que utilizarla. CodePipeline
  - Si se devuelve el éxito o el error, CodePipeline aparecen en el enlace de detalles de la acción.
- Ejemplo de función de Python que usa un AWS CloudFormation plantilla
  - Usar parámetros JSON-encoded de usuario para pasar varios valores de configuración a la función (`get_user_params`).
  - Interactuar con artefactos .zip en un bucket de artefactos (`get_template`)
  - Usar un token de continuación para monitorizar procesos asíncronos de ejecución prolongada (`continue_job_later`). Esto permite que la acción continúe y la función se realice correctamente aunque se supere el tiempo de ejecución de quince minutos (esto es un límite de Lambda).

Cada función de ejemplo incluye información sobre los permisos que debe añadir al rol. Para obtener información sobre los límites en AWS Lambda, consulte los límites en la Guía para AWS Lambda desarrolladores.

> **Important**
>
> Los roles, las políticas y el código de muestra que se incluyen en este tema son meramente ilustrativos y se ofrecen "tal cual".

Temas

- Paso 1: Crear una canalización
- Paso 2: Crear la función de Lambda

- Paso 3: Agregue la función Lambda a una canalización en la consola CodePipeline
- Paso 4: Probar la canalización con la función de Lambda
- Paso 5: Siguientes pasos
- Ejemplo de evento JSON
- Más funciones de ejemplo


## Paso 1: Crear una canalización

En este paso, va a crear una canalización a la que le añadirá luego la función de Lambda. Es la misma canalización que creó en CodePipeline tutoriales. Si esa canalización sigue estando configurada en su cuenta y se encuentra en la misma región en la que tiene pensado crear la función de Lambda, puede omitir este paso.

Para crear la canalización

1. Siga los tres primeros pasos Tutorial: Crear una canalización simple (bucket de S3) para crear un bucket de Amazon S3, CodeDeploy recursos y una canalización en dos etapas. Elija la opción Amazon Linux para los tipos de instancias. Puede usar el nombre que desee para la canalización, pero los pasos en este tema usan MyLambdaTestPipeline.
2. En la página de estado de tu canalización, en la CodeDeploy acción, elige Detalles. En la página de detalles de la implementación del grupo de implementaciones, elija un ID de instancia de la lista.
3. En la consola de Amazon EC2, en la pestaña Detalles de la instancia, copie la dirección IP en Dirección IP pública (por ejemplo, **`192.0.2.4`**). Usará esta dirección como objetivo de la función en AWS Lambda.

> **Note**
>
> La política de funciones de servicio predeterminada CodePipeline incluye los permisos de Lambda necesarios para invocar la función. Sin embargo, si ha modificado el rol de servicio predeterminado o seleccionado uno distinto, asegúrese de que la política o el rol admiten los permisos de `lambda:InvokeFunction` y `lambda:ListFunctions`. De lo contrario, las canalizaciones que incluyan las acciones de Lambda fallarán.

# Paso 2: Crear la función de Lambda

En este paso, va a crear una función de Lambda que realiza una solicitud HTTP y busca una línea de texto en una página web. Durante este paso, también creará una política de IAM y un rol de ejecución de Lambda. Para obtener más información, consulte Modelo de permisos en la Guía para desarrolladores de AWS Lambda .

Para crear el rol de ejecución

1. Inicie sesión en Consola de administración de AWS y abra la consola de IAM en. https://console.aws.amazon.com/iam/
2. Elija Policies y después, Create Policy. Seleccione la pestaña JSON y pegue la siguiente política en el campo correspondiente.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Action": [
        "codepipeline:PutJobSuccessResult",
        "codepipeline:PutJobFailureResult"
        ],
        "Effect": "Allow",
        "Resource": "*"
     }
  ]
}
```

3. Elija Revisar política.

4. En la página Review policy (Revisar política), en Name (Nombre), escriba un nombre para la política (por ejemplo, **`CodePipelineLambdaExecPolicy`**). En Description (Descripción), escriba **`Enables Lambda to execute code`**.

   Seleccione Crear política.

   > **Note**
   >
   > Estos son los permisos mínimos necesarios para que una función de Lambda interopere con CodePipeline Amazon. CloudWatch Si desea ampliar esta política para permitir las funciones que interactúan con otros AWS recursos, debe modificarla para permitir las acciones que requieren esas funciones de Lambda.

5. En la página del panel de la política, elija Roles (Funciones) y, a continuación, elija Create role (Crear función).
6. En la página Crear rol, elija Servicio de AWS. Elija Lambda y, a continuación, elija Siguiente: permisos.
7. En la página Attach permissions policies (Asociar políticas de permiso), seleccione la casilla de verificación junto a CodePipelineLambdaExecPolicy y, a continuación, elija Next: Tags (Siguiente: Etiquetas). Elija Siguiente: Revisar.
8. En la página Review (Revisión), en Role name (Nombre del rol), escriba el nombre que quiera darle y, a continuación, seleccione Create role (Crear rol).

Para crear la función Lambda de ejemplo que se utilizará con CodePipeline

1. Inicie sesión en Consola de administración de AWS y abra la AWS Lambda consola en https://console.aws.amazon.com/lambda/.
2. En la página Functions (Funciones), seleccione Create function (Crear función).

   > **Note**
   >
   > Si ve una página Bienvenida en lugar de la página Lambda elija Empezar ahora.

3. En la página Crear función, elija Diseñar desde cero. En Nombre de función, escriba un nombre para la función de Lambda (por ejemplo, **`MyLambdaFunctionForAWSCodePipeline`**). En Runtime, elige Node.js 20.x.

4. En Role (Rol), seleccione Choose an existing role (Elegir un rol existente). En Existing role (Función existente), seleccione su función y, a continuación, Create function (Crear función).

   Se abrirá la página de detalles de la función creada.

5. Copie el siguiente código en el cuadro Function code (Código de función):

> **Note**
>
> El objeto de evento, debajo de la CodePipeline.job clave, contiene los detalles del trabajo. Para ver un ejemplo completo de cómo el evento JSON CodePipeline regresa a Lambda, consulteEjemplo de evento JSON.

```
import { CodePipelineClient, PutJobSuccessResultCommand,
 PutJobFailureResultCommand } from "@aws-sdk/client-codepipeline";
import http from 'http';
import assert from 'assert';

export const handler = (event, context) => {

    const codepipeline = new CodePipelineClient();

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

    // Retrieve the value of UserParameters from the Lambda action configuration in
 CodePipeline, in this case a URL which will be
    // health checked by this function.
    const url =
 event["CodePipeline.job"].data.actionConfiguration.configuration.UserParameters;

    // Notify CodePipeline of a successful job
    const putJobSuccess = async function(message) {
        const command = new PutJobSuccessResultCommand({
            jobId: jobId
        });
         try {
             await codepipeline.send(command);
             context.succeed(message);
         } catch (err) {
             context.fail(err);
```

```
        }
    };

    // Notify CodePipeline of a failed job
    const putJobFailure = async function(message) {
        const command = new PutJobFailureResultCommand({
            jobId: jobId,
            failureDetails: {
                message: JSON.stringify(message),
                type: 'JobFailed',
                externalExecutionId: context.awsRequestId
            }
        });
        await codepipeline.send(command);
        context.fail(message);
    };

    // Validate the URL passed in UserParameters
    if(!url || url.indexOf('http://') === -1) {
        putJobFailure('The UserParameters field must contain a valid URL address to
test, including http:// or https://');
        return;
    }

    // Helper function to make a HTTP GET request to the page.
    // The helper will test the response and succeed or fail the job accordingly
    const getPage = function(url, callback) {
        var pageObject = {
            body: '',
            statusCode: 0,
            contains: function(search) {
                return this.body.indexOf(search) > -1;
            }
        };
        http.get(url, function(response) {
            pageObject.body = '';
            pageObject.statusCode = response.statusCode;

            response.on('data', function (chunk) {
                pageObject.body += chunk;
            });

            response.on('end', function () {
                callback(pageObject);
```

```
            });

            response.resume();
        }).on('error', function(error) {
            // Fail the job if our request failed
            putJobFailure(error);
        });
    };

    getPage(url, function(returnedPage) {
        try {
            // Check if the HTTP response has a 200 status
            assert(returnedPage.statusCode === 200);
            // Check if the page contains the text "Congratulations"
            // You can change this to check for different text, or add other tests
 as required
            assert(returnedPage.contains('Congratulations'));

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

6. Deje el valor predeterminado de Handler (Controlador) y, en Role (Función), deje **`CodePipelineLambdaExecRole`** como la opción predeterminada.
7. En Basic settings (Configuración básica), para Timeout (Tiempo de espera), introduzca **`20`** segundos.
8. Seleccione Save.

## Paso 3: Agregue la función Lambda a una canalización en la consola CodePipeline

En este paso, añadirá una nueva etapa a la canalización y después añadirá una acción de Lambda que llame a la función a esa etapa.

Para añadir una etapa

1. Inicie sesión en Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. En la página Welcome (Bienvenida), elija la canalización que acaba de crear.
3. En la página para ver la canalización, elija Edit.
4. En la página de edición, selecciona + Añadir fase para añadir una fase después de la fase de despliegue con la CodeDeploy acción. Escriba un nombre para la etapa (por ejemplo, **`LambdaStage`**) y elija Add stage (Añadir etapa).

   > **Note**
   >
   > También puede añadir su acción de Lambda a una etapa. Para fines de demostración, añadiremos la función de Lambda como la única acción de una etapa para que pueda ver fácilmente su progreso a medida que los artefactos progresan en la canalización.

5. Elija + Add action group (Añadir grupo de acciones). En el panel Editar acción, en Nombre de acción, escriba un nombre para la acción de Lambda (por ejemplo, **`MyLambdaAction`**). En Provider (Proveedor), elija AWS Lambda. En Nombre de función, elija o escriba el nombre de la función de Lambda (por ejemplo, **`MyLambdaFunctionForAWSCodePipeline`**). En Parámetros de usuario, especifique la dirección IP de la instancia de Amazon EC2 que ha copiado previamente (por ejemplo, **`http://`*`192.0.2.4`***) y, a continuación, elija Listo.

   > **Note**
   >
   > En este tema se usa una dirección IP, pero en una situación real podría proporcionar el nombre de su sitio web registrado (por ejemplo, **`http://`*`www.example.com`***). Para obtener más información sobre los datos y los controladores de eventos AWS Lambda, consulte el modelo de programación en la Guía para AWS Lambda desarrolladores.

6. En la página Edit action (Editar acción), elija Save (Guardar).

## Paso 4: Probar la canalización con la función de Lambda

Para probar la función, lance el cambio más reciente en la canalización.

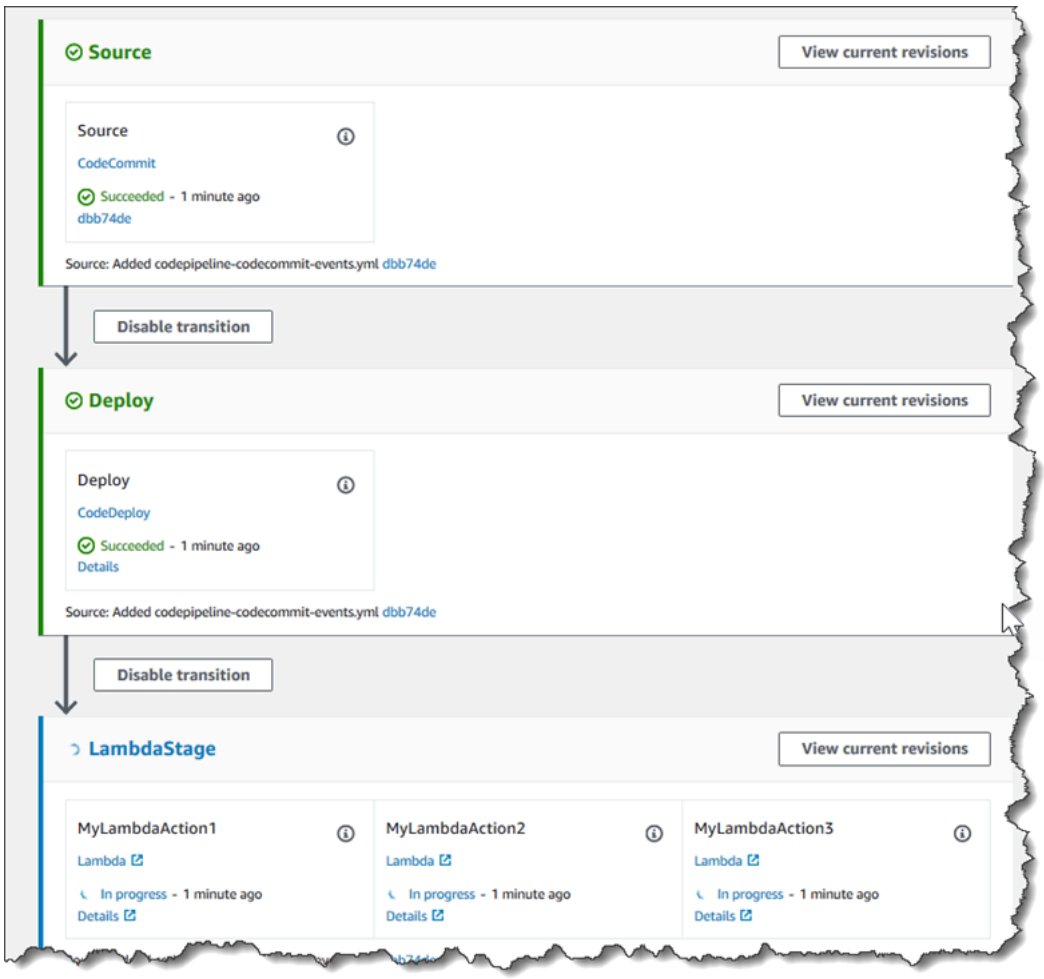
Para usar la consola para ejecutar la versión más reciente de un artefacto en una canalización

1. En la página de detalles de la canalización, elija Liberar cambio. Esto ejecuta la revisión más reciente disponible en cada ubicación de código fuente especificada en una acción de código fuente a través de la canalización.
2. Cuando finalice la acción de Lambda, seleccione el  enlace  Detalles para ver el flujo de registros de la función en Amazon CloudWatch, incluida la duración facturada del evento. Si la función falló, el CloudWatch registro proporciona información sobre la causa.

## Paso 5: Siguientes pasos

Ahora que ya ha creado una función de Lambda y la ha añadido como acción en una canalización, puede intentar lo siguiente:

- Añadir más acciones de Lambda a una etapa para comprobar otras páginas web.
- Modificar la función de Lambda para comprobar una cadena de texto distinta.
- Explorar las funciones de Lambda y crear y añadir sus propias funciones de lambda a las canalizaciones.




Cuando hayas terminado de experimentar con la función Lambda, considera la posibilidad de eliminarla de tu canalización, eliminarla de AWS Lambda IAM y eliminar el rol de IAM para evitar posibles cargos. Para obtener más información, consulte Editar una canalización en CodePipeline, Eliminar una canalización en CodePipeline y Eliminación de roles o perfiles de instancia.

## Ejemplo de evento JSON

En el siguiente ejemplo, se muestra un ejemplo de un evento JSON enviado a Lambda por. CodePipeline La estructura de este evento se parece a la respuesta a la `GetJobDetails API`, pero sin los tipos de datos `actionTypeId` ni `pipelineContext`. Se incluyen dos detalles de configuración de acción, `FunctionName` y `UserParameters`, tanto en el evento JSON como en la respuesta a la API `GetJobDetails`. Los valores *`red italic text`* que aparecen son ejemplos o explicaciones, no valores reales.

```
{
    "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"
            }
        }
    }
}
```

## Más funciones de ejemplo

Los siguientes ejemplos de funciones de Lambda muestran funciones adicionales que puedes usar en tus canalizaciones. CodePipeline Para utilizar estas funciones, puede que sea necesario modificar la política del rol de ejecución de Lambda, como se indica en la introducción de cada ejemplo.

Temas

- Ejemplo de función de Python que usa un AWS CloudFormation plantilla

### Ejemplo de función de Python que usa un AWS CloudFormation plantilla

El siguiente ejemplo muestra una función que crea o actualiza una pila en función de una CloudFormation plantilla proporcionada. La plantilla crea un bucket de Amazon S3. Tiene una finalidad ilustrativa únicamente, para reducir costos. Lo ideal es eliminar la pila antes de cargar cosas en el bucket. Si carga archivos en el bucket, no podrá eliminarlo cuando elimine la pila. Tendrá que quitar manualmente el contenido del bucket para poder eliminar el bucket en sí.

En este ejemplo de Python, se presupone que tiene una canalización que utiliza un bucket de Amazon S3 como acción de origen o bien que tiene acceso a un bucket de Amazon S3 con control de versiones que puede usar con la canalización. Creas la CloudFormation plantilla, la comprimes y la subes a ese bucket como un archivo.zip. Luego añadirá una acción de código fuente a la canalización para recuperar el archivo .zip del bucket.

> **Note**
>
> Si Amazon S3 es el proveedor de origen de la canalización, debe comprimir los archivos de origen en un solo archivo.zip y cargarlo en el bucket de origen. También puede cargar un

archivo sin comprimir; sin embargo, se producirá un error en las acciones posteriores que esperan un archivo.zip.

El siguiente ejemplo muestra:

- El uso de parámetros de JSON-encoded usuario para pasar varios valores de configuración a la función (`get_user_params`).
- La interacción con artefactos .zip en un bucket de artefactos (`get_template`)
- El uso de un token de continuación para monitorizar procesos asíncronos de ejecución prolongada (`continue_job_later`) Esto permite que la acción continúe y la función se realice correctamente aunque se supere el tiempo de ejecución de quince minutos (esto es un límite de Lambda).

Para utilizar este ejemplo de función de Lambda, la política del rol de ejecución de Lambda debe tener `Allow` permisos en CloudFormation Amazon S3 y CodePipeline, como se muestra en este ejemplo de política:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:logs:*:*:*"
        },
        {
            "Action": [
                "codepipeline:PutJobSuccessResult",
                "codepipeline:PutJobFailureResult"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
```

```
            "Action": [
                "cloudformation:DescribeStacks",
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Para crear la CloudFormation plantilla, abra cualquier editor de texto sin formato y copie y pegue el siguiente código:

```
{
  "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"
    }
  }
}
```

Guárdelo como un archivo JSON con el nombre **`template.json`** en un directorio denominado **`template-package`**. Cree un archivo comprimido (.zip) con este directorio, asígnele el nombre

**`template-package.zip`** y cárguelo en un bucket de Amazon S3 con control de versiones. Si ya tiene un bucket configurado para la canalización, puede usarlo. Edite la canalización para añadir una acción de código fuente que recupere el archivo .zip. Asigne un nombre a la salida para esta acción. *`MyTemplate`* Para obtener más información, consulte Editar una canalización en CodePipeline.

> **Note**
>
> La función de Lambda de ejemplo espera estos nombres de archivo y esta estructura comprimida. Sin embargo, puede sustituir este ejemplo por su propia CloudFormation plantilla. Si usa su propia plantilla, asegúrese de modificar la política del rol de ejecución de Lambda para permitir cualquier funcionalidad adicional que necesite la CloudFormation plantilla.

Para añadir el siguiente código como función en Lambda;

1. Abra la consola de Lambda; y elija Crear función.
2. En la página Crear función, elija Diseñar desde cero. En Nombre de función, introduzca un nombre para la función de Lambda.
3. En Runtime (Tiempo de ejecución), elija Python 2.7.
4. En Elegir o crear un rol de ejecución, seleccione Usar un rol existente. En Existing role (Función existente), seleccione su función y, a continuación, Create function (Crear función).

   Se abrirá la página de detalles de la función creada.
5. Copie el siguiente código en el cuadro Function code (Código de función):

```
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. Deje Controlador con el valor predeterminado y deje Rol con el nombre que seleccionó o creó anteriormente, **`CodePipelineLambdaExecRole`**.
7. En Basic settings (Configuración básica), para Timeout (Tiempo de espera), sustituya el valor predeterminado de 3 segundos por **`20`**.
8. Seleccione Save.
9. Desde la CodePipeline consola, edita la canalización para agregar la función como una acción en una etapa de tu canalización. Seleccione Editar para la etapa de la canalización que deseas cambiar y selecciona Añadir grupo de acciones. En la página Editar acción, en Nombre de la acción, introduzca un nombre para su acción. En Proveedor de la acción, elija Lambda.

   En Artefactos de entrada, elija `MyTemplate`. En UserParameters, debes proporcionar una cadena JSON con tres parámetros:

   - Nombre de pila
   - CloudFormation nombre de la plantilla y ruta al archivo
   - Artefactos de entrada

Use corchetes ({ }) y separe los parámetros con comas. Por ejemplo, para crear una pila denominada*MyTestStack*, para una canalización con el artefacto de entrada *MyTemplate* UserParameters, escriba: {"stack»:» *MyTestStack* «, "file» :"template- package/template .json», "artifact»:» «}. *MyTemplate*

> **Note**
>
> Aunque hayas especificado el artefacto de entrada en UserParameters, también debes especificar este artefacto de entrada para la acción en Artefacto de entrada.

10. Guarde los cambios en la canalización y lance manualmente un cambio para probar la acción y la función de Lambda.

## Incorporación de una acción de aprobación manual a una etapa

En AWS CodePipeline, puede añadir una acción de aprobación a una fase de una canalización en el punto en el que desee que se detenga la ejecución de la canalización para que alguien con los AWS Identity and Access Management permisos necesarios pueda aprobar o rechazar la acción.

Si la acción se aprueba, se reanuda la ejecución de la canalización. Si la acción es rechazada, o si nadie aprueba o rechaza la acción en un plazo de siete días desde que la canalización haya tomado la acción y se detenga, el resultado es el mismo que si la acción fracasa y la ejecución de la canalización no continúa.

Se puede realizar aprobaciones manuales por tres razones:

- Desea que alguien realice una revisión de código o una revisión de administración de cambios para que se permita una corrección en la siguiente etapa de una canalización.
- Desea que alguien realice una prueba de control de calidad manual en la última versión de una aplicación o para confirmar la integridad de un artefacto de compilación antes de su lanzamiento.
- Desea que alguien revise el texto nuevo o actualizado antes de que se publique en el sitio web de una compañía.

# Opciones de configuración para las acciones de aprobación manual en CodePipeline

CodePipeline proporciona tres opciones de configuración que puede usar para informar a los aprobadores sobre la acción de aprobación.

Publicar notificaciones de aprobación Puede configurar una acción de aprobación para publicar un mensaje en un tema del Amazon Simple Notification Service cuando la acción se detenga en el momento de la acción. Amazon SNS envía el mensaje a todos los puntos de conexión suscritos al tema. Debes usar un tema creado en la misma AWS región que el proceso que incluirá la acción de aprobación. Cuando cree un tema, recomendamos que le asigne un nombre que identifique su finalidad, con un formato como `MyFirstPipeline-us-east-2-approval`.

Al publicar notificaciones de aprobación en temas de Amazon SNS, puede elegir entre formatos como los destinatarios de correo electrónico o SMS, las colas de SQS, los HTTP/HTTPS puntos de enlace o AWS Lambda las funciones que invoca mediante Amazon SNS. Para obtener información acerca de las notificaciones de temas de Amazon SNS, consulte los siguientes temas.

- ¿Qué es Amazon Simple Notification Service?
- Crear un tema en Amazon SNS
- Sending Amazon SNS Messages to Amazon SQS Queues
- Subscribing a Queue to an Amazon SNS Topic
- Envío de mensajes de Amazon SNS a puntos de enlace HTTP/HTTPS
- Invoking Lambda Functions Using Amazon SNS Notifications

Para conocer la estructura de los datos JSON que se generan para una notificación de acción de aprobación, consulte Formato de datos JSON para las notificaciones de aprobación manual en CodePipeline.

Specify a URL for Review Como parte de la configuración de la acción de aprobación, puede especificar una URL que revisar. La URL puede ser un enlace a una aplicación web que quiere que prueben los aprobadores o una página con más información acerca de la solicitud de aprobación. La URL se incluye en la notificación que se publica para el tema de Amazon SNS. Los aprobadores pueden usar la consola o la CLI para verla.

Enter Comments for Approvers (Escribir comentarios para aprobadores) Cuando crea una acción de aprobación, también puede añadir comentarios dirigidos a quienes reciben las notificaciones o quienes vean la acción en la respuesta de la consola o la CLI.

No Configuration Options Puede decidir no configurar ninguna de estas tres opciones. Es posible que no las necesite si, por ejemplo, puede notificar a alguien que la acción está lista para la revisión o simplemente quiere que la canalización se detenga hasta que decida aprobarla usted mismo.

## Descripción general de la configuración y el flujo de trabajo para las acciones de aprobación en CodePipeline

A continuación se ofrece información general sobre la configuración y el uso de las aprobaciones manuales.

1. Los permisos de IAM necesarios para aprobar o rechazar las acciones de aprobación se conceden a uno o varios roles de IAM de la organización.
2. (Opcional) Si utiliza las notificaciones de Amazon SNS, asegúrese de que el rol de servicio que utiliza en sus CodePipeline operaciones tenga permiso para acceder a los recursos de Amazon SNS.
3. (Opcional) Si utiliza notificaciones de Amazon SNS, crea un tema de Amazon SNS y le añade uno o más suscriptores o punto de conexión.
4. Cuando usa la AWS CLI para crear la canalización o después de usar la CLI o la consola para crear la canalización, agrega una acción de aprobación a una etapa de la canalización.

   Si utiliza notificaciones, incluye el Nombre de recurso de Amazon (ARN) del tema de Amazon SNS en la configuración de la acción. (Un ARN es un identificador único para un recurso de Amazon. Los ARN de los temas de Amazon SNS están estructurados de la misma manera. `arn:aws:sns:us-east-2:80398EXAMPLE:MyApprovalTopic` Para obtener más información, consulte los nombres de recursos de Amazon (ARN) y los Servicio de AWS espacios de nombres en.) Referencia general de Amazon Web Services
5. La canalización se detiene cuando alcanza la acción de aprobación. Si se incluyó el ARN de un tema de Amazon SNS en la configuración de la acción, se publica una notificación para el tema de Amazon SNS y se entrega un mensaje a los suscriptores del tema o los puntos de conexión suscritos, con un enlace a la acción de aprobación en la consola.
6. Un aprobador examina la URL de destino y revisa los comentarios que haya.
7. El aprobador, con ayuda de la consola, la CLI o el SDK, resume un comentario y envía una respuesta:

- Aprobado: se reanuda la ejecución de la canalización.
- Rechazado: el estado de la etapa cambia a "Failed" y la ejecución de la canalización no se reanuda.

Si no se envía ninguna respuesta en siete días, la acción se marca como "Failed".

## Conceda permisos de aprobación a un usuario de IAM en CodePipeline

Para que los usuarios de IAM de una organización puedan aprobar o rechazar acciones de aprobación, deben recibir permisos para obtener acceso a las canalizaciones y para actualizar el estado de las acciones de aprobación. Puede conceder permiso para acceder a todas las canalizaciones y acciones de aprobación de una cuenta si asocia la política administrada `AWSCodePipelineApproverAccess` a un usuario, un rol o un grupo de IAM; o también puede conceder permisos limitados si especifica los recursos individuales a los que puede obtener acceso un usuario, un rol o un grupo de IAM.

> **Note**
>
> Los permisos descritos en este tema conceden un acceso muy limitado. Para permitir que un usuario, rol o grupo pueda hacer algo más que aprobar o rechazar acciones de aprobación, puede asociar otras políticas administradas. Para obtener información sobre las políticas administradas disponibles para las que está disponible CodePipeline, consulteAWS políticas gestionadas para AWS CodePipeline.

### Concesión de permisos de aprobación a todas las canalizaciones y acciones de aprobación

Para los usuarios que necesitan realizar acciones de aprobación CodePipeline, utilice la política `AWSCodePipelineApproverAccess` gestionada.

Para dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en AWS IAM Identity Center:

  Cree un conjunto de permisos. Siga las instrucciones de Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center .
- Usuarios gestionados en IAM a través de un proveedor de identidades:

Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.

- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o agregue un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

## Especificación del permiso de aprobación para determinadas canalizaciones y acciones de aprobación

Para los usuarios que necesitan realizar acciones de aprobación en CodePipeline, utilice la siguiente política personalizada. En la política siguiente, especifique los recursos individuales a los que puede acceder un usuario. Por ejemplo, la siguiente política otorga permisos a los usuarios para aprobar o rechazar solo la acción denominada `MyApprovalAction` en la canalización de `MyFirstPipeline` Este de EE. UU. (Ohio) (us-east-2):

> **Note**
>
> El `codepipeline:ListPipelines` permiso solo es necesario si los usuarios de IAM necesitan acceder al CodePipeline panel para ver esta lista de canalizaciones. Si no se requiere acceso a la consola, puede omitir `codepipeline:ListPipelines`.

Utilización del editor de política de JSON para la creación de una política

1. Inicia sesión en Consola de administración de AWS y abre la consola de IAM en. https://console.aws.amazon.com/iam/
2. En el panel de navegación de la izquierda, elija Políticas.

   Si es la primera vez que elige Políticas, aparecerá la página Welcome to Managed Policies (Bienvenido a políticas administradas). Elija Comenzar.
3. En la parte superior de la página, seleccione Crear política.
4. En la sección Editor de políticas, seleccione la opción JSON.

5. Ingrese el siguiente documento de política JSON:

```
 {
    "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. Elija Siguiente.

> **Note**
>
> Puede alternar entre las opciones Visual y JSON del editor en todo momento. No obstante, si realiza cambios o selecciona Siguiente en la opción Visual del editor, es posible que IAM reestructure la política, con el fin de optimizarla para el editor visual.

Para obtener más información, consulte Reestructuración de política en la Guía del usuario de IAM.

7. En la página Revisar y crear, introduzca el Nombre de la política y la Descripción (opcional) para la política que está creando. Revise los Permisos definidos en esta política para ver los permisos que concede la política.
8. Elija Crear política para guardar la nueva política.

## Otorgue permisos de Amazon SNS a un rol de servicio CodePipeline

Si piensa utilizar Amazon SNS para publicar notificaciones en temas en los que haya que revisar las acciones de aprobación, la función de servicio que utilice en sus CodePipeline operaciones debe tener permiso para acceder a los recursos de Amazon SNS. Puede utilizar la consola de IAM para añadir este permiso al rol de servicio.

En la política siguiente, especifique la política de publicación con SNS. A la siguiente política, puede asignarle el nombre `SNSPublish`. Use la siguiente política asociándola a su rol de servicio.

**Important**

Asegúrese de haber iniciado sesión Consola de administración de AWS con la misma información de cuenta que utilizó. Empezar con CodePipeline

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "*"
        }
    ]
}
```

Utilización del editor de política de JSON para la creación de una política

1. Inicia sesión en Consola de administración de AWS y abre la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación de la izquierda, elija Políticas.

   Si es la primera vez que elige Políticas, aparecerá la página Welcome to Managed Policies (Bienvenido a políticas administradas). Elija Comenzar.
3. En la parte superior de la página, seleccione Crear política.
4. En la sección Editor de políticas, seleccione la opción JSON.
5. Introduzca o pegue un documento de política de JSON. Para obtener más información sobre el lenguaje de la política de IAM, consulte Referencia de políticas JSON de IAM.
6. Resuelva las advertencias de seguridad, errores o advertencias generales que se generen durante la validación de la política y luego elija Siguiente.

   > **Note**
   >
   > Puede alternar entre las opciones Visual y JSON del editor en todo momento. No obstante, si realiza cambios o selecciona Siguiente en la opción Visual del editor, es posible que IAM reestructure la política, con el fin de optimizarla para el editor visual. Para obtener más información, consulte Reestructuración de política en la Guía del usuario de IAM.
7. (Opcional) Al crear o editar una política en Consola de administración de AWS, puedes generar una plantilla de política JSON o YAML que puedes usar en CloudFormation las plantillas.

   Para ello, en el editor de políticas, selecciona Acciones y, a continuación, elige Generar CloudFormation plantilla. Para obtener más información CloudFormation, consulte la referencia a AWS Identity and Access Management los tipos de recursos en la Guía del AWS CloudFormation usuario.
8. Cuando haya terminado de agregar permisos a la política, seleccione Siguiente.
9. En la página Revisar y crear, introduzca el Nombre de la política y la Descripción (opcional) para la política que está creando. Revise los Permisos definidos en esta política para ver los permisos que concede la política.

10. (Opcional) Agregar metadatos a la política al adjuntar las etiquetas como pares de clave-valor. Para obtener más información sobre el uso de etiquetas en IAM, consulte las etiquetas de los AWS Identity and Access Management recursos en la Guía del usuario de IAM.
11. Elija Crear política para guardar la nueva política.

# Añada una acción de aprobación manual a una canalización en CodePipeline

Puedes agregar una acción de aprobación a una fase de una CodePipeline canalización en el punto en el que quieras que la canalización se detenga para que alguien pueda aprobar o rechazar la acción manualmente.

> **Note**
>
> Las acciones de aprobación no se pueden añadir a las etapas de origen. Las etapas de origen solo pueden contener acciones de origen.

Si quiere usar Amazon SNS para enviar notificaciones cuando una acción de aprobación esté lista para la revisión, primero debe completar los siguientes requisitos previos:

- Conceda permiso a su rol CodePipeline de servicio para acceder a los recursos de Amazon SNS. Para obtener información, consulte Otorgue permisos de Amazon SNS a un rol de servicio CodePipeline.
- Conceda permiso a uno o más identidades de IAM de la organización para actualizar el estado de una acción de aprobación. Para obtener información, consulte Conceda permisos de aprobación a un usuario de IAM en CodePipeline.

En este ejemplo, se crea una nueva etapa de aprobación y se añade una acción de aprobación manual a la etapa. También puede añadir una acción de aprobación manual a una etapa existente que contenga otras acciones.

## Añada una acción de aprobación manual a una CodePipeline canalización (consola)

Puede usar la CodePipeline consola para agregar una acción de aprobación a una CodePipeline canalización existente. Debe usar la AWS CLI si desea agregar acciones de aprobación al crear una nueva canalización.

1. Abra la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En Name, elija la canalización.
3. En la página de detalles de la canalización, elija Edit.
4. Si desea añadir una acción de aprobación a una etapa nueva, elija + Add stage (Añadir etapa) en el punto de la canalización donde desea añadir la solicitud de aprobación y escriba el nombre de la etapa. En la página Add stage (Añadir etapa) en Stage name (Nombre de etapa), escriba el nombre de la nueva etapa. Por ejemplo, añada una nueva etapa y asígnele el nombre `Manual_Approval`.

   Si desea añadir una acción de aprobación a una etapa existente, seleccione Edit stage (Editar etapa).
5. En la etapa en la que desea añadir la acción de aprobación, elija + Add action group (Añadir grupo de acciones).
6. En la página Edit action (Editar acción), haga lo siguiente:

   1. En Action name (Nombre de la acción), escriba un nombre que identifique la acción.
   2. En Action provider (Proveedor de acción), en Approval (Aprobación), seleccione Manual approval (Aprobación manual).
   3. (Opcional) En SNS topic ARN (ARN de tema de SNS), elija el nombre del tema que usará para enviar notificaciones para la acción de aprobación.
   4. (Opcional) En URL for review, escriba la URL de la página o aplicación que desea que el aprobador examine. Los aprobadores pueden obtener acceso a esta URL a través de un enlace que se incluye en la vista de la canalización en la consola.
   5. (Opcional) En Comments (Comentarios), escriba la información adicional que desee compartir con el revisor.
   6. Seleccione Save.

## Agregue una acción de aprobación manual a una CodePipeline canalización (CLI)

Puede usar la CLI para añadir una acción de aprobación a una canalización existente o al crear una canalización. Para ello, puede incluir una acción de aprobación, con el tipo de aprobación manual, en una etapa que esté creando o editando.

Para obtener más información sobre la creación y edición de canalizaciones, consulte Creación de una canalización, etapas y acciones y Editar una canalización en CodePipeline.

Para añadir una etapa a una canalización que incluye solo una acción de aprobación, se incluiría algo similar al ejemplo siguiente en el momento de crear o actualizar la canalización.

> **Note**
>
> La sección `configuration` es opcional. Esto es solo un fragmento, no toda la estructura del archivo. Para obtener más información, consulte CodePipeline referencia de estructura de tuberías.

```
{
    "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
        }
    ]
}
```

Si la acción de aprobación se encuentra en una etapa con otras acciones, la sección del archivo JSON que contiene la etapa puede tener un aspecto similar al del siguiente ejemplo.

> **Note**
>
> La sección `configuration` es opcional. Esto es solo un fragmento, no toda la estructura del archivo. Para obtener más información, consulte CodePipeline referencia de estructura de tuberías.

```
,
{
    "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
        }
    ]
}
```

## Apruebe o rechace una acción de aprobación en CodePipeline

Cuando una canalización incluye una acción de aprobación, la ejecución de la canalización se detiene en el punto donde se ha añadido la acción. La canalización no se reanudará a menos que alguien apruebe la acción manualmente. Si un aprobador rechaza la acción o si no se recibe ninguna respuesta de aprobación en un plazo de siete días desde la detención de la canalización para la acción de aprobación, el estado de la canalización pasa a ser "Failed".

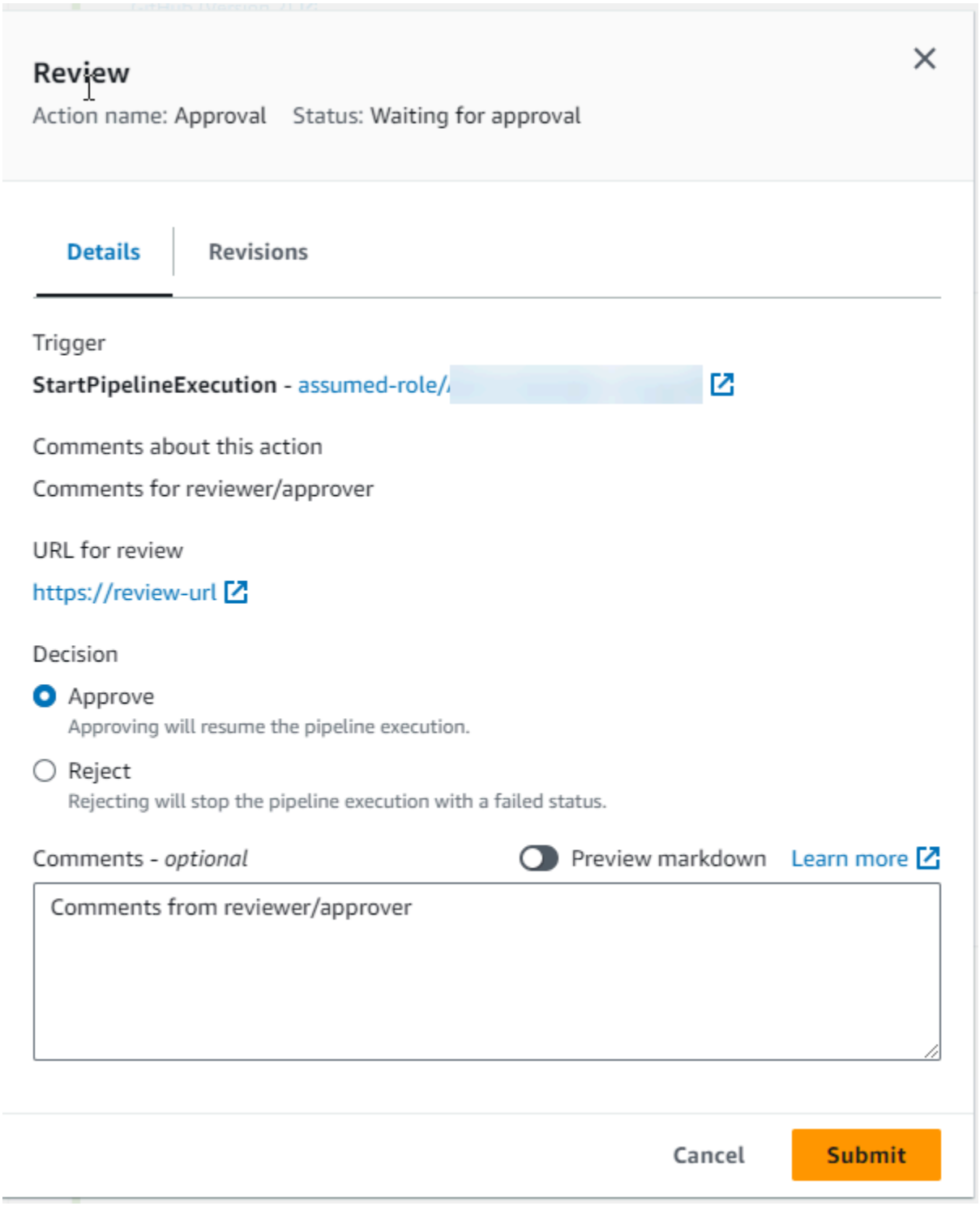
Si la persona que agregó la acción de aprobación a la canalización configuró las notificaciones, es posible que reciba un correo electrónico con la información de la canalización y el estado de la aprobación.

### Aprobación o rechazo de una acción de aprobación (consola)

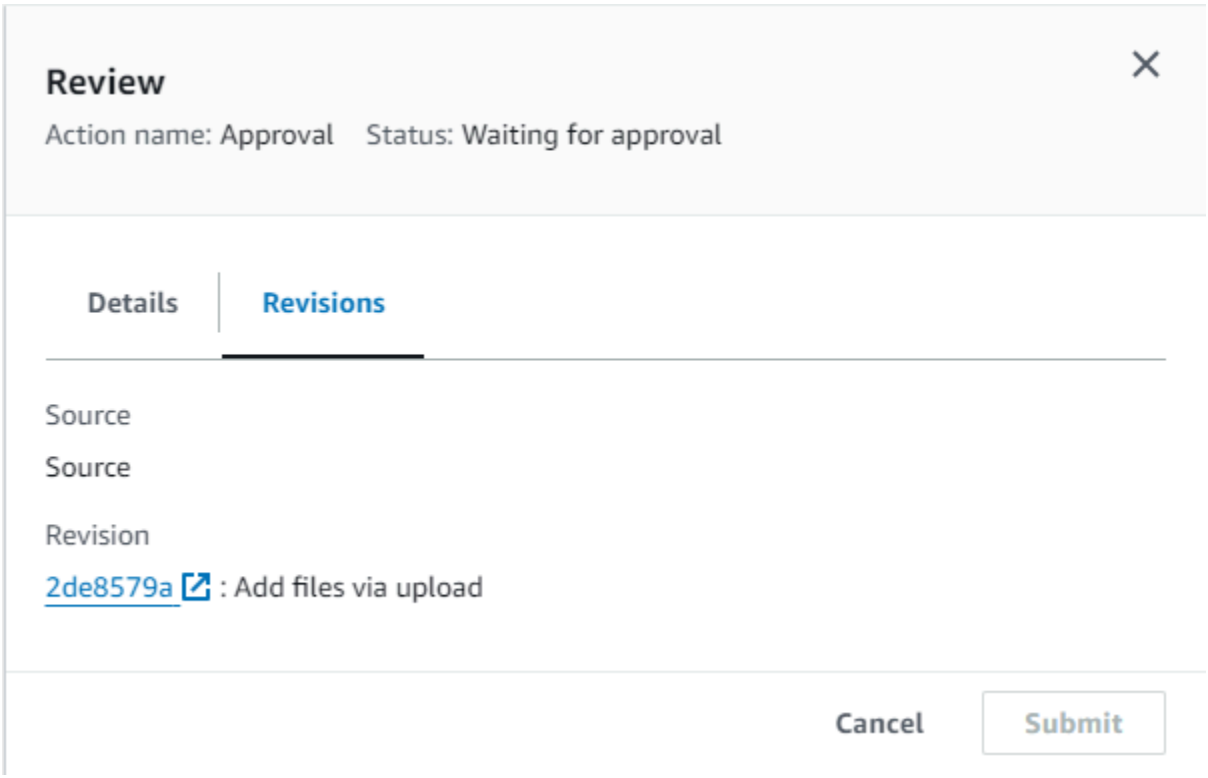
Si recibe una notificación que incluye un enlace directo a una acción de aprobación, elija el enlace Approve or reject (Aprobar o rechazar), inicie sesión en la consola si es necesario y, después, continúe con el paso 7. De lo contrario, siga estos pasos.

1. Abre la CodePipeline consola en https://console.aws.amazon.com/codepipeline/.
2. En la página All Pipelines elija el nombre de la canalización.
3. Localice la etapa con la acción de aprobación. Elija Revisar.

   Aparece el cuadro de diálogo Revisar. La pestaña Detalles muestra el contenido y los comentarios de la revisión.




La pestaña Revisiones muestra las revisiones de código fuente para la ejecución.




4. En la pestaña Detalles, consulte los comentarios y la URL, si los hubiera. El mensaje también muestra la URL de contenido para que la revise, en caso de que se hubiera incluido.
5. Si se ha proporcionado una URL, elija el enlace URL para revisión de la acción para abrir la página web de destino y revise el contenido.
6. En la ventana Revisar, escriba comentarios acerca de la revisión, p. ej., por qué aprueba o rechaza la acción, y después haga clic en el botón Aprobar o Rechazar.
7. Seleccione Enviar.

## Aprobación o rechazo de una solicitud de aprobación (CLI)

Para usar la CLI para responder a una acción de aprobación, primero debe usar el comando get-pipeline-state para recuperar el token asociado a la última ejecución de la acción de aprobación.

1. En una terminal (Linux, macOS o Unix) o símbolo del sistema (Windows), ejecute el comando get-pipeline-state en la canalización que incluye la acción de aprobación. Por ejemplo, para una canalización llamada *MyFirstPipeline*, escriba lo siguiente:

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

2. En respuesta al comando, ubique el valor `token`, que aparece en `latestExecution` en la sección `actionStates` de la acción de aprobación, tal como se muestra a continuación:

```
{
    "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. En un editor de texto sin formato, cree un archivo y añádale lo siguiente en formato JSON:

   - El nombre de la canalización que incluye la acción de aprobación.
   - El nombre de la etapa que incluye la acción de aprobación.
   - El nombre de la acción de aprobación.
   - El valor del token recopilado en el paso anterior.
   - Su respuesta a la acción, ya sea Approved (Aprobado) o Rejected (Rechazado). La respuesta debe ir con mayúscula inicial.
   - Sus comentarios de resumen.

   En el ejemplo de *MyFirstPipeline* anterior, el archivo sería así:

```
{
```

```
  "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. Guarde el archivo con un nombre como **`approvalstage-approved.json`**.
5. Ejecute el comando put-approval-result, especificando el nombre del archivo JSON de aprobación, similar al siguiente:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

```
aws codepipeline put-approval-result --cli-input-json file://approvalstage-approved.json
```

## Formato de datos JSON para las notificaciones de aprobación manual en CodePipeline

En el caso de las acciones de aprobación que usan notificaciones de Amazon SNS, se crean datos JSON sobre la acción y se publican en Amazon SNS cuando la canalización se detiene. La salida JSON se puede usar para enviar mensajes a colas de Amazon SQS o invocar funciones en AWS Lambda.

> **Note**
>
> Esta guía no trata sobre el modo de configurar las notificaciones en JSON. Para obtener más información, consulte Envío de mensajes de Amazon SNS a colas de Amazon SQS e Invocación de funciones de Lambda mediante notificaciones de Amazon SNS en la Guía para desarrolladores de Amazon SNS.

En el ejemplo siguiente se muestra la estructura de la salida JSON disponible con las aprobaciones de CodePipeline.

```
{
    "region": "us-east-2",
    "consoleLink": "https://console.aws.amazon.com/codepipeline/home?region=us-east-2#/
view/MyFirstPipeline",
    "approval": {
        "pipelineName": "MyFirstPipeline",
        "stageName": "MyApprovalStage",
        "actionName": "MyApprovalAction",
        "token": "1a2b3c4d-573f-4ea7-a67E-XAMPLETOKEN",
        "expires": "2016-07-07T20:22Z",
        "externalEntityLink": "http://example.com",
        "approvalReviewLink": "https://console.aws.amazon.com/codepipeline/
home?region=us-east-2#/view/MyFirstPipeline/MyApprovalStage/MyApprovalAction/
approve/1a2b3c4d-573f-4ea7-a67E-XAMPLETOKEN",
        "customData": "Review the latest changes and approve or reject within seven
 days."
    }
}
```

# Agregue una acción interregional en CodePipeline

AWS CodePipeline incluye una serie de acciones que ayudan a configurar los recursos de creación, prueba e implementación para el proceso de lanzamiento automatizado. Puedes añadir acciones a tu canalización que estén en una AWS región diferente a la tuya. Cuando un Servicio de AWS es el proveedor de una acción y este type/provider tipo de acción se encuentra en una AWS región diferente a la de tu canalización, se trata de una acción interregional.

> **Note**
>
> Cross-region Las acciones son compatibles y solo se pueden crear en AWS las regiones en las que están respaldadas. CodePipeline Para obtener una lista de las AWS regiones compatibles CodePipeline, consulteCuotas en AWS CodePipeline.

Puedes usar la consola o CloudFormation añadir acciones interregionales en las canalizaciones. AWS CLI

> **Note**
>
> CodePipeline Es posible que algunos tipos de acción solo estén disponibles en determinadas AWS regiones. Ten en cuenta también que puede haber AWS regiones en las que esté disponible un tipo de acción, pero no haya un AWS proveedor específico para ese tipo de acción.

Al crear o editar una canalización, debe tener un bucket de artefactos en la región de la canalización, así como un bucket de artefactos por cada región en la que tiene previsto ejecutar una acción. Para obtener más información sobre el parámetro `ArtifactStores`, consulte CodePipeline referencia de estructura de tuberías.

> **Note**
>
> CodePipeline gestiona la copia de artefactos de una AWS región a otras regiones al realizar acciones entre regiones.

Si utilizas la consola para crear una canalización o acciones entre regiones, los grupos de artefactos predeterminados se configuran en las regiones CodePipeline en las que tengas acciones. Cuando usas el AWS CLI SDK o un SDK para crear una canalización o acciones interregionales, proporcionas el depósito de artefactos para cada región en la que tengas acciones. CloudFormation

> **Note**
>
> Debes crear el depósito de artefactos y la clave de cifrado en la misma AWS región que la acción interregional y en la misma cuenta que tu canalización.

No puede crear acciones entre regiones para los siguientes tipos de acciones:

- Acciones de código fuente
- Third-party acciones
- Acciones personalizadas

> **Note**
>
> Al utilizar la acción de invocación de Lambda entre regiones CodePipeline, el estado de la ejecución de Lambda que utiliza el PutJobSuccessResult y PutJobFailureResult debe enviarse a la AWS región en la que esté presente la función Lambda y no a la región en la que exista. CodePipeline

Cuando una canalización incluye una acción interregional como parte de una etapa, solo CodePipeline replica los artefactos de entrada de la acción interregional desde la región de canalización a la región de la acción.

> **Note**
>
> La región del proceso y la región en la que se mantienen tus recursos de detección de cambios en los CloudWatch eventos siguen siendo las mismas. La región en la que se aloja la canalización no cambia.

## Gestiona las acciones interregionales en una canalización (consola)

Puede usar la CodePipeline consola para agregar una acción interregional a una canalización existente. Para crear una canalización nueva con acciones entre regiones mediante el asistente de creación de canalizaciones, consulte Creación de una canalización personalizada (consola).

En la consola, puede crear una acción interregional en una fase de tramitación seleccionando el proveedor de la acción y el campo Región, que muestra los recursos que ha creado en esa región para ese proveedor. Al añadir una acción interregional, CodePipeline utiliza un depósito de artefactos independiente en la región de la acción. Para obtener más información sobre los buckets de artefactos entre regiones, consulte CodePipeline referencia de estructura de tuberías.

### Añade una acción interregional a una fase de tramitación (consola)

Utilice la consola para añadir una acción entre regiones a una canalización.

> **Note**
>
> Si la canalización se está ejecutando al guardar los cambios, esa ejecución no se completa.

Para añadir una acción entre regiones

1. Inicie sesión en la consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Seleccione la canalización y, a continuación, elija Edit (Editar).
3. En la parte inferior del diagrama, elija + Add stage (Añadir etapa) si va a añadir una etapa nueva o elija Edit stage (Editar etapa) si desea añadir la acción a una etapa existente.
4. En Edit: <Stage> (Editar: <etapa>), elija + Add action group (Añadir grupo de acciones) para añadir una acción en serie. O elija + Add action (Añadir acción) para añadir una acción en paralelo.
5. En la página Edit action (Editar acción):
   a. En Action name (Nombre de la acción), escriba un nombre para la acción entre regiones.
   b. En Action provider (Proveedor de acción), elija el proveedor de la acción.
   c. En  Región, elige la AWS región en la que creaste o planeas crear el recurso para la acción. Al seleccionar la región, los recursos disponibles para esa región se muestran para su selección. El  campo  Región designa dónde se crean los AWS recursos para este tipo de acción y tipo de proveedor. Este campo solo se muestra en el caso de las acciones en las que el proveedor de la acción es un Servicio de AWS. El campo Región se establece de forma predeterminada en la misma Región de AWS que la canalización.
   d. En Input artifacts (Artefactos de entrada), elija la entrada apropiada de la etapa anterior. Por ejemplo, si la etapa anterior es una etapa de origen, elija  SourceArtifact.
   e. Rellene todos los campos obligatorios para el proveedor de la acción que está configurando.
   f. En Output artifacts (Artefactos de salida), elija la salida apropiada para la etapa siguiente. Por ejemplo, si la siguiente etapa es una etapa de despliegue, elija  BuildArtifact.
   g. Seleccione Save.
6. En Edit: <Stage> (Editar: <etapa>), elija Done (Listo).
7. Seleccione Save.

## Edite una acción interregional en una fase de canalización (consola)

Utilice la consola para editar una acción entre regiones existente en una canalización.

> **Note**
>
> Si la canalización se está ejecutando al guardar los cambios, esa ejecución no se completa.

Para editar una acción entre regiones

1. Inicie sesión en la consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Seleccione la canalización y, a continuación, elija Edit (Editar).
3. Elija Edit stage (Editar etapa).
4. En Edit: <Stage> (Editar: <etapa>), elija el icono para editar una acción existente.
5. En la página Edit action (Editar acción), realice los cambios oportunos en los campos.
6. En Edit: <Stage> (Editar: <etapa>), elija Done (Listo).
7. Seleccione Save.

## Eliminar una acción interregional de una fase de canalización (consola)

Utilice la consola para eliminar una acción entre regiones existente de una canalización.

> **Note**
>
> Si la canalización se está ejecutando al guardar los cambios, esa ejecución no se completa.

Para eliminar una acción entre regiones

1. Inicie sesión en la consola en http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Seleccione la canalización y, a continuación, elija Edit (Editar).
3. Elija Edit stage (Editar etapa).
4. En Edit: <Stage> (Editar: <etapa>), elija el icono para eliminar una acción existente.
5. En Edit: <Stage> (Editar: <etapa>), elija Done (Listo).
6. Seleccione Save.

# Agregar una acción interregional a una canalización (CLI)

Puede utilizarla AWS CLI para agregar una acción interregional a una canalización existente.

Para crear una acción interregional en una fase de canalización con AWS CLI, debes añadir la acción de configuración junto con un campo opcional`region`. También debes haber creado ya un depósito de artefactos en la región de la acción. En lugar de proporcionar el parámetro `artifactStore` de la canalización de región única, utilice el parámetro `artifactStores` para incluir un listado de los buckets de artefactos de cada región.

> **Note**
>
> En este tutorial y en sus ejemplos, *`RegionA`* se muestra la región en la que se creó la canalización. Tiene acceso al bucket de *`RegionA`* Amazon S3 que se utiliza para almacenar los artefactos de la canalización y al rol de servicio que utiliza. CodePipeline *`RegionB`*es la región en la que se crean la CodeDeploy aplicación, el grupo de implementación y el rol de servicio CodeDeploy que utilizan.

## Requisitos previos

Tiene que haber creado lo siguiente:

- Una canalización en*`RegionA`*.
- Ha llegado un cubo de artefactos de Amazon S3. *`RegionB`*
- Los recursos para su acción, como la CodeDeploy aplicación y el grupo de implementación para una acción de implementación entre regiones, están en. *`RegionB`*

## Agregue una acción interregional a una canalización (CLI)

Utilice la AWS CLI para agregar una acción interregional a una canalización.

Para añadir una acción entre regiones

1. En el caso de una canalización*`RegionA`*, ejecuta el get-pipeline comando para copiar la estructura de la canalización en un archivo JSON. Por ejemplo, para una canalización denominada `MyFirstPipeline`, escriba el siguiente comando:

```
aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
```

Este comando no devuelve nada, pero el archivo creado debería aparecer en el directorio en el que se ejecutó el comando.

2. Añada el campo `region` para añadir una nueva etapa con la acción entre regiones que incluya la región y los recursos de la acción. En el siguiente ejemplo de JSON, se agrega una etapa de implementación con una acción de implementación interregional en la que el proveedor se encuentra CodeDeploy en una nueva regiónus-east-1.

```
{
                "name": "Deploy",
                "actions": [
                    {
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "name": "Deploy",
                        "region": "RegionB",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "name",
                            "DeploymentGroupName": "name"
                        },
                        "runOrder": 1
                    }
```

3. En la estructura de la canalización, elimine el campo `artifactStore` y añada el mapa `artifactStores` de su nueva acción entre regiones. El mapeo debe incluir una entrada para cada AWS región en la que realices acciones. Para cada entrada del mapa, los recursos deben estar en la AWS región correspondiente. En el ejemplo siguiente, `ID-A` es el ID de la clave de cifrado de *RegionA* y `ID-B` es el ID de la clave de cifrado de*RegionB*.

```
"artifactStores":{
   "RegionA":{
      "encryptionKey":{
         "id":"ID-A",
         "type":"KMS"
      },
      "location":"Location1",
      "type":"S3"
   },
   "RegionB":{
      "encryptionKey":{
         "id":"ID-B",
         "type":"KMS"
      },
      "location":"Location2",
      "type":"S3"
   }
}
```

En el ejemplo siguiente de JSON se muestra el bucket de us-west-2 como `my-storage-bucket` y añade el nuevo bucket de us-east-1 denominado `my-storage-bucket-us-east-1`.

```
        "artifactStores": {
            "us-west-2": {
                "type": "S3",
                "location": "my-storage-bucket"
            },
            "us-east-1": {
                "type": "S3",
                "location": "my-storage-bucket-us-east-1"
            }
        },
```

4. Si está trabajando con la estructura de la canalización recuperada mediante el comando get-pipeline, elimine las líneas `metadata` del archivo JSON. De lo contrario, el comando update-pipeline no puede utilizarlo. Elimine las líneas `"metadata": { }` y los campos `"updated"`, `"created"` y `"pipelineARN"`.

   Por ejemplo, quite las siguientes líneas de la estructura:

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

Guarde el archivo.

5. Para aplicar los cambios, ejecute el comando update-pipeline especificando el archivo JSON:

> **Important**
>
> Asegúrese de incluir `file://` antes del nombre de archivo. Es obligatorio en este comando.

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

Este comando devuelve la estructura completa de la canalización editada. El resultado es similar al siguiente.

```
{
    "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**
>
> El comando update-pipeline detiene la canalización. Si se está ejecutando una revisión en la canalización al ejecutar el comando update-pipeline, dicha ejecución se detiene. Debe iniciar manualmente la canalización para ejecutar dicha revisión en la canalización actualizada. Utilice el comando **`start-pipeline-execution`** para iniciar manualmente la canalización.

6. Tras actualizar la canalización, la acción interregional se muestra en la consola.




## Añade una acción interregional a una canalización (CloudFormation)

Se puede usar CloudFormation para agregar una acción interregional a una canalización existente.

Para añadir una acción interregional con CloudFormation

1. Añada el parámetro `Region` al recurso `ActionDeclaration` en la plantilla, tal y como se muestra en este ejemplo:

```
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. En `Mappings`, añada el mapa de región como se muestra en este ejemplo para una asignación denominada `SecondRegionMap` que asigne valores a las claves `RegionA` y `RegionB`. En el recurso `Pipeline`, en el campo `artifactStore`, añada el mapa `artifactStores` para la nueva acción entre regiones tal y como se indica a continuación:

```
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
```

En el siguiente ejemplo de YAML, se muestra el *RegionA* bucket como `us-west-2` y se agrega el nuevo *RegionB* bucket: `eu-central-1`

```
Mappings:
  SecondRegionMap:
    us-west-2:
      SecondRegion: "eu-central-1"
    eu-central-1:
      SecondRegion: "us-west-2"

...

          Properties:
            ArtifactStores:
              -
                Region: eu-central-1
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-eu-central-1
              -
                Region: us-west-2
                ArtifactStore:
                  Type: "S3"
                  Location: test-cross-region-artifact-store-bucket-us-west-2
```

3. Guarde la plantilla actualizada en el equipo local y luego abra la consola de CloudFormation .
4. Seleccione la pila y luego elija Create Change Set for Current Stack (Crear conjuntos de cambios para la pila actual).
5. Cargue la plantilla y, a continuación, consulte los cambios indicados en CloudFormation. Estos son los cambios que se realizan en la pila. Debería ver los nuevos recursos en la lista.

6. Elija Ejecutar.

# Trabajar con variables

Algunas acciones CodePipeline generan variables. Para utilizar variables:

- Asigne un espacio de nombres a una acción para que las variables que produce estén disponibles para una configuración de acción descendente.
- Configure la acción descendente para que consuma las variables generadas por la acción.

  Puede ver los detalles de cada ejecución de acción para ver los valores de cada variable de salida generados por la acción en tiempo de ejecución.

Para ver step-by-step ejemplos del uso de variables:

- Para ver un tutorial sobre una acción de Lambda que utiliza variables de una acción anterior (CodeCommit) y genera variables de salida, consulte. Tutorial: Uso de variables con acciones de invocación de Lambda
- Para ver un tutorial con una CloudFormation acción que hace referencia a las variables de salida de la pila de una CloudFormation acción anterior, consulte. Tutorial: Cree una canalización que utilice variables de AWS CloudFormation acciones de despliegue
- Para ver un ejemplo de acción de aprobación manual con un texto de mensaje que haga referencia a las variables de salida que se resuelven en el ID de CodeCommit confirmación y el mensaje de confirmación, consulteEjemplo: Usar variables en aprobaciones manuales.
- Para ver un ejemplo de CodeBuild acción con una variable de entorno que se resuelve en el nombre de la GitHub rama, consulteEjemplo: usa una BranchName variable con variables de CodeBuild entorno.
- CodeBuild las acciones producen como variables todas las variables de entorno que se exportaron como parte de la compilación. Para obtener más información, consulte CodeBuild variables de salida de la acción. Para obtener una lista de las variables de entorno que puede utilizar CodeBuild, consulte Variables de entorno en entornos de compilación en la Guía del AWS CodeBuild usuario.

Temas

- Configuración de acciones para variables

- Ver variables de salida
- Ejemplo: Usar variables en aprobaciones manuales
- Ejemplo: usa una BranchName variable con variables de CodeBuild entorno

## Configuración de acciones para variables

Cuando agrega una acción a la canalización, puede asignarle un espacio de nombres y configurarla para que consuma variables de acciones anteriores.

### Configuración de acciones con variables (consola)

En este ejemplo, se crea una canalización con una acción de CodeCommit origen y una acción de CodeBuild compilación. La CodeBuild acción está configurada para consumir las variables producidas por la CodeCommit acción.

Si no se especifica el espacio de nombres, las variables no están disponibles para referencia en la configuración de la acción. Cuando se utiliza la consola para crear una canalización, el espacio de nombres de cada acción se genera automáticamente.

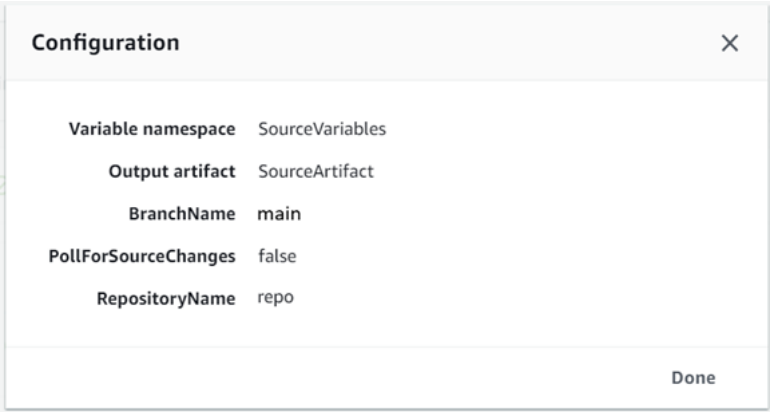
Para crear una canalización con variables

1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Elija Crear canalización. Escriba un nombre para la canalización y, a continuación, elija Next (Siguiente).
3. En Fuente, en Proveedor, elija CodeCommit. Elija el CodeCommit repositorio y la rama para la acción de origen y, a continuación, elija Siguiente.
4. En Build, en Provider, elija CodeBuild. Elija un nombre de proyecto de CodeBuild construcción existente o elija Crear proyecto. En Crear proyecto de construcción, cree un proyecto de construcción y, a continuación, elija Volver a CodePipeline.

   En Environment variables (Variables de entorno), elija Add environment variables (Añadir variables de entorno). Por ejemplo, introduzca el ID de ejecución con la sintaxis de la variable `#{codepipeline.PipelineExecutionId}` y el ID de confirmación con la sintaxis de la variable `#{SourceVariables.CommitId}`.

> **Note**
>
> Puede introducir sintaxis de variable en cualquier campo de configuración de acción del asistente.

5. Seleccione Crear.
6. Una vez creada la canalización, puede ver el espacio de nombres creado por el asistente. En la canalización, elija el icono de la etapa para el que desea ver el espacio de nombres. En este ejemplo, se muestra el espacio de nombres generado automáticamente por la acción de origen, `SourceVariables`.




Para editar el espacio de nombres de una acción existente

1. Inicie sesión Consola de administración de AWS y abra la CodePipeline consola en http:// console.aws.amazon. com/codesuite/codepipeline/home.
2. Elija la canalización que desea editar y, a continuación, elija Edit (Editar). En la etapa de código fuente, elija Edit stage (Editar etapa). Añade la CodeCommit acción.
3. En Edit action (Editar acción), consulte el campo Variable namespace (Espacio de nombres de variable). Si la acción existente se creó previamente o sin utilizar el asistente, debe agregar un espacio de nombres. En Variable namespace (Espacio de nombres de variable), introduzca un nombre de espacio de nombres y, a continuación, elija Save (Guardar).

Para ver variables de salida

1. Inicie sesión en la CodePipeline consola Consola de administración de AWS y ábrala en http:// console.aws.amazon. com/codesuite/codepipeline/home.

2. Una vez creada la canalización y ejecutada correctamente, puede ver las variables en la página Action execution details (Detalles de ejecución de acciones). Para obtener información, consulte Ver variables (consola).

## Configuración de acciones para variables (CLI)

Cuando utilizas el create-pipeline comando para crear una canalización o el update-pipeline comando para editar una canalización, reference/use puedes modificar la configuración de una acción.

Si no se especifica el espacio de nombres, las variables producidas por la acción no están disponibles para hacer referencia en ninguna configuración de acción descendente.

Para configurar una acción con un espacio de nombres

1. Siga los pasos de Creación de una canalización, etapas y acciones para crear una canalización mediante la CLI. Inicie un archivo de entrada para proporcionar el comando create-pipeline con el parámetro `--cli-input-json`. En la estructura de la canalización, agregue el `namespace` parámetro y especifique un nombre, como `SourceVariables`.

```
. . .
{
          "inputArtifacts": [],
          "name": "Source",
          "region": "us-west-2",
          "namespace": "SourceVariables",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeCommit"
          },
          "outputArtifacts": [

. . .
```

2. Guarde el archivo con un nombre como **`MyPipeline.json`**.
3. En un terminal (Linux, macOS o Unix) o símbolo del sistema (Windows), ejecute el comando create-pipeline y cree la canalización.

   Llame el archivo que creó al ejecutar el comando create-pipeline. Por ejemplo:

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

Para configurar acciones descendentes para consumir variables

1. Edite un archivo de entrada para proporcionar el comando update-pipeline con el parámetro `--cli-input-json`. En la acción descendente, agregue la variable a la configuración de esa acción. Una variable se compone de un espacio de nombres y clave, separados por un punto. Por ejemplo, para agregar variables para el ID de ejecución de canalización y el ID de confirmación de origen, especifique el espacio de nombres `codepipeline` de la variable `#{codepipeline.PipelineExecutionId}`. Especifique el espacio de nombres `SourceVariables` para la variable `#{SourceVariables.CommitId}`.

```
{
    "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. Guarde el archivo con un nombre como **`MyPipeline.json`**.
3. En un terminal (Linux, macOS o Unix) o símbolo del sistema (Windows), ejecute el comando create-pipeline y cree la canalización.

   Llame el archivo que creó al ejecutar el comando create-pipeline. Por ejemplo:

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

## Ver variables de salida

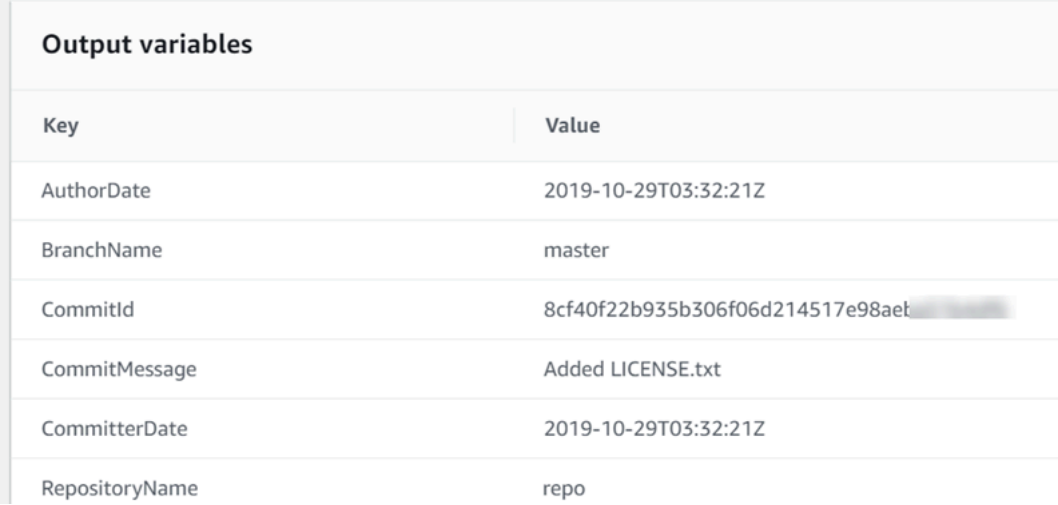
Puede ver los detalles de ejecución de la acción para ver las variables de esa acción, específicas de cada ejecución.

### Ver variables (consola)

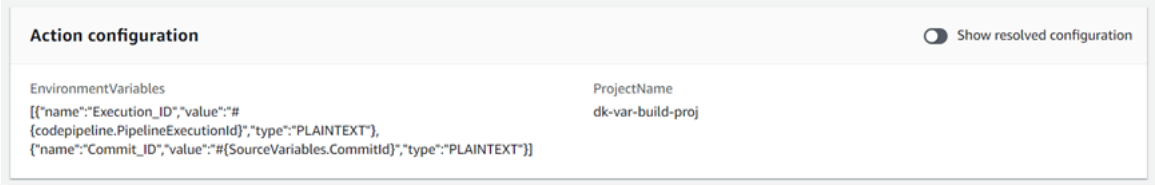
Puede utilizar la consola para ver las variables de una acción.

1. Inicia sesión en la CodePipeline consola Consola de administración de AWS y ábrela en http://console.aws.amazon. com/codesuite/codepipeline/home.

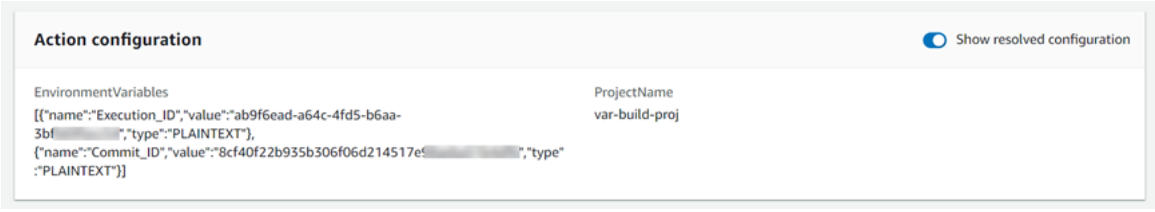
   Se muestran los nombres de todas las canalizaciones asociadas a tu AWS cuenta.
2. En Name, elija el nombre de la canalización.
3. Seleccione Ver historial.
4. Después de que la canalización se ejecute correctamente, puede ver las variables producidas por la acción de origen. Seleccione Ver historial. Selecciona Fuente en la lista de acciones de la ejecución de la canalización para ver los detalles de la ejecución de la CodeCommit acción. En la pantalla de detalles de la acción, vea las variables en Output variables (Variables de salida).




5. Después de que la canalización se ejecute correctamente, puede ver las variables consumidas por la acción de compilación. Seleccione Ver historial. En la lista de acciones de la ejecución de la canalización, seleccione Crear para ver los detalles de la ejecución de la CodeBuild acción. En la página de detalles de la acción, vea las variables en Action configuration (Configuración de acción). Se muestra el espacio de nombres generado automáticamente.




De forma predeterminada, Action configuration (Configuración de acción) muestra la sintaxis de la variable. Puede elegir Show resolved configuration (Mostrar configuración resuelta) para alternar la lista y mostrar los valores que se produjeron durante la ejecución de la acción.




## Ver variables (CLI)

Puede utilizar el comando list-action-executions para ver las variables de una acción.

1. Utilice el siguiente comando :

```
aws codepipeline list-action-executions
```

La salida muestra el parámetro `outputVariables` como se muestra aquí.

```
"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. Utilice el siguiente comando :

```
aws codepipeline get-pipeline --name <pipeline-name>
```

En la configuración de la CodeBuild acción, puedes ver las variables:

```
{
    "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
        }
    ]
},
```

## Ejemplo: Usar variables en aprobaciones manuales

Cuando especifica un espacio de nombres para una acción y esa acción produce variables de salida, puede añadir una aprobación manual que muestre variables en el mensaje de aprobación. En este ejemplo se muestra cómo añadir sintaxis de variables a un mensaje de aprobación manual.

1. Inicie sesión Consola de administración de AWS y abra la CodePipeline consola en http://console.aws.amazon. com/codesuite/codepipeline/home.

   Se muestran los nombres de todas las canalizaciones asociadas a tu AWS cuenta. Elija la canalización la que desea añadir la aprobación.
2. Para editar la canalización, elija Editar. Añada una aprobación manual después de la acción de código fuente. En Action name (Nombre de la acción), escriba el nombre de la acción de aprobación.
3. En Action provider (Proveedor de acciones), elija Manual approval (Aprobación manual).
4. En la URL para su revisión, añade la sintaxis variable correspondiente `CommitId` a tu CodeCommit URL. Asegúrese de utilizar el espacio de nombres asignado a la acción de código fuente. Por ejemplo, la sintaxis de variables para una acción de CodeCommit con el espacio de nombres predeterminado `SourceVariables` es `#{SourceVariables.CommitId}`.

   En Comentarios, en `CommitMessage`, introduzca el mensaje de confirmación:

   ```
   Please approve this change. Commit message: #{SourceVariables.CommitMessage}
   ```
5. Una vez que la canalización se ejecute correctamente, puede ver los valores de las variables en el mensaje de aprobación.

# Ejemplo: usa una BranchName variable con variables de CodeBuild entorno

Cuando agregas una CodeBuild acción a tu canalización, puedes usar variables de CodeBuild entorno para hacer referencia a una variable de `BranchName` salida de una acción fuente anterior. Con una variable de salida de una acción CodePipeline, puedes crear tus propias variables de CodeBuild entorno para usarlas en tus comandos de compilación.

En este ejemplo, se muestra cómo añadir la sintaxis de una variable de salida de una acción de GitHub origen a una variable de CodeBuild entorno. La sintaxis de la variable de salida de este ejemplo representa la variable de salida de la acción de GitHub origen para`BranchName`. Una vez que la acción se ejecute correctamente, la variable resuelve mostrar el nombre de la GitHub rama.

1. Inicie sesión Consola de administración de AWS y abra la CodePipeline consola en http:// console.aws.amazon. com/codesuite/codepipeline/home.

   Se muestran los nombres de todas las canalizaciones asociadas a tu AWS cuenta. Elija la canalización la que desea añadir la aprobación.
2. Para editar la canalización, elija Editar. En la etapa que contiene tu CodeBuild acción, selecciona Editar etapa.
3. Elige el icono para editar la CodeBuild acción.
4. En la página Editar acción, en Variables de entorno, introduzca lo siguiente:

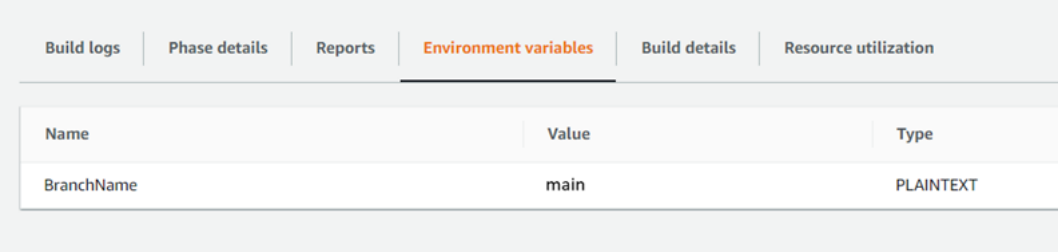
   - En Nombre, introduzca un nombre para su variable de entorno.
   - En Valor, introduzca la sintaxis de la variable de salida de la canalización, que incluye el espacio de nombres asignado a la acción de origen. Por ejemplo, la sintaxis de la variable de salida para una GitHub acción con el espacio de nombres `SourceVariables` predeterminado es. `#{SourceVariables.BranchName}`
   - En Tipo, seleccione Texto sin formato.
5. Una vez que la canalización se ejecute correctamente, podrá ver cómo la variable de salida resuelta es el valor de la variable de entorno. Seleccione una de las siguientes opciones:

   - CodePipeline consola: elige tu canalización y, a continuación, selecciona Historial. Elija la ejecución de la canalización más reciente.
     - En Línea temporal, seleccione el selector de origen. Esta es la acción de origen que genera las variables GitHub de salida. Elija Ver detalles de ejecución. En Variables de salida, consulte la lista de variables de salida generadas por esta acción.

- En Línea temporal, seleccione el selector de Compilar. Esta es la acción de compilación que especifica las variables de CodeBuild entorno del proyecto de compilación. Elija Ver detalles de ejecución. En Configuración de acciones, consulta las variables de CodeBuild entorno. Seleccione Mostrar configuración resuelta. El valor de la variable de entorno es la variable `BranchName` de salida resuelta de la acción de GitHub origen. En este ejemplo, el valor resuelto es `main`.

  Para obtener más información, consulte Ver variables (consola).

- CodeBuild consola: elige tu proyecto de compilación y elige el enlace para la ejecución de la compilación. En Variables de entorno, la variable de salida resuelta es el valor de la variable de CodeBuild entorno. En este ejemplo, el nombre de la variable de entorno es `BranchName` y el valor es la variable de `BranchName` salida resuelta de la acción de GitHub origen. En este ejemplo, el valor resuelto es `main`.



# Trabajando con transiciones de escenario en CodePipeline

Las transiciones son enlaces entre etapas de canalización que pueden habilitarse o deshabilitarse. De forma predeterminada, están habilitadas. Cuando se vuelve a habilitar una transición deshabilitada, la última revisión se ejecuta en las demás etapas de la canalización, salvo que hayan transcurrido más de 30 días. La ejecución de la canalización no continúa para las transiciones que hayan estado deshabilitadas durante más de 30 días, salvo que se de detecte un cambio o que se ejecute de nuevo la canalización manualmente.

Puede usar la AWS CodePipeline consola o la AWS CLI para deshabilitar o habilitar las transiciones entre las etapas de una canalización.

> **Note**
>
> Puede utilizar una acción de aprobación para detener la ejecución de una canalización hasta que se apruebe su continuación manualmente. Para obtener más información, consulte Incorporación de una acción de aprobación manual a una etapa.

Temas


- Activar o desactivar transiciones (consola)
- Activar o desactivar transiciones (CLI)

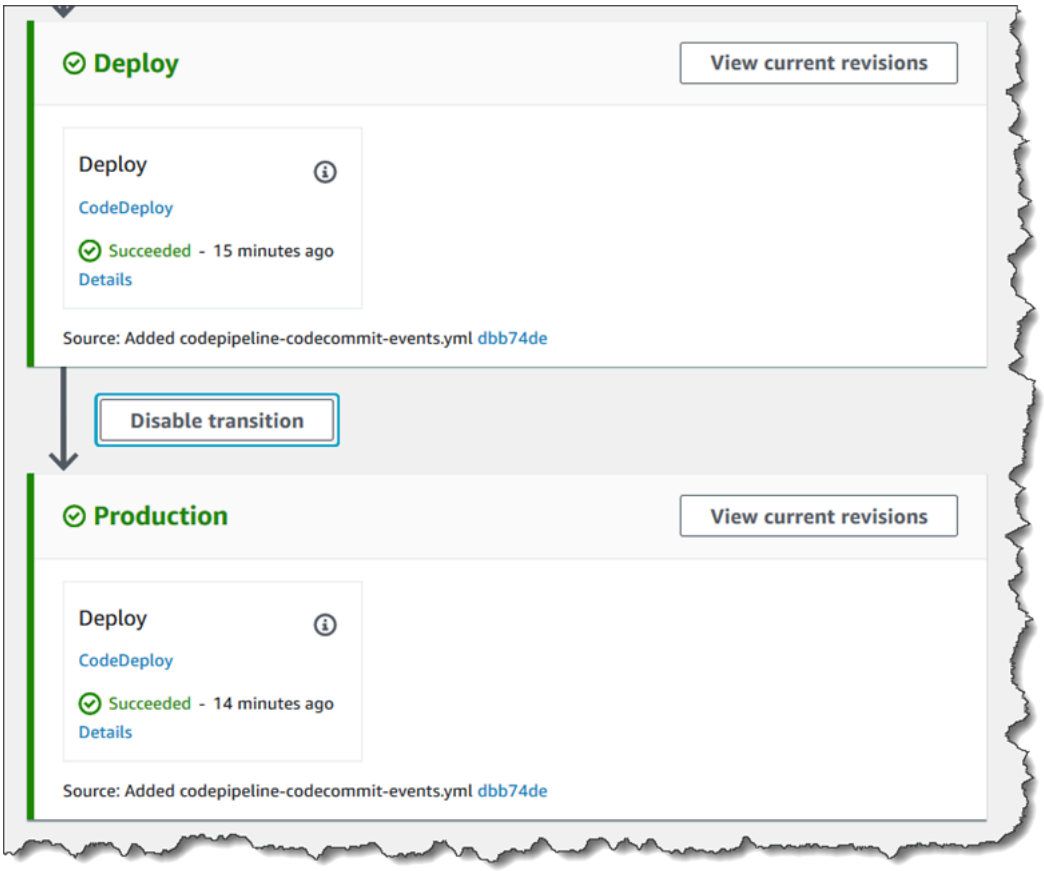

## Activar o desactivar transiciones (consola)

Para deshabilitar o habilitar transiciones en una canalización

1. Inicia sesión en Consola de administración de AWS y abre la CodePipeline consola en http://console.aws.amazon.com/codesuite/codepipeline/home.

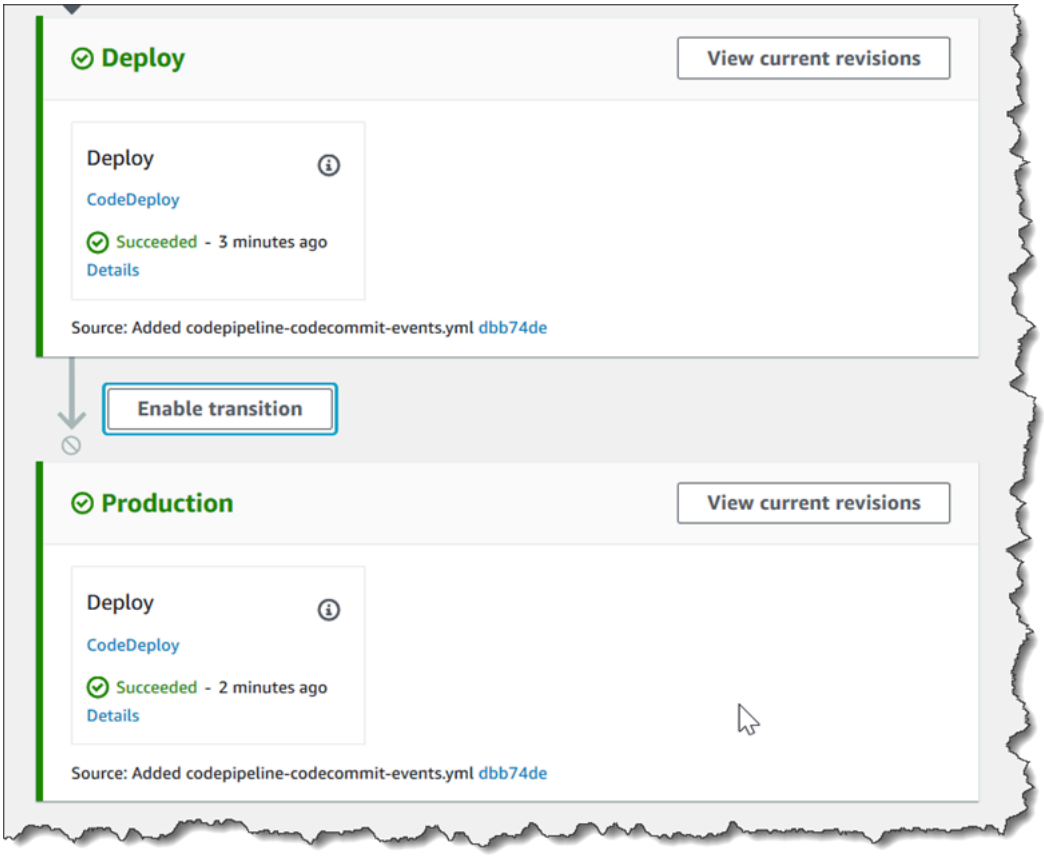
   Se muestran los nombres de todas las canalizaciones asociadas con su cuenta de AWS .
2. En Name, elija el nombre de la canalización para la que desea habilitar o deshabilitar transiciones. Esto abre una vista detallada de la canalización, que incluye las transiciones entre las etapas de la canalización.
3. Busque la flecha después de la última etapa que desea ejecutar y, a continuación, elija el botón situado junto a ella. Por ejemplo, en la siguiente canalización, si quiere que se ejecuten

las acciones de la etapa Staging (Ensayo), pero no las de la etapa Production (Producción), seleccione el botón Disable transition (Deshabilitar transición) entre estas dos etapas:




4. En el cuadro de diálogo Disable transition (Deshabilitar transición), escriba un motivo para deshabilitar la transición y después elija Disable (Deshabilitar).

   El botón cambia para mostrar que las transiciones entre la etapa anterior a la flecha y la etapa posterior a la flecha se han deshabilitado. Las revisiones que ya se ejecutan en las etapas y que se incluyen después de la transición deshabilitada continúan por la canalización, pero las revisiones subsiguientes no.




5. Haga clic en el botón Enable transition (Habilitar transición) situado junto a la flecha. En el cuadro de diálogo Enable transition, elija Enable. La canalización habilita inmediatamente la transición entre las dos etapas. Si se han ejecutado revisiones en las etapas anteriores una vez deshabilitada la transición, la canalización comienza a ejecutar la última revisión en las etapas posteriores a la transición deshabilitada anteriormente. La canalización ejecuta la revisión en todas las etapas restantes de la canalización.

> **Note**
>
> Es posible que los cambios tarden unos segundos en aparecer en la CodePipeline consola después de habilitar la transición.

## Activar o desactivar transiciones (CLI)

Para deshabilitar una transición entre etapas mediante el AWS CLI, ejecute el disable-stage-transition comando. Para habilitar una transición deshabilitada ejecute el comando enable-stage-transition.

Para deshabilitar una transición

1. Abra una terminal (Linux, macOS o Unix) o una línea de comandos (Windows) y utilícela AWS CLI para ejecutar el comando disable-stage-transition, especificando el nombre de la canalización, el nombre de la etapa en la que desea deshabilitar las transiciones, el tipo de transición y el motivo por el que se deshabilitan las transiciones a esa etapa. A diferencia del uso de la consola, debe especificar si está deshabilitando las transiciones hacia la etapa (entrante) o las transiciones fuera de la etapa una vez completadas todas las acciones (salida).

   Por ejemplo, para deshabilitar la transición a una etapa denominada *Staging* en una canalización denominada*MyFirstPipeline*, debe escribir un comando similar al siguiente:

   ```
   aws codepipeline disable-stage-transition --pipeline-name MyFirstPipeline --stage-
   name Staging --transition-type Inbound --reason "My Reason"
   ```

   El comando no devuelve nada.
2. Para verificar que la transición que se ha desactivado, visualice la canalización en la consola de CodePipeline o ejecute el comando get-pipeline-state. Para obtener más información, consulte Ver canalizaciones (consola) y Visualización de los detalles y el historial de la canalización (CLI).

Para habilitar una transición

1. Abra una terminal (Linux, macOS o Unix) o una línea de comandos (Windows) y utilice la AWS CLI para ejecutar el comando enable-stage-transition, especificando el nombre de la canalización, el nombre de la etapa en la que desea habilitar las transiciones y el tipo de transición.

   Por ejemplo, para habilitar la transición a una etapa denominada *Staging* en una canalización denominada*MyFirstPipeline*, debes escribir un comando similar al siguiente:

   ```
   aws codepipeline enable-stage-transition --pipeline-name MyFirstPipeline --stage-
   name Staging  --transition-type Inbound
   ```

   El comando no devuelve nada.
2. Para comprobar que la transición está habilitada, consulta la canalización en la CodePipeline consola o ejecuta el get-pipeline-state comando. Para obtener más información, consulte Ver canalizaciones (consola) y Visualización de los detalles y el historial de la canalización (CLI).

# Monitorear canalizaciones

La monitorización es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de AWS CodePipeline. Debe recopilar datos de monitoreo de todas las partes de su solución de AWS para que le resulte más sencillo depurar cualquier error que se produzca en distintas partes del código, en caso de que ocurra. Antes de empezar la monitorización, debe crear un plan de monitorización que responda a las siguientes preguntas:

- ¿Cuáles son los objetivos de la monitorización?
- ¿Qué recursos va a monitorizar?
- ¿Con qué frecuencia va a monitorizar estos recursos?
- ¿Qué herramientas de monitorización están disponibles?
- ¿Quién se encargará de realizar las tareas de supervisión?
- ¿Quién debería recibir una notificación si surgen problemas?

Puede utilizar las siguientes herramientas para monitorizar sus recursos y canalizaciones de CodePipeline:

- Eventos de bus de eventos de EventBridge: puede supervisar los eventos de CodePipeline en EventBridge, que detecta los cambios en el estado de ejecución de su proceso, fase o acción. EventBridge dirige esos datos a los objetivos, como AWS Lambda y Amazon Simple Notification Service. Los eventos de EventBridge son los mismos que aparecen en Eventos de Amazon CloudWatch.
- Notificaciones de eventos de pipeline en la consola de Developer Tools: puede supervisar los eventos de CodePipeline con las notificaciones que configure en la consola y, a continuación, crear un tema y una suscripción a Amazon Simple Notification Service. Para obtener más información, consulte ¿Qué son las notificaciones? en la Guía del usuario de Developer Tools Console.
- AWS CloudTrail: utilice CloudTrail para capturar las llamadas a las API efectuadas por CodePipeline o en su nombre en su cuenta de AWS, y entregue los archivos de registro al bucket de Amazon S3. Puede hacer que CloudWatch publique notificaciones de Amazon SNS cuando se entreguen archivos de registro nuevos, lo que le permitirá actuar con rapidez.
- Consola y CLI: puede usar la consola de CodePipeline y la CLI para ver los detalles del estado de una canalización o de la ejecución de una determinada canalización.

Temas


- Supervisión de CodePipeline eventos
- Referencia del bucket de marcadores de posición de eventos
- Registro de llamadas a la CodePipeline API con AWS CloudTrail
- CodePipeline CloudWatch métricas


# Supervisión de CodePipeline eventos

Puede monitorear CodePipeline los eventos en EventBridge, lo que proporciona un flujo de datos en tiempo real desde sus propias aplicaciones, aplicaciones de software como servicio (SaaS) y. Servicios de AWS EventBridge dirige esos datos a objetivos como AWS Lambda Amazon Simple Notification Service. Estos eventos son los mismos que aparecen en Amazon CloudWatch Events, que ofrece un flujo casi en tiempo real de los eventos del sistema que describen los cambios en AWS los recursos. Para obtener más información, consulte ¿Qué es Amazon EventBridge? en la Guía del EventBridge usuario de Amazon.

> **Note**
>
> Amazon EventBridge es la forma preferida de gestionar tus eventos. Amazon CloudWatch Events y EventBridge son el mismo servicio y API subyacentes, pero EventBridge ofrecen más funciones. Los cambios que realices en cualquiera de CloudWatch los eventos EventBridge aparecerán en cada consola.

Los eventos se componen de reglas. Una regla se configura seleccionando lo siguiente:

- Patrón de eventos. Cada regla se expresa como un patrón de eventos con el origen y el tipo de eventos que se van a supervisar y los objetivos del evento. Para supervisar los eventos, crea una regla con el servicio que está supervisando como origen de los eventos, por ejemplo CodePipeline. Por ejemplo, puede crear una regla con un patrón de eventos que se utilice CodePipeline como fuente de eventos para activar la regla cuando se produzcan cambios en el estado de una canalización, una etapa o una acción.
- Destinos La nueva regla recibe un servicio seleccionado como destino de eventos. En función del tipo de cambio de estado, es posible que desee enviar notificaciones, capturar información de estado, tomar medidas correctivas, iniciar eventos o adoptar otras medidas. Cuando añadas

tu destino, también debes conceder permisos EventBridge para que pueda invocar el servicio de destino seleccionado.

Cada tipo de evento de cambio de estado de ejecución emite notificaciones con un contenido de mensaje específico, donde:

- La entrada `version` inicial muestra el número de versión del evento.
- La entrada `version` de la sección `detail` de la canalización muestra el número de versión de la estructura de la canalización.
- La entrada `execution-id` de la sección `detail` de la canalización muestra el ID de ejecución de la canalización que ha provocado el cambio de estado. Consulte la llamada a la API GetPipelineExecution en la AWS CodePipeline API Reference.
- La entrada `pipeline-execution-attempt` muestra el número de intentos, o reintentos, del ID de ejecución específico.

CodePipeline informa de un evento EventBridge cada vez que Cuenta de AWS cambia el estado de un recurso de su sitio. Los eventos se emiten de forma garantizada, al menos una vez, para los siguientes recursos:

- Ejecuciones de canalización
- Ejecución de etapas
- Ejecuciones de acción

Los eventos se emiten EventBridge con el patrón de eventos y el esquema detallados anteriormente. En el caso de los eventos procesados, como los que recibe a través de notificaciones que ha configurado en la consola de Developer Tools, el mensaje del evento incluye campos de patrones de eventos con algunas variaciones. Por ejemplo, el campo `detail-type` se convierte en `detailType`. Para obtener más información, consulte la llamada a la PutEvents API en la referencia de la EventBridge API de Amazon.

Los siguientes ejemplos muestran eventos para CodePipeline. Siempre que sea posible, en cada ejemplo se muestra el esquema de un evento emitido junto con el esquema de un evento procesado.

Temas

- Tipos de detalles

- Pipeline-level eventos
- Stage-level eventos
- Action-level eventos
- Creación de una regla que envíe una notificación sobre un evento de canalización

## Tipos de detalles

Al configurar los eventos para monitorizarlos, puede elegir el tipo de detalle del evento.

Puede configurar notificaciones para que se envíen cuando cambie el estado de:

- Determinadas canalizaciones o todas las canalizaciones. Esto se controla utilizando `"detail-type": "CodePipeline Pipeline Execution State Change"`.
- Determinadas etapas o todas las etapas, de una canalización especificada o de todas las canalizaciones. Esto se controla utilizando `"detail-type": "CodePipeline Stage Execution State Change"`.
- Determinadas acciones o todas las acciones, de una etapa especificada o de todas las etapas, de una canalización especificada o de todas las canalizaciones. Esto se controla utilizando `"detail-type": "CodePipeline Action Execution State Change"`.

> **Note**
>
> Los eventos emitidos por EventBridge contienen el `detail-type` parámetro, que se convierte `detailType` cuando se procesan los eventos.

| Tipo de detalle | Estado | Description (Descripción) |
| --- | --- | --- |
| CodePipeline Cambio del estado de ejecución de la canalización | CANCELADO | La ejecución de la canalización se ha cancelado porque se ha actualizado su estructura. |
| | ERROR | La ejecución de la canalización no finalizó correctamente. |
| | RESUMED | Se ha reintentado la ejecución de una canalización que ha fallado en respuesta a la llamada a la API `RetryStageExecution` . |

| Tipo de detalle | Estado | Description (Descripción) |
|---|---|---|
| | STARTED | La ejecución de la canalización está en curso. |
| | STOPPED | El proceso de detención se ha completado y la ejecución de canalización se detiene. |
| | STOPPING | La ejecución de canalización se detiene debido a una solicitud para detener y esperar, o bien detener y abandonar, la ejecución de canalización. |
| | SUCCEEDED | La ejecución de la canalización finalizó correctamente. |
| | SUPERSEDED | Mientras la ejecución de esta canalización esperaba a que se completase la siguiente etapa, una nueva ejecución de la canalización avanzó y continuó a través de la canalización en su lugar. |
| CodePipeline Cambio en el estado de ejecución de la etapa | CANCELADO | La etapa se ha cancelado porque se ha actualizado la estructura de la canalización. |
| | ERROR | La etapa no ha finalizado correctamente. |
| | RESUMED | Se ha reintentado la ejecución de una etapa que ha fallado en respuesta a la llamada a la API `RetryStageExecution` . |
| | STARTED | La etapa se está ejecutando actualmente. |
| | STOPPED | El proceso de detención se ha completado y la ejecución de etapa se detiene. |
| | STOPPING | La ejecución de etapa se detiene debido a una solicitud para detener y esperar, o bien detener y abandonar, la ejecución de canalización. |
| | SUCCEEDED | La etapa ha finalizado correctamente. |

| Tipo de detalle | Estado | Description (Descripción) |
| --- | --- | --- |
| CodePipeline Cambio en el estado de ejecución de la acción | ABANDONED | La acción se abandona debido a una solicitud de detener y abandonar la ejecución de canalización. |
| | CANCELADO | La acción se ha cancelado porque se ha actualizado la estructura de la canalización. |
| | ERROR | Para las acciones de aprobación, el estado FAILED significa que la acción fue rechazada por el revisor o que ha tenido un error debido a una configuración incorrecta de la acción. |
| | STARTED | La acción se está ejecutando actualmente. |
| | SUCCEEDED | La acción ha finalizado correctamente. |

## Pipeline-level eventos

Pipeline-level los eventos se emiten cuando hay un cambio de estado para la ejecución de una canalización.

Temas

- Evento STARTED de canalización
- Evento STOPPING de inanición
- Evento SUCCEEDED de canalización
- Evento SUCCEEDED de canalización (ejemplo con etiquetas Git)
- Evento FAILED de canalización
- Evento FAILED de canalización (ejemplo con etiquetas Git)

### Evento STARTED de canalización

Cuando se inicia la ejecución de una canalización, se emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `"myPipeline"` de la región `us-east-1`. El campo `id` representa el ID del evento y el campo `account` representa el ID de la cuenta en la que se creó la canalización.

## Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:03:07Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "trigger-type": "StartPipelineExecution",
            "trigger-detail": "arn:aws:sts::123456789012:assumed-role/Admin/my-user"
        },
        "state": "STARTED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

## Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-east-1",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:44:50Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
east-1:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "trigger-type": "StartPipelineExecution",
```

```
            "trigger-detail": "arn:aws:sts::123456789012:assumed-role/Admin/my-user"
        },
        "state": "STARTED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

## Evento STOPPING de inanición

Cuando la ejecución de una canalización se detiene, emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `myPipeline` de la región `us-west-2`.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:02:20Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "STOPPING",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
        "stop-execution-comments": "Stopping the pipeline for an update"
    }
}
```

## Evento SUCCEEDED de canalización

Cuando la ejecución de una canalización se realiza correctamente, emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `myPipeline` de la región `us-east-1`.

### Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:03:44Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "SUCCEEDED",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

### Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-east-1",
    "source": "aws.codepipeline",
    "time": "2021-06-30T22:13:51Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
```

```
        "state": "SUCCEEDED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

## Evento SUCCEEDED de canalización (ejemplo con etiquetas Git)

Cuando la ejecución de una canalización tiene una etapa que se ha reintentado y se ha realizado correctamente, emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `myPipeline` en la región `eu-central-1` en la que `execution-trigger` está configurada para las etiquetas de Git.

> **Note**
>
> El campo `execution-trigger` tendrá `tag-name` o `branch-name`, según el tipo de evento que haya activado la canalización.

```
{
    "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
    }
}
```

## Evento FAILED de canalización

Cuando se produce un error en la ejecución de una canalización, se emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `"myPipeline"` de la región `us-west-2`.

Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:55:43Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "FAILED",
        "version": 4.0,
        "pipeline-execution-attempt": 1.0
    }
```

```
}
```

Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:46:16Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "FAILED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "failedActionCount": 1,
        "failedActions": [
            {
                "action": "Deploy",
                "additionalInformation": "Deployment <ID> failed"
            }
        ],
        "failedStage": "Deploy"
    }
```

## Evento FAILED de canalización (ejemplo con etiquetas Git)

A menos que se produzca un error en la fase de origen, en el caso de una canalización configurada con desencadenadores, emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `myPipeline` en la región `eu-central-1` en la que `execution-trigger` está configurada para las etiquetas de Git.

> **Note**
>
> El campo `execution-trigger` tendrá `tag-name` o `branch-name`, según el tipo de evento que haya activado la canalización.

### Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:55:43Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "author-display-name": "Mary Major",
            "full-repository-name": "mmajor/sample-project",
            "provider-type": "GitLab",
            "author-email": "email_address",
            "commit-message": "Update file README.md",
            "author-date": "2023-08-16T21:08:08Z",
            "tag-name": "gitlab-v4.2.1",
            "commit-id": "commit_ID",
            "connection-arn": "arn:aws:codestar-connections:eu-
central-1:123456789012:connection/0f5b706a-1a1d-46c5-86b6-f177321bcfb2",
            "author-id": "Mary Major"
        },
        "state": "FAILED",
        "version": 4.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

## Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:46:16Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "execution-trigger": {
            "author-display-name": "Mary Major",
            "full-repository-name": "mmajor/sample-project",
            "provider-type": "GitLab",
            "author-email": "email_address",
            "commit-message": "Update file README.md",
            "author-date": "2023-08-16T21:08:08Z",
            "tag-name": "gitlab-v4.2.1",
            "commit-id": "commit_ID",
            "connection-arn": "arn:aws:codestar-connections:eu-
central-1:123456789012:connection/0f5b706a-1a1d-46c5-86b6-f177321bcfb2",
            "author-id": "Mary Major"
        },
        "state": "FAILED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "failedActionCount": 1,
        "failedActions": [
            {
                "action": "Deploy",
                "additionalInformation": "Deployment <ID> failed"
            }
        ],
        "failedStage": "Deploy"
```

```
    }
```

## Stage-level eventos

Stage-level los eventos se emiten cuando hay un cambio de estado para la ejecución de una etapa.

Temas

- Evento STARTED e etapa
- Evento STOPPING de etapa
- Evento STOPPED de etapa
- La etapa se reanudó (RESUMED) tras el evento de reintento

### Evento STARTED e etapa

Cuando se inicia la ejecución de una etapa, se emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `"myPipeline"` de la región `us-east-1` para la etapa `Prod`.

Emitted event

```
{
    "version": "0",
    "id": 01234567-EXAMPLE,
    "detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": 123456789012,
    "time": "2020-01-24T22:03:07Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "version": 1.0,
        "execution-id": 12345678-1234-5678-abcd-12345678abcd,
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Prod",
        "state": "STARTED",
        "pipeline-execution-attempt": 1.0
```

```
    }
}
```

## Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Stage Execution State Change",
    "region": "us-east-1",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:45:40Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Source",
        "state": "STARTED",
        "version": 1.0,
        "pipeline-execution-attempt": 0.0
    },
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "sourceActions": [
            {
                "sourceActionName": "Source",
                "sourceActionProvider": "CodeCommit",
                "sourceActionVariables": {
                    "BranchName": "main",
                    "CommitId": "<ID>",
                    "RepositoryName": "my-repo"
                }
            }
        ]
    }
}
```

## Evento STOPPING de etapa

Cuando una ejecución en fase se detiene, emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `myPipeline` de la región `us-west-2` para la etapa `Deploy`.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-24T22:02:20Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Deploy",
        "state": "STOPPING",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

## Evento STOPPED de etapa

Cuando se inicia la ejecución de una etapa, se emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `myPipeline` de la región `us-west-2` para la etapa `Deploy`.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:21:39Z",
    "region": "us-west-2",
```

```
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "stage": "Deploy",
        "state": "STOPPED",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

## La etapa se reanudó (RESUMED) tras el evento de reintento

Cuando se reanuda la ejecución de una etapa y hay una etapa que se ha vuelto a intentar, emite un evento que envía notificaciones con el siguiente contenido.

Cuando se vuelve a intentar una etapa, se muestra el campo `stage-last-retry-attempt-time`, como se muestra en el ejemplo. El campo se muestra en todos los eventos de la etapa si se ha realizado un reintento.

> **Note**
>
> El campo `stage-last-retry-attempt-time` estará presente en todos los eventos de fase posteriores una vez que se haya reintentado una etapa.

```
{
    "version": "0",
    "id": "38656bcd-a798-5f92-c738-02a71be484e1",
    "detail-type": "CodePipeline Stage Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2023-10-26T14:14:56Z",
    "region": "eu-central-1",
    "resources": [
        "arn:aws:codepipeline:eu-central-1:123456789012:BuildFromTag"
    ],
    "detail": {
        "pipeline": "BuildFromTag",
```

```
        "execution-id": "05dafb6a-5a56-4951-a858-968795364846",
        "stage-last-retry-attempt-time": "2023-10-26T14:14:56.305Z",
        "stage": "Build",
        "state": "RESUMED",
        "version": 32.0,
        "pipeline-execution-attempt": 2.0
    }
}
```

## Action-level eventos

Action-level los eventos se emiten cuando hay un cambio de estado para la ejecución de una acción.

Temas

- Evento STARTED de acción
- Evento SUCCEEDED de acción
- Evento FAILED de acción
- Evento ABANDONED de acción

### Evento STARTED de acción

Cuando se inicia la ejecución de una acción, se emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `myPipeline` de la región `us-east-1` y para la acción de implementación `myAction`.

Emitted event

```
{
    "version": "0",
    "id": 01234567-EXAMPLE,
    "detail-type": "CodePipeline Action Execution State Change",
    "source": "aws.codepipeline",
    "account": 123456789012,
    "time": "2020-01-24T22:03:07Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
```

```
        "execution-id": 12345678-1234-5678-abcd-12345678abcd,
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Prod",
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "myAction",
        "state": "STARTED",
        "type": {
            "owner": "AWS",
            "category": "Deploy",
            "provider": "CodeDeploy",
            "version": "1"
        },
        "version": 2.0
        "pipeline-execution-attempt": 1.0
        "input-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-east-1-BUCKETEXAMPLE",
                    "key": "myPipeline/SourceArti/KEYEXAMPLE"
                }
            }
        ]
    }
}
```

## Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Action Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:45:44Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Deploy",
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
```

```
        "input-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-east-1-EXAMPLE",
                    "key": "myPipeline/SourceArti/EXAMPLE"
                }
            }
        ],
        "state": "STARTED",
        "region": "us-east-1",
        "type": {
            "owner": "AWS",
            "provider": "CodeDeploy",
            "category": "Deploy",
            "version": "1"
        },
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

## Evento SUCCEEDED de acción

Cuando la ejecución de una acción se realiza correctamente, emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `"myPipeline"` de la región `us-west-2` y para la acción `"Source"`. Para este tipo de evento, hay dos campos `region` diferentes. El campo `region` de evento especifica la región del evento de canalización. El campo `region` de la sección `detail` especifica la región de la acción.

Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Action Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
```

```
    "time": "2020-01-24T22:03:11Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Source",
        "execution-result": {
            "external-execution-url": "https://us-west-2.console.aws.amazon.com/
codecommit/home#/repository/my-repo/commit/8cf40f2EXAMPLE",
            "external-execution-summary": "Added LICENSE.txt",
            "external-execution-id": "8cf40fEXAMPLE"
        },
        "output-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-west-2-BUCKETEXAMPLE",
                    "key": "myPipeline/SourceArti/KEYEXAMPLE"
                }
            }
        ],
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Source",
        "state": "SUCCEEDED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeCommit",
            "category": "Source",
            "version": "1"
        },
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

## Processed event

```
{
```

```
    "account": "123456789012",
    "detailType": "CodePipeline Action Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:45:44Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:ACCOUNT:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "arn:aws:codepipeline:us-west-2:123456789012:myPipeline",
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Source",
        "execution-result": {
            "external-execution-url": "https://us-west-2.console.aws.amazon.com/
codecommit/home#/repository/my-repo/commit/8cf40f2EXAMPLE",
            "external-execution-summary": "Edited index.html",
            "external-execution-id": "36ab3ab7EXAMPLE"
        },
        "output-artifacts": [
            {
                "name": "SourceArtifact",
                "s3location": {
                    "bucket": "codepipeline-us-west-2-EXAMPLE",
                    "key": "myPipeline/SourceArti/EXAMPLE"
                }
            }
        ],
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Source",
        "state": "SUCCEEDED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeCommit",
            "category": "Source",
            "version": "1"
        },
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
```

```
}
```

## Evento FAILED de acción

Cuando se produce un error en la ejecución de una acción, se emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `"myPipeline"` de la región `us-west-2` y para la acción `"Deploy"`.

Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Action Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:55:43Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:51:09.981Z",
        "stage": "Deploy",
        "execution-result": {
            "external-execution-url": "https://us-west-2.console.aws.amazon.com/
codedeploy/home?#/deployments/<ID>",
            "external-execution-summary": "Deployment <ID> failed",
            "external-execution-id": "<ID>",
            "error-code": "JobFailed"
        },
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
        "state": "FAILED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeDeploy",
            "category": "Deploy",
            "version": "1"
```

```
        },
        "version": 4.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

## Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Action Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:46:16Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "stage": "Deploy",
        "execution-result": {
            "external-execution-url": "https://console.aws.amazon.com/codedeploy/
home?region=us-west-2#/deployments/<ID>",
            "external-execution-summary": "Deployment <ID> failed",
            "external-execution-id": "<ID>",
            "error-code": "JobFailed"
        },
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
        "state": "FAILED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeDeploy",
            "category": "Deploy",
            "version": "1"
        },
        "version": 13.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
```

```
    "additionalAttributes": {
        "additionalInformation": "Deployment <ID> failed"
    }
}
```

## Evento ABANDONED de acción

Cuando se suspende la ejecución de una acción, se emite un evento que envía notificaciones con el siguiente contenido. Este ejemplo es para la canalización denominada `"myPipeline"` de la región `us-west-2` y para la acción `"Deploy"`.

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Action Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:21:39Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "stage": "Deploy",
        "action-execution-id": "47f821c5-a902-44b2-ae61-b878d31ecd21",
        "action": "Deploy",
        "state": "ABANDONED",
        "region": "us-west-2",
        "type": {
            "owner": "AWS",
            "provider": "CodeDeploy",
            "category": "Deploy",
            "version": "1"
        },
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

# Creación de una regla que envíe una notificación sobre un evento de canalización

Una regla detecta ciertos eventos y, a continuación, los dirige a AWS los objetivos que usted elija. Puede crear una regla que ejecute una AWS acción automáticamente cuando se produzca otra AWS acción, o una regla que ejecute una AWS acción de forma regular según un cronograma establecido.

Temas


- Enviar una notificación cada vez que cambie el estado de una canalización (consola)
- Enviar una notification cuando cambie el estado de una canalización (CLI)


## Enviar una notificación cada vez que cambie el estado de una canalización (consola)

Estos pasos muestran cómo usar la EventBridge consola para crear una regla que envíe notificaciones cuando se produzcan cambios CodePipeline.

Para crear una EventBridge regla que se dirija a su canalización con una fuente de Amazon S3

1. Abra la EventBridge consola de Amazon en https://console.aws.amazon.com/events/.
2. En el panel de navegación, seleccione Reglas. Deje el bus predeterminado seleccionado o elija un bus de eventos. Elija Crear regla.
3. En Nombre, introduzca un nombre para la regla.
4. En Tipo de regla, elija Regla con un patrón de evento. Elija Siguiente.
5. En Patrón de eventos, seleccione Servicios de AWS .
6. En la lista desplegable Tipo de evento, elija el nivel de cambio de estado de la notificación.
   - Para ver una regla que se aplique a los eventos a nivel de canalización, elija CodePipeline Pipeline Execution State Change.
   - Para una regla que se aplique a los eventos a nivel de etapa, elija CodePipeline Stage Execution State Change.
   - Para una regla que se aplique a los eventos a nivel de acción, elija Action CodePipeline Execution State Change.
7. Especifique los cambios de estado a los que se aplica la regla:
   - Para una regla que se aplica a todos los cambios de estado, elija Any state.

- Para una regla que se aplica únicamente a algunos cambios de estado, elija Specific state(s) y, a continuación, elija uno o varios valores de estado de la lista.

8. Si necesita patrones de eventos más detalladas de lo que permiten los selectores, también puede utilizar la opción Editar patrón de la ventana Vista previa de patrón de evento para designar un patrón de eventos con formato JSON.

> **Note**
>
> Si no se especifica lo contrario, se crea el patrón de eventos para todas las pipelines/ stages /actions y states.

Para obtener patrones de eventos más detallados, puede copiar y pegar los siguientes patrones de eventos de muestra en la ventana Editar.

- Example

  Utilice este patrón de eventos de muestra para capturar acciones de implementación y compilación con errores en todas las canalizaciones.

```
{
"source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Deploy", "Build"]
    }
  }
}
```

- Example

  Utilice este patrón de eventos de muestra para capturar todas las acciones de aprobación rechazadas y con errores en todas las canalizaciones.

```
{
 "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Approval"]
    }
  }
}
```

- Example

  Utilice este patrón de eventos de muestra para capturar todos los eventos de las canalizaciones especificadas.

```
{
"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. Elija Siguiente.

10. En Tipos de destino, elija Servicio de AWS .
11. En  Seleccione un objetivo, elija  CodePipeline. En ARN de canalización, Introduzca el ARN de la canalización que se iniciará cuando la active esta regla.

> **Note**
>
> Para obtener el ARN de la canalización, ejecute el comando get-pipeline. El ARN de la canalización aparece en la salida. Se crea con el siguiente formato:
> arn:aws:codepipeline::: *region account pipeline-name*
> ARN de canalización de muestra:
> arn:aws:codepipeline: us-east - 2:80398 EXAMPLE:MyFirstPipeline

12. Para crear o especificar un rol de servicio de IAM que conceda EventBridge permisos para invocar el objetivo asociado a tu regla (en este caso, el objetivo es): EventBridge CodePipeline
    - Elige  Crear un nuevo rol para este recurso específico  a fin de crear un rol de servicio que otorgue EventBridge permisos para iniciar las ejecuciones de tu canalización.
    - Elige  Usar el rol existente  para ingresar un rol de servicio que otorgue EventBridge permisos para iniciar las ejecuciones de tu canalización.
13. Elija Siguiente.
14. En la página Etiquetas, elija Siguiente:
15. En la página Revisar y crear, revise la configuración de la regla. Si está satisfecho con la regla, elija Create rule (Crear regla).

## Enviar una notification cuando cambie el estado de una canalización (CLI)

En estos pasos se muestra cómo usar la CLI para crear una regla de CloudWatch eventos que envíe notificaciones cuando se produzcan cambios CodePipeline.

Para usar la AWS CLI para crear una regla, llame al put-rule comando y especifique:

- Un nombre que identifique de forma inequívoca la regla que está creando. Este nombre debe ser único en todas las canalizaciones que crees y que estén CodePipeline asociadas a tu AWS cuenta.
- El patrón de eventos para el origen y los campos de detalles utilizados por la regla. Para obtener más información, consulta  Amazon EventBridge y Event Patterns.

Para crear una EventBridge regla con CodePipeline como fuente de eventos

1. Use el comando put-rule para crear una regla que especifique el patrón de eventos. (Consulte las tablas anteriores para ver los estados válidos).

   El siguiente comando de ejemplo se utiliza --event-pattern para crear una regla denominada `"MyPipelineStateChanges"` que emite el CloudWatch evento cuando se produce un error en la ejecución de una canalización en la canalización denominada «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. Llame al comando put-targets e incluya los siguientes parámetros:

   - El parámetro `--rule` se usa con el `rule_name` que creó con el comando put-rule.
   - El parámetro `--targets` se usa con el `Id` del destino de la lista de destinos y el `ARN` del tema de Amazon SNS.

   El siguiente comando de muestra especifica que, para la regla denominada `MyPipelineStateChanges`, el destino `Id` se compone del número uno, lo que indica que, en lo que puede ser una lista de destinos de la regla, se trata del destino 1. El comando de muestra también especifica un `ARN` de ejemplo para el tema de Amazon SNS.

   ```
   aws events put-targets --rule MyPipelineStateChanges --targets
    Id=1,Arn=arn:aws:sns:us-west-2:11111EXAMPLE:MyNotificationTopic
   ```

3. Añada permisos EventBridge para usar el servicio de destino designado para invocar la notificación. Para obtener más información, consulte Uso de políticas basadas en recursos para Amazon. EventBridge

# Referencia del bucket de marcadores de posición de eventos

Esta sección se incluye solo como referencia. Para obtener información acerca de cómo crear una canalización con recursos de detección de eventos, consulte Acciones de origen y métodos de detección de cambios.

Las acciones de origen proporcionadas por Amazon S3 y CodeCommit utilizan recursos de detección de cambios basados en eventos para activar la canalización cuando se realiza un cambio en el bucket o repositorio de origen. Estos recursos son las reglas de Eventos de CloudWatch que están configuradas para responder a los eventos que se producen en el origen de la canalización, como un cambio de código en el repositorio de CodeCommit. Cuando utilice Eventos de CloudWatch para un origen de Amazon S3, debe activar CloudTrail para que se registren los eventos. CloudTrail necesita un bucket de S3 al que pueda enviar sus resúmenes. Puede acceder a los archivos de registro de sus recursos de Eventos de CloudWatch desde el depósito personalizado, pero no puede acceder a los datos desde el depósito de marcadores de posición.

- Si utilizó la CLI o CloudFormation para configurar los recursos de Eventos de CloudWatch, encontrará los archivos de CloudTrail en el bucket que especificado al configurar la canalización.
- Si ha utilizado la consola para configurar su canalización con un origen de S3, la consola utiliza un bucket de marcadores de posición de CloudTrail cuando crea los recursos de Eventos de CloudWatch por usted. Los resúmenes de CloudTrail se almacenan en el bucket de marcadores de posición en la Región de AWS en la que se creó la canalización.

Puede cambiar la configuración si desea utilizar otro bucket que no sea el de marcadores de posición.

> **Note**
>
> Los datos escritos en los buckets de marcadores de posición de CloudTrail caducan automáticamente al cabo de un día y no se retienen.

Para obtener más información acerca de cómo buscar y administrar los archivos de registro de CloudTrail , consulte Obtención y visualización de archivos de registro de CloudTrail.

Temas

- Nombres de buckets de marcadores de posición de eventos por región

## Nombres de buckets de marcadores de posición de eventos por región

En esta tabla, se muestran los nombres de los buckets de marcadores de posición de S3 que contienen archivos de registro que hacen un seguimiento de los eventos de detección de cambios en las canalizaciones con acciones de origen de Amazon S3.

| Nombre de la región | Nombre del bucket de marcadores de posición | Identificador de la región |
|---|---|---|
| Este de EE. UU. (Ohio) | codepipeline-cloudtrail-placeholder-bucket-us-east-2 | us-east-2 |
| Este de EE. UU. (Norte de Virginia) | codepipeline-cloudtrail-placeholder-bucket-us-east-1 | us-east-1 |
| Oeste de EE. UU. (Norte de California) | codepipeline-cloudtrail-placeholder-bucket-us-west-1 | us-west-1 |
| Oeste de EE. UU. (Oregón) | codepipeline-cloudtrail-placeholder-bucket-us-west-2 | us-west-2 |
| Canadá (centro) | codepipeline-cloudtrail-placeholder-bucket-ca-central-1 | ca-central-1 |
| Europa (Fráncfort) | codepipeline-cloudtrail-placeholder-bucket-eu-central-1 | eu-central-1 |
| Europa (Irlanda) | codepipeline-cloudtrail-placeholder-bucket-eu-west-1 | eu-west-1 |
| Europa (Londres) | codepipeline-cloudtrail-placeholder-bucket-eu-west-2 | eu-west-2 |
| Europa (París) | codepipeline-cloudtrail-placeholder-bucket-eu-west-3 | eu-west-3 |
| Europa (Estocolmo) | codepipeline-cloudtrail-placeholder-bucket-eu-north-1 | eu-north-1 |
| Asia-Pacífico (Hong Kong) | codepipeline-cloudtrail-placeholder-bucket-ap-east-1 | ap-east-1 |
| Asia-Pacífico (Hyderabad) | codepipeline-cloudtrail-placeholder-bucket-ap-south-2 | ap-south-2 |

| Nombre de la región | Nombre del bucket de marcadores de posición | Identificador de la región |
|---|---|---|
| Asia-Pacífico (Yakarta) | codepipeline-cloudtrail-placeholder-bucket-ap-southeast-3 | ap-southeast-3 |
| Asia-Pacífico (Melbourne) | codepipeline-cloudtrail-placeholder-bucket-ap-southeast-4 | ap-southeast-4 |
| Asia-Pacífico (Mumbai) | codepipeline-cloudtrail-placeholder-bucket-ap-south-1 | ap-south-1 |
| Asia-Pacífico (Osaka) | codepipeline-cloudtrail-placeholder-bucket-ap-northeast-3-prod | ap-northeast-3 |
| Asia-Pacífico (Tokio) | codepipeline-cloudtrail-placeholder-bucket-ap-northeast-1 | ap-northeast-1 |
| Asia-Pacífico (Seúl) | codepipeline-cloudtrail-placeholder-bucket-ap-northeast-2 | ap-northeast-2 |
| Asia-Pacífico (Singapur) | codepipeline-cloudtrail-placeholder-bucket-ap-southeast-1 | ap-southeast-1 |
| Asia-Pacífico (Sídney) | codepipeline-cloudtrail-placeholder-bucket-ap-southeast-2 | ap-southeast-2 |
| Asia-Pacífico (Tokio) | codepipeline-cloudtrail-placeholder-bucket-ap-northeast-1 | ap-northeast-1 |

| Nombre de la región | Nombre del bucket de marcadores de posición | Identificador de la región |
|---|---|---|
| Canadá (centro) | codepipeline-cloudtrail-placeholder-bucket-ca-central-1 | ca-central-1 |
| Europa (Fráncfort) | codepipeline-cloudtrail-placeholder-bucket-eu-central-1 | eu-central-1 |
| Europa (Irlanda) | codepipeline-cloudtrail-placeholder-bucket-eu-west-1 | eu-west-1 |
| Europa (Londres) | codepipeline-cloudtrail-placeholder-bucket-eu-west-2 | eu-west-2 |
| Europa (Milán) | codepipeline-cloudtrail-placeholder-bucket-eu-south-1 | eu-south-1 |
| Europa (París) | codepipeline-cloudtrail-placeholder-bucket-eu-west-3 | eu-west-3 |
| Europa (España) | codepipeline-cloudtrail-placeholder-bucket-eu-south-2 | eu-south-2 |
| Europa (Estocolmo) | codepipeline-cloudtrail-placeholder-bucket-eu-north-1 | eu-north-1 |
| Europa (Zúrich)* | codepipeline-cloudtrail-placeholder-bucket-eu-central-2 | eu-central-2 |
| Israel (Tel Aviv) | codepipeline-cloudtrail-placeholder-bucket-il-central-1 | il-central-1 |
| Medio Oriente (Baréin)* | codepipeline-cloudtrail-placeholder-bucket-me-south-1 | me-south-1 |
| Medio Oriente (EAU) | codepipeline-cloudtrail-placeholder-bucket-me-central-1 | me-central-1 |

| Nombre de la región | Nombre del bucket de marcadores de posición | Identificador de la región |
| --- | --- | --- |
| América del Sur (São Paulo) | codepipeline-cloudtrail-placeholder-bucket-sa-east-1 | sa-east-1 |

# Registro de llamadas a la CodePipeline API con AWS CloudTrail

AWS CodePipeline está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o una Servicio de AWS entrada CodePipeline. CloudTrail captura todas las llamadas a la API CodePipeline como eventos. Las llamadas capturadas incluyen llamadas desde la CodePipeline consola y llamadas de código a las operaciones de la CodePipeline API. Si crea un registro, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos para CodePipeline. Si no configuras un registro, puedes seguir viendo los eventos más recientes en la CloudTrail consola, en el Historial de eventos. Con la información recopilada por CloudTrail, puedes determinar la solicitud a la que se hizo CodePipeline, la dirección IP desde la que se hizo la solicitud, quién la hizo, cuándo se hizo y otros detalles.

Para obtener más información CloudTrail, consulta la Guía AWS CloudTrail del usuario.

## CodePipeline información en CloudTrail

CloudTrail está activado en tu Cuenta de AWS momento de crear la cuenta. Cuando se produce una actividad en CodePipeline, esa actividad se registra en un CloudTrail evento junto con otros Servicio de AWS eventos en el historial de eventos. Puedes ver, buscar y descargar los eventos recientes en tu AWS cuenta. Para obtener más información, consulta Visualización de eventos con el historial de CloudTrail eventos.

Para obtener un registro continuo de los eventos de tu Cuenta de AWS cuenta, incluidos los eventos de CodePipeline, crea una ruta. Un registro permite CloudTrail enviar archivos de registro a un bucket de Amazon S3. De forma predeterminada, al crear una ruta en la consola, la ruta se aplica a todas AWS las regiones. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al bucket de Amazon S3 que usted especifique. Además, puede configurar otros Servicios de AWS para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para más información, consulte los siguientes temas:

- Introducción a la creación de registros de seguimiento

- CloudTrail Servicios e integraciones compatibles
- Configuración de las notificaciones de Amazon SNS para CloudTrail
- Recepción de archivos de CloudTrail registro de varias regiones y recepción de archivos de CloudTrail registro de varias cuentas

Todas CodePipeline las acciones se registran en la referencia de la CodePipeline API CloudTrail y se documentan en ella. Por ejemplo, las llamadas a la `CreatePipeline` `GetPipelineExecution` y `UpdatePipeline` las acciones generan entradas en los archivos de CloudTrail registro.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales raíz o AWS Identity and Access Management (IAM).
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro Servicio de AWS.

Para obtener más información, consulte el Elemento userIdentity de CloudTrail .

## Descripción de las entradas del archivo de CodePipeline registro

Un registro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una única solicitud de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail los archivos de registro no son un seguimiento ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro de un evento de canalización de actualizaciones, en la que el usuario denominado JaneDoe-CodePipeline con el ID de cuenta 80398EXAMPLE MyFirstPipeline ha editado una canalización denominada. El usuario cambió el nombre de la etapa de origen de una canalización de `Source` a `MySourceStage`. Dado que tanto el elemento `requestParameters` como el elemento `responseElements` del registro de CloudTrail contienen la estructura completa de la canalización modificada, aparecen abreviados en el ejemplo siguiente. Se ha añadido Emphasis a la parte `requestParameters` de la canalización en la que se produjo el cambio, el número de versión anterior de la canalización y la parte `responseElements`,

que muestra el número de versión incrementado en una unidad (+1). Las partes modificadas se marcan con puntos suspensivos (...) para indicar que la entrada de log real incluye más datos.


```
{
 "eventVersion":"1.03",
  "userIdentity": {
   "type":"IAMUser",
   "principalId":"AKIAI44QH8DHBEXAMPLE",
   "arn":"arn:aws:iam::80398EXAMPLE:user/JaneDoe-CodePipeline",
   "accountId":"80398EXAMPLE",
   "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
   "userName":"JaneDoe-CodePipeline",
   "sessionContext": {
   "attributes":{
    "mfaAuthenticated":"false",
    "creationDate":"2015-06-17T14:44:03Z"
    }
   },
 "invokedBy":"signin.amazonaws.com"},
 "eventTime":"2015-06-17T19:12:20Z",
 "eventSource":"codepipeline.amazonaws.com",
 "eventName":"UpdatePipeline",
 "awsRegion":"us-east-2",
 "sourceIPAddress":"192.0.2.64",
 "userAgent":"signin.amazonaws.com",
 "requestParameters":{
   "pipeline":{
   "version":1,
   "roleArn":"arn:aws:iam::80398EXAMPLE:role/CodePipeline_Service_Role",
   "name":"MyFirstPipeline",
   "stages":[
     {
     "actions":[
      {
          "name":"MySourceStage",
        "actionType":{
        "owner":"AWS",
        "version":"1",
        "category":"Source",
        "provider":"S3"
     },
    "inputArtifacts":[],
    "outputArtifacts":[
```

```
     {"name":"MyApp"}
     ],
    "runOrder":1,
    "configuration":{
     "S3Bucket":"amzn-s3-demo-source-bucket",
     "S3ObjectKey":"sampleapp_linux.zip"
      }
      }
     ],
       "name":"Source"
     },
     (...)
               },
 "responseElements":{
   "pipeline":{
     "version":2,
     (...)
           },
     "requestID":"2c4af5c9-7ce8-EXAMPLE",
     "eventID":""c53dbd42-This-Is-An-Example"",
     "eventType":"AwsApiCall",
     "recipientAccountId":"80398EXAMPLE"
     }
      ]
}
```

# CodePipeline CloudWatch métricas

Puede utilizar las métricas CloudWatch para medir las CodePipeline canalizaciones. Puede utilizar las siguientes métricas para medir la duración de las canalizaciones y las ejecuciones de canalizaciones con errores.

Puede supervisar las canalizaciones en dos niveles:

Nivel de canalización

Estas métricas se encuentran por canalización. Selecciona Por canalización en CloudWatch. Las canalizaciones disponibles se enumerarán en CloudWatch.

AWS nivel de cuenta

Estas métricas corresponden a todas las canalizaciones de una cuenta. Para ver las métricas a nivel de AWS cuenta, selecciona Estadísticas de la cuenta en CloudWatch.

Para obtener más información sobre la visualización de las métricas, consulta Ver las métricas disponibles en la Guía del CloudWatch usuario de Amazon.

Temas


- PipelineDuration
- FailedPipelineExecutions


## PipelineDuration

La métrica `PipelineDuration` mide la duración de la ejecución de la canalización. Esta métrica solo está disponible por canalización.

Unidades: segundos

## FailedPipelineExecutions

La métrica `FailedPipelineExecutions` mide el número de ejecuciones de canalización con errores. Esta métrica está disponible por canalización y por cuenta.

Esta métrica incluye un recuento independiente para las ejecuciones de canalización con reintentos de acciones fallidas. En otras palabras, cuando se reintenta una acción fallida en una canalización, se cuenta como una métrica de ejecución de canalización independiente. Sería el caso, por ejemplo, de una canalización con una acción de origen de S3 con un primer intento fallido que se ejecuta nuevamente y resulta en un segundo intento fallido. Para este ejemplo, la métrica sería la siguiente:

```
FailedPipelineExecutionAttempts: 2
```

Unidades: recuento

# Solución de problemas CodePipeline

La siguiente información puede ayudarle a solucionar problemas comunes en AWS CodePipeline.

Temas


- Error de canalización: una canalización configurada con AWS Elastic Beanstalk devuelve un mensaje de error: «Falló la implementación. El rol proporcionado no tiene permisos suficientes: Service:AmazonElasticLoadBalancing»
- Error de implementación: una canalización configurada con un AWS Elastic Beanstalk la acción de despliegue se bloquea en lugar de fallar si falta el permiso DescribeEvents «»
- Error de canalización: una acción de origen devuelve el mensaje de permisos insuficientes: «No se ha podido acceder al nombre del CodeCommit repositorio». Asegúrese de que rol de IAM de la canalización tenga permisos suficientes para obtener acceso al repositorio".
- Error de canalización: una acción de compilación o prueba de Jenkins se ejecuta de manera prolongada y da un error debido a la falta de credenciales o permisos
- Error de canalización: se creó una canalización en una AWS Región que utiliza un depósito creado en otro AWS La región devuelve un "InternalError" con el código "JobFailed»
- Error de despliegue: un archivo ZIP que contiene un archivo WAR se ha desplegado correctamente en AWS Elastic Beanstalk, pero la URL de la aplicación indica un error 404 no encontrado
- Los nombres de la carpeta de artefactos de la canalización parecen estar truncados
- Añade CodeBuild GitClone permisos para las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com
- Agregue CodeBuild GitClone permisos para las acciones CodeCommit de origen
- Error de proceso: una implementación con la CodeDeployToECS acción devuelve un mensaje de error: «Se produjo una excepción al intentar leer el archivo de artefactos de definición de tareas desde: nombre del artefacto de < origen» >
- GitHub (a través de la aplicación OAuth) acción de origen: la lista de repositorios muestra diferentes repositorios
- GitHub (mediante GitHub la aplicación) acción de origen: no se ha podido completar la conexión a un repositorio
- Error de Amazon S3: al CodePipeline ARN del rol < de servicio > se le deniega el acceso a S3 para el bucket de S3 < BucketName >

- Las canalizaciones con Amazon S3, Amazon ECR o CodeCommit fuente ya no se inician automáticamente
- Error de conexión al conectarse a GitHub: «Se ha producido un problema, asegúrate de que las cookies estén habilitadas en tu navegador» o «El propietario de una organización debe instalar la GitHub aplicación»
- Las canalizaciones con el modo de ejecución cambiado a MODO EN COLA o PARALELO fallan cuando se alcanza el límite de ejecución
- Las canalizaciones en modo PARALELO tienen una definición de canalización desactualizada si se editan al cambiar al modo EN COLA o SUPERSEDED
- Las canalizaciones que cambien del modo PARALELO a otro mostrarán un modo de ejecución anterior
- Es posible que las canalizaciones con conexiones que utilicen el filtrado de desencadenadores por rutas de archivo no se inicien al crear la ramificación
- Es posible que las canalizaciones con conexiones que utilicen el filtrado de desencadenadores por rutas de archivo no se inicien cuando se alcance el límite de archivos
- CodeCommit o las revisiones del código fuente de S3 en modo paralelo podrían no coincidir con el EventBridge evento
- La acción de despliegue de EC2 falla y muestra el mensaje de error No existe tal archivo
- La acción EKS Deploy falla y aparece un mensaje de error inalcanzable para el clúster
- La canalización no se activa en todas las sucursales cuando varias acciones de origen hacen referencia al mismo repositorio
- ¿Necesita ayuda con otro problema?


# Error de canalización: una canalización configurada con AWS Elastic Beanstalk devuelve un mensaje de error: «Falló la implementación. El rol proporcionado no tiene permisos suficientes: Service:AmazonElasticLoadBalancing»

Problema: el rol de servicio CodePipeline no tiene permisos suficientes para AWS Elastic Beanstalk, entre otras, algunas operaciones en Elastic Load Balancing. El rol de servicio CodePipeline se actualizó el 6 de agosto de 2015 para solucionar este problema. Los clientes que crearon su rol

de servicio antes de esta fecha deben modificar la instrucción de política de su rol de servicio para añadir los permisos necesarios.

Posibles correcciones: la solución más fácil es editar la instrucción de política para su rol de servicio como se detalla en Agrega permisos al rol de CodePipeline servicio.

Una vez aplicada la política editada, siga los pasos de Iniciar la canalización manualmente para volver a ejecutar manualmente las canalizaciones que usa Elastic Beanstalk..

En función de sus requisitos de seguridad, puede modificar también los permisos de otras dos formas.

## Error de implementación: una canalización configurada con un AWS Elastic Beanstalk la acción de despliegue se bloquea en lugar de fallar si falta el permiso DescribeEvents «»

Problema: la función de servicio CodePipeline debe incluir la `"elasticbeanstalk:DescribeEvents"` acción de cualquier canalización que se utilice. AWS Elastic Beanstalk Sin este permiso, las acciones de AWS Elastic Beanstalk despliegue se bloquean sin fallar ni indicar un error. Si tu función de servicio no incluye esta acción, significa que CodePipeline no tienes permisos para ejecutar la fase de despliegue de la canalización AWS Elastic Beanstalk en tu nombre.

Posibles soluciones: revisa tu rol CodePipeline de servicio. Si falta la acción `"elasticbeanstalk:DescribeEvents"`, utilice los pasos de Agrega permisos al rol de CodePipeline servicio para añadirla mediante la característica Edit Policy de la consola de IAM.

Una vez aplicada la política editada, siga los pasos de Iniciar la canalización manualmente para volver a ejecutar manualmente las canalizaciones que usa Elastic Beanstalk..

# Error de canalización: una acción de origen devuelve el mensaje de permisos insuficientes: «No se ha podido acceder al nombre del CodeCommit **`repositorio`**». Asegúrese de que rol de IAM de la canalización tenga permisos suficientes para obtener acceso al repositorio".

Problema: el rol de servicio CodePipeline no tiene permisos suficientes CodeCommit y probablemente se creó antes de que se añadiera la compatibilidad con el uso de CodeCommit repositorios el 18 de abril de 2016. Los clientes que crearon su rol de servicio antes de esta fecha deben modificar la instrucción de política de su rol de servicio para añadir los permisos necesarios.

Posibles soluciones: añade los permisos necesarios CodeCommit a la política de tu función de CodePipeline servicio. Para obtener más información, consulte Agrega permisos al rol de CodePipeline servicio.

# Error de canalización: una acción de compilación o prueba de Jenkins se ejecuta de manera prolongada y da un error debido a la falta de credenciales o permisos

Problema: si el servidor Jenkins está instalado en una instancia de Amazon EC2, es posible que la instancia no se haya creado con un rol de instancia que tenga los permisos necesarios. CodePipeline Si utiliza un usuario de IAM en un servidor de Jenkins, en una instancia local o en una instancia de Amazon EC2 creada sin el rol de IAM necesario, el usuario de IAM no tiene los permisos necesarios o el servidor de Jenkins no puede tener acceso a esas credenciales a través del perfil configurado en el servidor.

Soluciones posibles: asegúrese de que el rol de la instancia de Amazon EC2 o el usuario de IAM estén configurados con la política administrada `AWSCodePipelineCustomActionAccess` o con los permisos equivalentes. Para obtener más información, consulte AWS políticas gestionadas para AWS CodePipeline.

Si utiliza un usuario de IAM, asegúrese de que el AWS perfil configurado en la instancia utilice el usuario de IAM configurado con los permisos correctos. Puede que tengas que proporcionar las credenciales de usuario de IAM que configuraste para la integración entre Jenkins y CodePipeline

directamente en la interfaz de usuario de Jenkins. Este procedimiento no se recomienda. Si tiene que hacerlo, asegúrese de que el servidor de Jenkins está protegido y usa HTTPS en lugar de HTTP.

## Error de canalización: se creó una canalización en una AWS Región que utiliza un depósito creado en otro AWS La región devuelve un "InternalError" con el código "JobFailed»

Problema: La descarga de un artefacto almacenado en un bucket de Amazon S3 fallará si la canalización y el bucket se crean en AWS regiones diferentes.

Posibles soluciones: asegúrate de que el depósito de Amazon S3 en el que está almacenado tu artefacto esté en la misma AWS región que la canalización que has creado.

## Error de despliegue: un archivo ZIP que contiene un archivo WAR se ha desplegado correctamente en AWS Elastic Beanstalk, pero la URL de la aplicación indica un error 404 no encontrado

Problema: un archivo WAR se ha implementado correctamente en un entorno de AWS Elastic Beanstalk , pero la URL de la aplicación devuelve un error 404 Not Found.

Posibles soluciones: AWS Elastic Beanstalk puede desempaquetar un archivo ZIP, pero no un archivo WAR contenido en un archivo ZIP. En lugar de especificar un archivo WAR en su archivo `buildspec.yml`, especifique una carpeta que incluya el contenido que se va a implementar. Por ejemplo:

```
version: 0.2

phases:
  post_build:
    commands:
      - mvn package
      - mv target/my-web-app ./
artifacts:
  files:
    - my-web-app/**/*
 discard-paths: yes
```

Para ver un ejemplo, consulte Ejemplo de AWS Elastic Beanstalk para CodeBuild.

# Los nombres de la carpeta de artefactos de la canalización parecen estar truncados

Problema: Al ver los nombres de los artefactos de la canalización CodePipeline, estos aparecen truncados. Esto puede hacer que parezca que los nombres son similares o que ya no contienen el nombre completo de la canalización.

Explicación: CodePipeline trunca los nombres de los artefactos para garantizar que la ruta completa de Amazon S3 no supere los límites de tamaño de la política al CodePipeline generar credenciales temporales para los trabajadores.

Aunque el nombre del artefacto parezca estar truncado, se CodePipeline asigna al depósito de artefactos de una forma que no se ve afectada por los artefactos con nombres truncados. La canalización puede funcionar con normalidad. Esto no supone un problema con la carpeta ni con los artefactos. Los nombres de las canalizaciones tienen una longitud máxima de 100 caracteres. Aunque el nombre de la carpeta de artefactos parezca estar acortado, sigue siendo único para la canalización.

# Añade CodeBuild GitClone permisos para las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com

Cuando utilizas una acción AWS CodeConnections en una fuente y una CodeBuild acción, hay dos maneras de pasar el artefacto de entrada a la compilación:

- Predeterminado: la acción de origen genera un archivo zip que contiene el código que se CodeBuild descarga.
- Clon completo: el código fuente se puede descargar directamente al entorno de compilación.

  El modo de clonación completa te permite interactuar con el código fuente como un repositorio Git en funcionamiento. Para usar este modo, debes conceder permisos a tu CodeBuild entorno para usar la conexión.

Para agregar permisos a su política de rol de CodeBuild servicio, debe crear una política administrada por el cliente y adjuntarla a su rol de CodeBuild servicio. Los siguientes pasos crean una política en la que el `UseConnection` permiso se especifica en el `action` campo, el ARN de la conexión se especifica en el `Resource` campo y el ID del repositorio de origen se limita a través de. `Condition`

Para usar la consola para añadir los permisos UseConnection

1. Para encontrar el ARN de conexión y el ID del repositorio de origen de tu canalización, abre tu canalización, haz clic en el icono (i) de la acción de origen y cambia a la pestaña Entrada.

   Un ejemplo de ARN de conexión es:

   ```
   arn:aws:codeconnections:eu-central-1:123456789123:connection/
   sample-1908-4932-9ecc-2ddacee15095
   ```

   Un ejemplo del ID del repositorio de origen:

   ```
   owner/test-app
   ```

   Agrega el ARN de conexión y el ID del repositorio a la política de rol CodeBuild de servicio.
2. Para encontrar tu rol CodeBuild de servicio, elige el proyecto de compilación utilizado en tu proceso y navega hasta la pestaña de detalles de la compilación.
3. Seleccione el enlace Service role (Rol de servicio). Se abre la consola de IAM, donde puede añadir una nueva política que conceda acceso a la conexión.
4. En la consola de IAM, selecciona Añadir permisos y, a continuación, selecciona Crear política integrada.

   Utilice la siguiente plantilla de política de ejemplo. Agrega tu ARN de conexión en el `Resource` campo y tu ID `codeconnections:FullRepositoryId` de repositorio en el `Condition` campo, como se muestra en este ejemplo:

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": "codeconnections:UseConnection",
               "Resource": "arn:aws:codeconnections:eu-
   central-1:123456789123:connection/my-connection-id",
               "Condition": {
                   "StringEquals": {
                       "codeconnections:FullRepositoryId": "my-repository-id"
   ```

```
                }
            }
        }
    ]
}
```

Usa el `Condition` campo para reducir aún más los permisos de tu política en función de tus requisitos de especificaciones de compilación (consulta la documentación sobre `CodeConnection` las condiciones).

En la pestaña JSON pegue la política.

5. Elija Siguiente. Escriba un nombre para la política (por ejemplo, **`connection-permissions`**) y elija Create policy (Crear política).

   Verás la **`connection-permissions`** política adjunta a las políticas de permisos de tu rol.

# Agregue CodeBuild GitClone permisos para las acciones CodeCommit de origen

Cuando tu canalización tiene una acción de CodeCommit origen, puedes pasar el artefacto de entrada a la compilación de dos maneras:

- Predeterminado: la acción de origen genera un archivo zip que contiene el código que se CodeBuild descarga.
- Clonación completa: el código fuente se puede descargar directamente en el entorno de compilación.

  El modo Clonación completa le permite interactuar con el código fuente como un repositorio de Git de trabajo. Para usar este modo, debe agregar permisos para que su CodeBuild entorno los extraiga del repositorio.

Para agregar permisos a su política de rol de CodeBuild servicio, debe crear una política administrada por el cliente y adjuntarla a su rol de CodeBuild servicio. Los siguientes pasos crean una política que especifica el permiso `codecommit:GitPull` en el campo `action`.

Para usar la consola para añadir los permisos GitPull

1. Para encontrar tu rol CodeBuild de servicio, abre el proyecto de compilación utilizado en tu proceso y navega hasta la pestaña de detalles de la compilación.
2. Seleccione el enlace Service role (Rol de servicio). Se abre la consola de IAM, donde puede añadir una nueva política que conceda acceso al repositorio.
3. En la consola de IAM, elija Attach policies (Asociar políticas), y, a continuación, elija Create policy (Crear política).
4. En la pestaña JSON, pegue el ejemplo de política siguiente:

```
{
    "Action": [
        "codecommit:GitPull"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
```

5. Elija Revisar política. Escriba un nombre para la política (por ejemplo, **`codecommit-gitpull`**) y elija Create policy (Crear política).
6. Vuelva a la página en la que asoció los permisos, actualice la lista de políticas y seleccione la política que acaba de crear. Seleccione Asociar políticas.

# Error de proceso: una implementación con la CodeDeployToECS acción devuelve un mensaje de error: «Se produjo una excepción al intentar leer el archivo de artefactos de definición de tareas desde: nombre del artefacto de < origen» >

Problema:

El archivo de definición de tareas es un artefacto obligatorio para la acción de CodePipeline implementación en Amazon ECS a través de CodeDeploy (la `CodeDeployToECS` acción). El tamaño máximo del ZIP del artefacto en la acción de implementación `CodeDeployToECS` es de 3 MB. Se devuelve el siguiente mensaje de error cuando no se encuentra el archivo o el tamaño del artefacto supera los 3 MB:

Excepción al intentar leer el archivo de artefactos de definición de tareas de: <nombre del artefacto del origen>

Posibles correcciones: asegúrese de que el archivo de definición de la tarea se incluya como un artefacto. Si el archivo ya existe, asegúrese de que el tamaño comprimido sea inferior a 3 MB.

## GitHub (a través de la aplicación OAuth) acción de origen: la lista de repositorios muestra diferentes repositorios

Problema:

Tras autorizar correctamente una acción GitHub (mediante la aplicación OAuth) en la CodePipeline consola, puedes elegir entre una lista de tus repositorios. GitHub Si la lista no incluye los repositorios que esperaba ver, puede solucionar el problema de la cuenta utilizada para la autorización.

Correcciones posibles: la lista de repositorios que aparece en la CodePipeline consola se basa en la GitHub organización a la que pertenece la cuenta autorizada. Compruebe que la cuenta con la que va a realizar la autorización GitHub es la cuenta asociada a la GitHub organización en la que se creó el repositorio.

## GitHub (mediante GitHub la aplicación) acción de origen: no se ha podido completar la conexión a un repositorio

Problema:

Dado que una conexión a un GitHub repositorio utiliza el AWS Conector GitHub, necesitas permisos de propietario de la organización o permisos de administrador del repositorio para crear la conexión.

Posibles correcciones: para obtener información sobre los niveles de permisos de un GitHub repositorio, consulta https://docs.github.com/en/free-pro-team@ latest/github teams/permission / setting-up-and-managing-organizations-and- -levels-for-an-organization.

## Error de Amazon S3: al CodePipeline ARN del rol < de servicio > se le deniega el acceso a S3 para el bucket de S3 < BucketName >

Problema:

Mientras está en curso, la CodeCommit acción CodePipeline comprueba que el depósito de artefactos de canalización existe. Si la acción no tiene permiso para comprobarse, se produce un `AccessDenied` error en Amazon S3 y aparece el siguiente mensaje de error CodePipeline:

CodePipeline Al rol de servicio «arn:aws:iam::*AccountID*: role/service -role/*RoleID*" se le deniega el acceso a S3 para el bucket de S3 "» *BucketName*

Los CloudTrail registros de la acción también registran el error. `AccessDenied`

Posibles soluciones: haga lo siguiente:

- Para la política asociada a su función de CodePipeline servicio, agréguela `s3:ListBucket` a la lista de acciones de su política. Para obtener instrucciones sobre cómo ver la política de funciones de servicio, consulteVer el ARN de la canalización y el ARN del rol de servicio (consola). Edite la declaración de política de su rol de servicio tal y como se detalla en Agrega permisos al rol de CodePipeline servicio.
- En el caso de la política basada en recursos adjunta al depósito de artefactos de Amazon S3 de su canalización, también denominada política de cubos de artefactos, añada una declaración que permita que su rol de servicio utilice el `s3:ListBucket` permiso. CodePipeline

  Para añadir su política al bucket de artefactos

  1. Siga los pasos que se indican en Ver el ARN de la canalización y el ARN del rol de servicio (consola) para elegir su bucket de artefactos en la página Configuración de la canalización y, a continuación, visualícelo en la consola de Amazon S3.
  2. Elija Permisos.
  3. En Bucket Policy (Política de bucket), elija Edit (Editar).
  4. En el campo de texto Política, introduzca una nueva política de bucket o edite la política existente, tal y como se muestra en el siguiente ejemplo. La política de bucket es un archivo JSON, por lo que debe introducir un JSON válido.

     En el siguiente ejemplo, se muestra una declaración de política de bucket para un bucket de artefactos en la que se muestra el ID de rol de ejemplo para el rol de servicio. *AROAEXAMPLEID*

     ```
     {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "s3:ListBucket",
     ```

```
    "Resource": "arn:aws:s3:::BucketName",
    "Condition": {
        "StringLike": {
            "aws:userid": "AROAEXAMPLEID:*"
        }
    }
}
```

En el siguiente ejemplo, se muestra la misma instrucción de política de bucket después de añadir el permiso.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringLike": {
                    "aws:userid": "AROAEXAMPLEID:*"
                }
            }
        },
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
```

```
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": false
                }
            }
        }
    ]
}
```

Para obtener más información, consulte los pasos en https://aws.amazon.com/blogs/security/writing-iam-policies-how-to-grant-access-to-an-amazon-s3-bucket/.

5. Seleccione Guardar.

Una vez aplicada la política editada, siga los pasos de Iniciar la canalización manualmente para volver a ejecutar manualmente la canalización.

## Las canalizaciones con Amazon S3, Amazon ECR o CodeCommit fuente ya no se inician automáticamente

Problema:

Tras realizar un cambio en los ajustes de configuración de una acción que utiliza reglas de eventos (EventBridgeo CloudWatch eventos) para la detección de cambios, es posible que la consola no detecte un cambio si los identificadores del desencadenante de origen son similares y tienen caracteres iniciales idénticos. Como la consola no crea la nueva regla de eventos, la canalización ya no se inicia automáticamente.

Un ejemplo de un cambio menor al final del nombre del parámetro CodeCommit sería cambiar el nombre de la CodeCommit sucursal `MyTestBranch-1` a`MyTestBranch-2`. Como el cambio se produce al final del nombre de la rama, es posible que la regla de eventos de la acción de origen no actualice ni cree una regla para la nueva configuración del origen.

Esto se aplica a las acciones de origen que utilizan eventos de CWE para la detección de cambios, de la siguiente manera:

| Acción de origen | Parámetros/identificadores de desencadenador (consola) |
| --- | --- |
| Amazon ECR | Nombre del repositorio<br>Etiquetas de la imagen |
| Amazon S3 | Bucket<br>Clave de objeto de S3 |
| CodeCommit | Nombre del repositorio<br>Nombre de ramificación |

Posibles soluciones:

Realice una de las siguientes acciones:

- Cambie los ajustes de CodeCommit/S3/ECR configuración para que los cambios se realicen en la parte inicial del valor del parámetro.

  Ejemplo: cambiar el nombre de ramificación de `release-branch` a `2nd-release-branch`. Evite cambios al final del nombre, por ejemplo `release-branch-2`.
- Cambie los ajustes CodeCommit/S3/ECR de configuración de cada canalización.

  Ejemplo: cambiar el nombre de ramificación de `myRepo/myBranch` a `myDeployRepo/myDeployBranch`. Evite cambios al final del nombre, por ejemplo `myRepo/myBranch2`.
- En lugar de la consola, utilice la CLI o CloudFormation cree y actualice las reglas de eventos de detección de cambios. Para obtener instrucciones sobre cómo crear reglas de eventos para una acción de origen de S3, consulte Conectarse a las acciones de origen de Amazon S3 que utilizan EventBridge y AWS CloudTrail. Para obtener instrucciones sobre cómo crear reglas de eventos para una acción de Amazon ECR, consulte Acciones y recursos de origen de Amazon ECR EventBridge. Para obtener instrucciones sobre cómo crear reglas de eventos para una CodeCommit acción, consulte. CodeCommit acciones de origen y EventBridge

Tras editar la configuración de las acciones en la consola, acepte los recursos de detección de cambios actualizados creados por la consola.

# Error de conexión al conectarse a GitHub: «Se ha producido un problema, asegúrate de que las cookies estén habilitadas en tu navegador» o «El propietario de una organización debe instalar la GitHub aplicación»

Problema:

Para crear la conexión para una acción de GitHub origen CodePipeline, debes ser el propietario de la GitHub organización. Para los repositorios que no pertenecen a una organización, debe ser el propietario del repositorio. Cuando alguien que no sea el propietario de la organización crea una conexión, se crea una solicitud para el propietario de la organización y se muestra uno de los siguientes errores:

Se ha producido un problema, asegúrese de que las cookies estén habilitadas en el navegador

OR

El propietario de la organización debe instalar la GitHub aplicación

Posibles soluciones: en el caso de los repositorios de una GitHub organización, el propietario de la organización debe crear la conexión con el GitHub repositorio. Para los repositorios que no pertenecen a una organización, debe ser el propietario del repositorio.

# Las canalizaciones con el modo de ejecución cambiado a MODO EN COLA o PARALELO fallan cuando se alcanza el límite de ejecución

Problema: el número máximo de ejecuciones simultáneas para una canalización en modo EN COLA es de 50 ejecuciones. Cuando se alcanza este límite, la canalización falla sin mostrar ningún mensaje de estado.

Posibles soluciones: al editar la definición de la canalización para el modo de ejecución, realice la edición por separado de otras acciones de edición.

Para obtener más información sobre el modo de ejecución EN COLA o PARALELO, consulte CodePipeline conceptos.

# Las canalizaciones en modo PARALELO tienen una definición de canalización desactualizada si se editan al cambiar al modo EN COLA o SUPERSEDED

Problema: para las canalizaciones en modo paralelo, al editar el modo de ejecución de la canalización en el modo EN COLA o SUPERSEDED, la definición de canalización para el modo PARALELO no se actualizará. La definición de canalización actualizada al actualizar el modo PARALELO no se utiliza en los modos REEMPLAZADO o EN COLA.

Posibles soluciones: para las canalizaciones en modo paralelo, al editar el modo de ejecución de la canalización en el modo EN COLA o SUPERSEDED, evite actualizar la definición de la canalización al mismo tiempo.

Para obtener más información sobre el modo de ejecución EN COLA o PARALELO, consulte CodePipeline conceptos.

# Las canalizaciones que cambien del modo PARALELO a otro mostrarán un modo de ejecución anterior

Problema: en el caso de las canalizaciones en modo PARALELO, al editar el modo de ejecución de la canalización en el modo EN COLA o SUPERSEDED, el estado de la canalización no mostrará el estado actualizado como PARALELO. Si la canalización cambia del modo PARALELO al modo EN COLA o SUPERSEDED, el estado de la canalización en modo SUPERSEDED o EN COLA será el último estado conocido en cualquiera de esos modos. Si la canalización nunca se ejecutó en ese modo anteriormente, el estado quedará vacío.

Posibles soluciones: para las canalizaciones en modo paralelo, al editar el modo de ejecución de la canalización en el modo EN COLA o SUPERSEDED, tenga en cuenta que la pantalla del modo de ejecución no mostrará el estado PARALELO.

Para obtener más información sobre el modo de ejecución EN COLA o PARALELO, consulte CodePipeline conceptos.

# Es posible que las canalizaciones con conexiones que utilicen el filtrado de desencadenadores por rutas de archivo no se inicien al crear la ramificación

Descripción: Para las canalizaciones con acciones de origen que utilizan conexiones, como una acción de BitBucket origen, puedes configurar un disparador con una configuración de Git que te permita filtrar por rutas de archivo para iniciar la canalización. En algunos casos, en el caso de las canalizaciones con activadores que se filtran según las rutas de los archivos, es posible que la canalización no se inicie cuando se cree por primera vez una rama con un filtro de rutas de archivos, ya que esto no permite que la AWS CodeConnections conexión resuelva los archivos que han cambiado. Si la configuración de Git para el activador está configurada para filtrar las rutas de los archivos, la canalización no se iniciará cuando se acabe de crear la rama con el filtro en el repositorio de origen. Para obtener más información sobre el filtrado de las rutas de los archivos, consulteAgregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código.

Resultado: Por ejemplo, las canalizaciones CodePipeline que tengan un filtro de rutas de archivos en una rama «B» no se activarán cuando se cree la rama «B». Si no hay filtros de rutas de archivo, la canalización seguirá iniciándose.

# Es posible que las canalizaciones con conexiones que utilicen el filtrado de desencadenadores por rutas de archivo no se inicien cuando se alcance el límite de archivos

Descripción: Para las canalizaciones con acciones de origen que utilizan conexiones, como una acción de BitBucket origen, puedes configurar un disparador con una configuración de Git que te permita filtrar por rutas de archivo para iniciar la canalización. CodePipeline recupera hasta los 100 primeros archivos; por lo tanto, cuando la configuración de Git del desencadenador está configurada para filtrar las rutas de los archivos, es posible que la canalización no se inicie si hay más de 100 archivos. Para obtener más información sobre el filtrado de las rutas de los archivos, consulteAgregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código.

Resultado: por ejemplo, si una diferencia contiene 150 archivos, CodePipeline examina los primeros 100 archivos (sin ningún orden en particular) para compararlos con el filtro de ruta de archivos

especificado. Si el archivo que coincide con el filtro de rutas de archivos no está entre los 100 archivos recuperados CodePipeline, no se invocará la canalización.

## CodeCommit o las revisiones del código fuente de S3 en modo paralelo podrían no coincidir con el EventBridge evento

Descripción: En el caso de las ejecuciones en proceso en modo PARALELO, una ejecución puede comenzar con el cambio más reciente, como la confirmación del CodeCommit repositorio, que puede no ser el mismo que el cambio del EventBridge evento. En algunos casos, cuando transcurre una fracción de segundo entre confirmaciones o etiquetas de imagen que inician la canalización, cuando se CodePipeline recibe el evento e inicia la ejecución, se ha insertado otra confirmación o etiqueta de imagen CodePipeline (por ejemplo, la CodeCommit acción), que clonará la confirmación de HEAD en ese momento.

Resultado: en el caso de las canalizaciones en modo paralelo con una fuente CodeCommit o S3, independientemente del cambio que haya desencadenado la ejecución de la canalización, la acción de origen siempre clonará el HEAD en el momento en que se inicie. Por ejemplo, para una canalización en modo PARALELO, se inserta una confirmación, que inicia la canalización para la ejecución 1, y la segunda ejecución de la canalización utiliza la segunda confirmación.

## La acción de despliegue de EC2 falla y muestra el mensaje de error **`No existe tal archivo`**

Descripción: una vez que la acción de implementación de EC2 descomprime los artefactos en el directorio de destino de las instancias, la acción ejecuta el script. Si el script está en el directorio de destino pero la acción no puede ejecutarlo, la acción fallará en esa instancia y las instancias restantes no se podrán implementar.

En los registros de una implementación donde el directorio de destino es `/home/ec2-user/deploy/` y la ruta del repositorio de origen es `myRepo/postScript.sh`, aparece un error similar a los siguientes.

- ```
  Instance i-0145a2d3f3EXAMPLE is FAILED on event AFTER_DEPLOY, message:
  ----------ERROR-------
  chmod: cannot access '/home/ec2-user/deploy/myRepo/postScript.sh': No such file or
   directory
  /var/lib/<path>/_script.sh: line 2: /home/ec2-user/deploy/myRepo/postScript.sh: No
   such file or directory
  ```
  ```

```
failed to run commands: exit status 127
```

- ```
  Executing commands on instances i-0145a2d3f3EXAMPLE, SSM command id <ID>, commands:
   chmod u+x /home/ec2-user/deploy/script.sh
  ----------ERROR-------: No such file or directory
  ```

Resultado: se produce un error de la acción de implementación en la canalización.

Posibles soluciones: para solucionar el problema, realice estos pasos.

1. Consulte los registros para comprobar qué instancia provocó el error del script.
2. Cambie el directorio (`cd`) al directorio de destino de su instancia. Pruebe la ejecución del script en la instancia.
3. En el repositorio de origen, edite el script para eliminar cualquier comentario o comando que pueda estar causando el problema.

## La acción EKS Deploy falla y aparece un mensaje de error **`inalcanzable para el clúster`**

Descripción: una vez ejecutada la acción de implementación de EKS, la acción falla con el mensaje de error `cluster unreachable`. El mensaje muestra un problema de acceso en el clúster debido a la falta de permisos. En función del tipo de archivo (gráfico de Helm o archivo de manifiesto de Kubernetes), el mensaje de error se muestra de la siguiente manera.

- Si se trata de una acción de implementación de EKS que utiliza un gráfico de Helm, se muestra un error similar al siguiente.

  ```
  error message:

  helm upgrade --install my-release test-chart --wait
  Error: Kubernetes cluster unreachable: the server has asked for the client to provide
   credentials
  ```

- Si se trata de una acción de implementación de EKS que utiliza archivos de manifiesto de Kubernetes, se muestra un error similar al siguiente.

  ```
  kubectl apply -f deployment.yaml
  ```
  ```

```
Error: error validating "deployment.yaml": error validating data: failed to download
 openapi: the server has asked for the client to provide credentials
```

Resultado: se produce un error de la acción de implementación en la canalización.

Posibles soluciones: si se utiliza una función existente, la función de CodePipeline servicio debe actualizarse con los permisos necesarios para utilizar la acción de despliegue de EKS. Además, para permitir que la función de CodePipeline servicio acceda a su clúster, debe añadir una entrada de acceso a su clúster y especificar la función de servicio para la entrada de acceso.

1. Compruebe que el rol CodePipeline de servicio tenga los permisos necesarios para la acción de despliegue de EKS. Puede consultar la referencia de los permisos en Permisos para las políticas de roles de servicio.
2. Agregue una entrada de acceso a su clúster y especifique la función de CodePipeline servicio para el acceso. Para ver un ejemplo, consulta Paso 4: Cree una entrada de acceso para la función CodePipeline de servicio.

## La canalización no se activa en todas las sucursales cuando varias acciones de origen hacen referencia al mismo repositorio

Problema: una canalización tiene varias acciones de origen que utilizan conexiones (como un Bitbucket o una GitLab conexión), y cada acción de origen apunta a una rama diferente del mismo repositorio. GitHub Solo una de las ramas activa la canalización al introducir cambios. La suscripción a webhook de la conexión se registra para la combinación de canalización y repositorio, no para cada sucursal. Como resultado, no se admiten varias acciones de origen dirigidas a distintas ramas del mismo repositorio dentro de una misma canalización.

Posibles soluciones: usa una canalización independiente para cada rama que desees activar de forma independiente.

## ¿Necesita ayuda con otro problema?

Pruebe estos otros recursos:

- Contacte con AWS Support.
- Plantee una pregunta en el foro de CodePipeline.

- Solicitar un aumento de cuota. Para obtener más información, consulte Cuotas en AWS CodePipeline.

> **Note**
>
> Puede tardar hasta dos semanas procesar las solicitudes de aumento de cuota.

# Seguridad en AWS CodePipeline

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de los centros de datos y las arquitecturas de red diseñados para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre AWS usted y usted. El https://aws.amazon.com/compliance/shared-responsibility-model/ describe estos conceptos como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la Nube de AWS. AWS también le proporciona servicios que puede utilizar de forma segura. Third-party los auditores prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los programas de AWS cumplimiento de los programas de . Para obtener más información sobre los programas de cumplimiento aplicables AWS CodePipeline, consulte AWS Servicios incluidos en el ámbito de aplicación por programa de conformidad y AWS servicios incluidos .
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos vigentes.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza CodePipeline. Los siguientes temas muestran cómo configurarlo CodePipeline para cumplir sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros AWS servicios que le ayudan a supervisar y proteger sus CodePipeline recursos.

Temas


- Protección de datos en AWS CodePipeline
- Gestión de identidad y acceso para AWS CodePipeline
- Inicio de sesión y supervisión CodePipeline
- Validación de conformidad para AWS CodePipeline
- Resiliencia en AWS CodePipeline
- Seguridad de la infraestructura en AWS CodePipeline
- Prácticas recomendadas de seguridad

# Protección de datos en AWS CodePipeline

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos en . Como se describe en este modelo, AWS es responsable de proteger la infraestructura global en la que se ejecutan todos los Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulte las Preguntas frecuentes sobre privacidad de datos y los .  Para obtener más información sobre la protección de datos en Europa, consulte el Centro del Reglamento General de Protección de Datos (RGPD).

Para fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Se utiliza SSL/TLS para comunicarse con AWS los recursos. Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte Cómo trabajar con CloudTrail senderos en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger la información confidencial almacenada en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-3 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabaja o Servicios de AWS utiliza la consola, la API o los SDK. AWS CLI AWS Cualquier dato que introduzca en etiquetas o campos de formato

libre utilizados para los nombres se pueden emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Las siguientes prácticas recomendadas sobre seguridad también evalúan la protección de datos en CodePipeline:

- Configurar el cifrado del lado del servidor para los artefactos almacenados en Amazon S3 para CodePipeline
- Uso AWS Secrets Manager para rastrear las contraseñas de las bases de datos o las claves de API de terceros

## Privacidad del tráfico entre redes

Amazon VPC es una Servicio de AWS que puede utilizar para lanzar AWS recursos en una red virtual (nube privada virtual) que usted defina. CodePipelinees compatible con los puntos de enlace de Amazon VPC con tecnología AWS PrivateLink, una AWS tecnología que facilita la comunicación privada entre el Servicios de AWS uso de una interfaz de red elástica con direcciones IP privadas. Esto significa que puede conectarse directamente a CodePipeline través de un punto final privado en su VPC, manteniendo todo el tráfico dentro de su VPC y de la red. AWS Anteriormente, las aplicaciones que se ejecutaban dentro de una VPC necesitaban acceso a Internet para conectarse a CodePipeline. Con una VPC, tiene control sobre los ajustes de red, como por ejemplo:

- Rango de direcciones IP,
- Subredes,
- Tablas de enrutamiento y
- Gateways de red

Para conectar su VPC CodePipeline, debe definir un punto final de VPC de interfaz para. CodePipeline Este tipo de punto de conexión le permite conectar su VPC a servicios de Servicios de AWS. El punto final proporciona una conectividad fiable y escalable CodePipeline sin necesidad de una puerta de enlace a Internet, una instancia de traducción de direcciones de red (NAT) ni una conexión VPN. Para obtener más información acerca de cómo configurar una VPC, consulte la Guía del usuario de VPC.

## Cifrado en reposo

Los datos entrantes CodePipeline se cifran en reposo mediante AWS KMS keys. Los artefactos de código se almacenan en un depósito de S3 propiedad del cliente y se cifran con la clave gestionada por el cliente Clave administrada de AWS o con una clave gestionada por el cliente. Para obtener más información, consulte Configurar el cifrado del lado del servidor para los artefactos almacenados en Amazon S3 para CodePipeline.

## Cifrado en tránsito

Todas las comunicaciones de servicio a servicio se cifran en tránsito mediante. SSL/TLS

## Administración de claves de cifrado

Si elige la opción predeterminada para cifrar los artefactos de código, CodePipeline utiliza la. Clave administrada de AWS No puede cambiarlo ni eliminarlo Clave administrada de AWS. Si utiliza una clave gestionada por el cliente AWS KMS para cifrar o descifrar los artefactos del depósito de S3, puede cambiar o rotar esta clave gestionada por el cliente según sea necesario.

> **Important**
>
> CodePipeline solo admite claves KMS simétricas. No utilice una clave de KMS asimétrica para cifrar los datos en el bucket de S3.

## Configurar el cifrado del lado del servidor para los artefactos almacenados en Amazon S3 para CodePipeline

Hay dos formas de configurar el cifrado del lado servidor para los artefactos de Amazon S3:

- CodePipeline crea un depósito de artefactos de S3 y lo establece de forma predeterminada Clave administrada de AWS al crear una canalización mediante el asistente Crear canalización. Clave administrada de AWS Se cifra junto con los datos del objeto y se gestiona mediante AWS.
- Puede crear y administrar su propia clave administrada por el cliente.

> **Important**
>
> CodePipeline solo admite claves KMS simétricas. No utilice una clave de KMS asimétrica para cifrar los datos en el bucket de S3.

Si utiliza la clave de S3 predeterminada, no podrá cambiar ni eliminar esta Clave administrada de AWS. Si utiliza una clave administrada por el cliente AWS KMS para cifrar o descifrar los artefactos del bucket de S3, puede cambiar o rotar esta clave administrada por el cliente según sea necesario.

Amazon S3 admite políticas de bucket que podrá usar si requiere cifrado del lado del servidor para todos los objetos almacenados en un bucket. Por ejemplo, la siguiente política de bucket deniega el permiso de carga del objeto (`s3:PutObject`) a todo el mundo si la solicitud no incluye el `x-amz-server-side-encryption` encabezado con el que se solicita el cifrado del lado del servidor. SSE-KMS

JSON

```
{
    "Version":"2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::codepipeline-us-west-2-89050EXAMPLE/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::codepipeline-us-west-2-89050EXAMPLE/*",
            "Condition": {
```

```
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

Para obtener más información sobre el cifrado del lado del servidor AWS KMS, consulte Protección de datos mediante el cifrado y Protección de datos mediante el Server-Side cifrado del lado del servidor con claves KMS almacenadas en (). AWS Key Management Service SSE-KMS

Para obtener más información al respecto AWS KMS, consulte la Guía para desarrolladores.AWS Key Management Service

Temas

- Vea su Clave administrada de AWS
- Configure el cifrado del lado del servidor para los buckets de S3 mediante CloudFormation o el AWS CLI

## Vea su Clave administrada de AWS

Cuando se usa el asistente Create Pipeline (Crear canalización) para crear la primera canalización, se crea un bucket de S3 automáticamente en la misma región en la que se creó la canalización. El bucket se utiliza para almacenar artefactos de canalización. Cuando se ejecuta una canalización, los artefactos se colocan y se recuperan en el bucket de S3. De forma predeterminada, CodePipeline utiliza el cifrado del lado del servidor con el AWS KMS uso Clave administrada de AWS de Amazon S3 (la `aws/s3` clave). Clave administrada de AWS Se crea y almacena en su cuenta. AWS Cuando se recuperan los artefactos del depósito de S3, se CodePipeline utiliza el mismo SSE-KMS proceso para descifrar el artefacto.

Para ver información sobre su Clave administrada de AWS

1. Inicie sesión en la AWS KMS consola Consola de administración de AWS y ábrala.
2. Si aparece una página de bienvenida, elija Empezar ahora.
3. En el panel de navegación, elija Claves administradas de AWS .

4. En Filtrar, elija la región de la canalización. Por ejemplo, si la canalización se creó en `us-east-2`, asegúrese de que el filtro esté establecido en Este de EE. UU. (Ohio).

   Para obtener más información sobre las regiones y los puntos de conexión disponibles CodePipeline, consulta los puntos de conexión y las AWS CodePipeline cuotas.

5. En la lista, elige la clave con el alias utilizado para tu canalización (de forma predeterminada, aws/s3). Se mostrará información básica acerca de la clave.

## Configure el cifrado del lado del servidor para los buckets de S3 mediante CloudFormation o el AWS CLI

Cuando utilice CloudFormation o AWS CLI para crear una canalización, debe configurar el cifrado del lado del servidor manualmente. Utilice la política de bucket de ejemplo anterior y, a continuación, cree su propia clave administrada por el cliente. También puede usar sus propias claves en lugar de la clave predeterminada de Clave administrada de AWS. Algunas de las razones para elegir su propia clave son las siguientes:

- Desea rotar la clave según una programación para satisfacer los requisitos de negocio o de seguridad de la organización.
- Desea crear una canalización que use los recursos asociados a otra cuenta de AWS . Esto requiere el uso de una clave administrada por el cliente. Para obtener más información, consulte Cree una canalización en la CodePipeline que se utilicen los recursos de otra AWS inscrita.

Las prácticas criptográficas recomendadas desaconsejan la reutilización generalizada de claves de cifrado. Es recomendable que rote la clave con regularidad. Para crear nuevo material criptográfico para sus AWS KMS claves, puede crear una clave gestionada por el cliente y, a continuación, cambiar las aplicaciones o los alias para utilizar la nueva clave gestionada por el cliente. También puede habilitar la rotación automática de claves para una clave administrada por el cliente existente.

Para rotar la clave gestionada por el cliente, consulte Rotación de claves.

> **Important**
>
> CodePipeline solo admite claves KMS simétricas. No utilice una clave de KMS asimétrica para cifrar los datos en el bucket de S3.

## Uso AWS Secrets Manager para rastrear las contraseñas de las bases de datos o las claves de API de terceros

Le recomendamos que lo utilice AWS Secrets Manager para rotar, administrar y recuperar las credenciales de la base de datos, las claves de API y otros datos secretos a lo largo de su ciclo de vida. Secrets Manager le permite reemplazar las credenciales codificadas en el código (incluidas contraseñas), con una llamada a la API de Secrets Manager para recuperar el secreto mediante programación. Para obtener más información, consulte ¿Qué es AWS Secrets Manager? en la Guía del usuario de AWS Secrets Manager.

En el caso de las canalizaciones en las que pasas parámetros que son secretos (como las credenciales de OAuth) a una CloudFormation plantilla, debes incluir referencias dinámicas en la plantilla que accedan a los secretos que has almacenado en Secrets Manager. Para ver patrones y ejemplos del ID de referencia, consulte Secrets Manager Secrets en la Guía del usuario de AWS CloudFormation . Para ver un ejemplo en el que se utilizan referencias dinámicas en un fragmento de plantilla para un GitHub webhook en una canalización, consulte Configuración de recursos de Webhook.

### Véase también

Los siguientes recursos relacionados pueden ayudarle mientras trabaja con la administración de secretos.

- Secrets Manager puede rotar automáticamente las credenciales de la base de datos, por ejemplo, para la rotación de secretos de Amazon RDS. Para obtener más información, consulte Rotación de AWS los secretos de Secrets Manager en la Guía del usuario de AWS Secrets Manager.
- Para consultar las instrucciones para agregar referencias dinámicas de Secrets Manager a las plantillas de CloudFormation , consulte https://aws.amazon.com/blogs/security/how-to-create-and-retrieve-secrets-managed-in-aws-secrets-manager-using-aws-cloudformation-template/.

# Gestión de identidad y acceso para AWS CodePipeline

AWS Identity and Access Management (IAM) es una herramienta Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. Los administradores de IAM controlan quién puede autenticarse (iniciar sesión) y quién puede autorizarse (tener permisos) para usar los recursos. CodePipeline La IAM es una Servicio de AWS opción que puede utilizar sin coste adicional.

Temas


- Público
- Autenticación con identidades
- Administración del acceso con políticas
- Cómo AWS CodePipeline funciona con IAM
- AWS CodePipeline ejemplos de políticas basadas en identidades
- AWS CodePipeline ejemplos de políticas basadas en recursos
- Resolución de problemas AWS CodePipeline identidad y acceso
- referencia de permisos
- Administre la función CodePipeline de servicio


## Público

La forma de usar AWS Identity and Access Management (IAM) varía según la función que desempeñes:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte Resolución de problemas AWS CodePipeline identidad y acceso).
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte Cómo AWS CodePipeline funciona con IAM).
- Administrador de IAM: escribe las políticas para administrar el acceso (consulte AWS CodePipeline ejemplos de políticas basadas en identidades).

## Autenticación con identidades

La autenticación es la forma en que inicias sesión AWS con tus credenciales de identidad. Debe autenticarse como usuario de Usuario raíz de la cuenta de AWS IAM o asumir una función de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de una fuente de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales. Google/Facebook Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In .

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

### Usuario raíz de la cuenta de AWS

Al crear una Cuenta de AWS, se comienza con una identidad de inicio de sesión denominada usuario Cuenta de AWS raíz, que tiene acceso completo a todos los Servicios de AWS recursos. Se recomiendaencarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver la lista completa de las tareas que requieren credenciales de usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

### Usuarios y grupos de IAM

Un usuario de IAM es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales de larga duración. Para obtener más información, consulte Exigir a los usuarios humanos que utilicen la federación con un proveedor de identidad para acceder AWS mediante credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

### Roles de IAM

Un Rol de IAM es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol cambiando de un rol de usuario a uno de IAM (consola) o llamando a una AWS CLI operación de AWS API. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuario federado, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

## Administración del acceso con políticas

AWS Para controlar el acceso, puede crear políticas y adjuntarlas a AWS identidades o recursos. Una política define los permisos cuando están asociados a una identidad o un recurso. AWS evalúa

estas políticas cuando un director hace una solicitud. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre los documentos de políticas de JSON, consulte Información general de políticas de JSON en la Guía del usuario de IAM.

Mediante las políticas, los administradores especifican quién tiene acceso a qué, definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a roles, que los usuarios pueden asumir posteriormente. Las políticas de IAM definen permisos independientemente del método que se utilice para realizar la operación.

### Identity-based políticas

Identity-based las políticas son documentos de política de permisos de JSON que se adjuntan a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Identity-based las políticas pueden ser políticas integradas (integradas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte Selección entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

### Resource-based políticas

Resource-based las políticas son documentos de políticas de JSON que se adjuntan a un recurso. Los ejemplos incluyen las Políticas de confianza de roles de IAM y las Políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Debe especificar una entidad principal en una política basada en recursos.

Resource-based las políticas son políticas en línea que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

### Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer los permisos máximos que conceden los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations .
- Políticas de control de recursos (RCP): definen los permisos máximos disponibles para los recursos de las cuentas. Para obtener más información, consulte Políticas de control de recursos (RCP) en la Guía del usuario de AWS Organizations .
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para obtener más información, consulte Políticas de sesión en la Guía del usuario de IAM.

## Cómo AWS CodePipeline funciona con IAM

Antes de utilizar IAM para gestionar el acceso CodePipeline, debe comprender las funciones de IAM disponibles para su uso. CodePipeline Para obtener una visión general de cómo CodePipeline y otras Servicios de AWS formas de trabajar con IAM, consulte Servicios de AWS Cómo funcionan con IAM en la Guía del usuario de IAM.

Temas

- CodePipeline políticas basadas en la identidad
- CodePipeline políticas basadas en recursos
- Autorización basada en etiquetas CodePipeline
- CodePipeline Funciones de IAM

### CodePipeline políticas basadas en la identidad

Con las políticas basadas en identidad de IAM, puede especificar las acciones permitidas o denegadas y los recursos, además de las condiciones en las que se permiten o deniegan las acciones. CodePipeline admite acciones, recursos y claves de condiciones específicos. Para obtener información sobre todos los elementos que utiliza en una política JSON, consulte Referencia de los elementos de las políticas JSON de IAM en la Guía del usuario de IAM.

Acciones

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Las acciones políticas CodePipeline utilizan el siguiente prefijo antes de la acción:`codepipeline:`.

Por ejemplo, para conceder a alguien permiso para ver las canalizaciones existentes en la cuenta, debe incluir la acción `codepipeline:GetPipeline` en su política. Las declaraciones de política deben incluir un `NotAction` elemento `Action` o. CodePipeline define su propio conjunto de acciones que describen las tareas que puede realizar con este servicio.

Para especificar varias acciones en una única instrucción, sepárelas con comas del siguiente modo:

```
"Action": [
      "codepipeline:action1",
      "codepipeline:action2"
```

Puede utilizar caracteres comodín para especificar varias acciones (*). Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Get`, incluya la siguiente acción:

```
"Action": "codepipeline:Get*"
```

Para obtener una lista de CodePipeline acciones, consulte las acciones definidas por AWS CodePipeline en la Guía del usuario de IAM.

Recursos

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). En el caso de las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

recursos y operaciones

En , el recurso principal es una canalización. En una política, se usa un nombre de recurso de Amazon (ARN) para identificar el recurso al que se aplica la política. es compatible con otros recursos que se pueden usar con el recurso principal, como etapas, acciones y acciones personalizadas. Estos elementos se denominan subrecursos. Los recursos y los subrecursos tienen nombres de recurso de Amazon (ARN) únicos asociados a ellos. Para obtener más información acerca de los ARN, consulte Nombres de recursos de Amazon (ARN) Servicio de AWS y espacios de nombres en. Referencia general de Amazon Web Services Para obtener el ARN de canalización asociado a tu canalización, puedes encontrar el ARN de la canalización en Configuración de la consola. Para obtener más información, consulte Ver el ARN de la canalización y el ARN del rol de servicio (consola).

| Tipo de recurso | Formato de ARN |
| --- | --- |
| Canalización | arn:aws:codepipeline::: *region account pipeline-name* |
| Etapa | arn:aws:codepipeline:::/*regionaccountpipeline-name stage-name* |
| Action | arn:aws:codepipeline:::/*regionaccountpipeline-name stage-name action-name* |
| Acción personalizada | arn:aws:codepipeline: ::tipo de acción:// /*regionaccountownercategoryproviderversion* |
| Todos los recursos | arn:aws:codepipeline:* |
| Todos los recursos de que pertenecen a la cuenta especificada en la región indicada | arn:aws:codepipeline*region*::*account*: * |

> **Note**
>
> La mayoría de los servicios AWS tratan los dos puntos (:)) o una barra diagonal (/) como el mismo carácter en los ARN. Sin embargo, utiliza una coincidencia exacta en los patrones de eventos y reglas. Se deben usar los caracteres de ARN correctos al crear patrones de eventos para que coincidan con la sintaxis de ARN de la canalización que desee usar.

En, hay llamadas a la API que admiten permisos a nivel de recurso. Resource-level los permisos indican si una llamada a la API puede especificar un ARN de recurso o si la llamada a la API solo puede especificar todos los recursos mediante el comodín. Consulte referencia de permisos para obtener una descripción detallada de los permisos a nivel de recursos y una lista de las llamadas a la CodePipeline API que admiten los permisos a nivel de recursos.

Por ejemplo, puede indicar una canalización específica (*myPipeline*) en su declaración utilizando su ARN de la siguiente manera:

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:myPipeline"
```

También puede especificar todas las canalizaciones que pertenezcan a una cuenta específica mediante el carácter comodín (*) del modo siguiente:

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:*"
```

Para especificar todos los recursos o cuando una acción de API no admite ARN, utilice el carácter comodín (*) en el elemento `Resource` de la siguiente manera:

```
"Resource": "*"
```

> **Note**
>
> Al crear políticas de IAM, siga los consejos de seguridad estándar de concesión de privilegios mínimos, es decir, conceder solo los permisos necesarios para realizar una tarea. Si una llamada a la API admite ARN, significa que admite permisos de nivel de recursos y que no es necesario utilizar el carácter comodín (*).

Algunas llamadas a la API aceptan varios recursos (por ejemplo,`GetPipeline`). Para especificar varios recursos en una única instrucción, separe sus ARN con comas, tal y como se indica a continuación:

```
"Resource": ["arn1", "arn2"]
```

proporciona un conjunto de operaciones para trabajar con los recursos. Para ver la lista de las operaciones disponibles, consulte referencia de permisos.

Claves de condición

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` especifica cuándo se ejecutan las instrucciones en función de criterios definidos. Puede crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

CodePipeline define su propio conjunto de claves de condición y también admite el uso de algunas claves de condición globales. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Todas las acciones de Amazon EC2 admiten las claves de condición `aws:RequestedRegion` y `ec2:Region`. Para obtener más información, consulte Ejemplo: restricción del acceso a una región específica.

Para ver una lista de claves de CodePipeline condición, consulte las claves de condición AWS CodePipeline en la Guía del usuario de IAM. Para saber con qué acciones y recursos puede utilizar una clave de condición, consulte Acciones definidas por AWS CodePipeline.

Ejemplos

Para ver ejemplos de políticas CodePipeline basadas en la identidad, consulte. AWS CodePipeline ejemplos de políticas basadas en identidades

## CodePipeline políticas basadas en recursos

CodePipeline no admite políticas basadas en recursos. Sin embargo, se proporciona un ejemplo de política basada en recursos para el servicio S3 relacionado con. CodePipeline

### Ejemplos

Para ver ejemplos de políticas CodePipeline basadas en recursos, consulte, AWS CodePipeline ejemplos de políticas basadas en recursos

## Autorización basada en etiquetas CodePipeline

Puede adjuntar etiquetas a CodePipeline los recursos o pasarles etiquetas en una solicitud CodePipeline. Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `codepipeline:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`. Para obtener más información acerca del etiquetado de recursos de CodePipeline , consulte Etiquetado de recursos.

Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Uso de etiquetas para controlar el acceso a los recursos CodePipeline.

## CodePipeline Funciones de IAM

Un rol de IAM es una entidad de tu AWS cuenta que tiene permisos específicos.

### Usar credenciales temporales con CodePipeline

Puede utilizar credenciales temporales para iniciar sesión con federación, asumir un rol de IAM o asumir un rol de acceso entre cuentas. Las credenciales de seguridad temporales se obtienen llamando a operaciones de AWS STS API como AssumeRoleo GetFederationToken.

CodePipeline admite el uso de credenciales temporales.

### Roles de servicio

CodePipeline permite que un servicio asuma una función de servicio en su nombre. Este rol permite que el servicio obtenga acceso a los recursos de otros servicios para completar una acción en su nombre. Los roles de servicio aparecen en su cuenta de IAM y son propiedad de la cuenta. Esto

significa que un administrador de IAM puede cambiar los permisos de este rol. Sin embargo, hacerlo podría deteriorar la funcionalidad del servicio.

CodePipeline apoya las funciones de servicio.

## AWS CodePipeline ejemplos de políticas basadas en identidades

De forma predeterminada, los usuarios y los roles de IAM no tienen permiso para crear, ver ni modificar recursos de CodePipeline . Tampoco pueden realizar tareas con la AWS API Consola de administración de AWS AWS CLI, o. Un administrador de IAM debe crear políticas de IAM que concedan permisos a los usuarios y a los roles para realizar operaciones de la API concretas en los recursos especificados que necesiten. El administrador debe adjuntar esas políticas a los usuarios o grupos de IAM que necesiten esos permisos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM con estos documentos de políticas JSON de ejemplo, consulte Creación de políticas en la pestaña JSON en la Guía del usuario de IAM.

Para obtener información sobre cómo crear una canalización que utilice recursos de otra cuenta y ver los ejemplos de políticas relacionadas, consulte Cree una canalización en la CodePipeline que se utilicen los recursos de otra AWS inscrita.

Temas

- Prácticas recomendadas sobre las políticas
- Visualización de recursos en la consola
- Cómo permitir a los usuarios consultar sus propios permisos
- Identity-based ejemplos de políticas (IAM)
- Uso de etiquetas para controlar el acceso a los recursos CodePipeline
- Permisos necesarios para usar la consola
- Permisos necesarios para ver los registros de procesamiento en la consola
- AWS políticas gestionadas para AWS CodePipeline
- Ejemplos de políticas administradas por el cliente

## Prácticas recomendadas sobre las políticas

Identity-based las políticas determinan si alguien puede crear CodePipeline recursos de tu cuenta, acceder a ellos o eliminarlos. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos para muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

## Visualización de recursos en la consola

La consola requiere el `ListRepositories` permiso para mostrar una lista de los repositorios de tu AWS cuenta en la AWS región en la que has iniciado sesión. La consola también incluye una función Go to resource (Ir al recurso) para realizar una búsqueda rápida de recursos sin distinción entre mayúsculas y minúsculas. Esta búsqueda se realiza en tu AWS cuenta de la AWS región en la que has iniciado sesión. Los siguientes recursos se muestran en los siguientes servicios:

- AWS CodeBuild: proyectos de compilación
- AWS CodeCommit: repositorios
- AWS CodeDeploy: aplicaciones
- AWS CodePipeline: canalizaciones

Para realizar esta búsqueda en los recursos de todos los servicios, debe contar con los siguientes permisos:

- AWS CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`
- CodePipeline: `ListPipelines`

Los resultados de los recursos de un servicio no se devuelven si no tiene permisos para ese servicio. Aunque tenga permisos para ver los recursos, algunos recursos no se devolverán si hay un permiso `Deny` explícito para ver esos recursos.

## Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la API AWS CLI o AWS .

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

## Identity-based ejemplos de políticas (IAM)

Puede asociar políticas a identidades de IAM. Por ejemplo, puede hacer lo siguiente:

- Adjunta una política de permisos a un usuario o grupo de tu cuenta: para conceder a un usuario permisos para ver las canalizaciones en la consola, puedes adjuntar una política de permisos a un usuario o grupo al que pertenezca el usuario.
- Adjuntar una política de permisos a un rol (conceder permisos para cuentas cruzadas): puede adjuntar una política de permisos basada en identidades a un rol de IAM para conceder permisos para cuentas cruzadas. Por ejemplo, el administrador de la cuenta A puede crear un rol para

conceder permisos multicuenta a otra cuenta (por ejemplo, la AWS cuenta B) o a una de las Servicio de AWS siguientes maneras:

1. El administrador de la CuentaA crea un rol de IAM y asocia una política de permisos a dicho rol, que concede permisos sobre los recursos de la CuentaA.
2. El administrador de la CuentaA asocia una política de confianza al rol que identifica la Cuenta B como la entidad principal que puede asumir el rol.
3. De este modo, el administrador de la cuenta B puede delegar los permisos para que asuman el rol en cualquier usuario de la cuenta B. De este modo, los usuarios de la cuenta B pueden crear recursos de la cuenta A. El director de la política de confianza también puede ser el Servicio de AWS principal si desea conceder Servicio de AWS permisos para asumir el rol.

Para obtener más información sobre el uso de IAM para delegar permisos, consulte Administración de accesos en la Guía del usuario de IAM.

A continuación, se muestra un ejemplo de una política de permisos que permite a un usuario habilitar y deshabilitar todas las transiciones entre las etapas de la canalización llamada `MyFirstPipeline` en la `us-west-2 region`:

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codepipeline:EnableStageTransition",
        "codepipeline:DisableStageTransition"
      ],
      "Resource" : [
        "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/*"
      ]
    }
  ]
}
```

En el siguiente ejemplo, se muestra una política en la cuenta 111222333444 que permite a los usuarios ver, pero no cambiar, la canalización nombrada `MyFirstPipeline` en la consola. Esta política se basa en la política administrada `AWSCodePipeline_ReadOnlyAccess`, pero como es específica de la canalización `MyFirstPipeline`, no puede usar la política administrada directamente. Si no desea restringir la política a una canalización específica, considere la posibilidad de usar una de las políticas administradas que crea y mantiene. Para obtener más información, consulte Uso de políticas administradas. Debe asociar esta política a un rol de IAM que cree para obtener acceso (por ejemplo, a un rol denominado `CrossAccountPipelineViewers`:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:GetPipelineExecution",
        "codepipeline:ListPipelineExecutions",
        "codepipeline:ListActionExecutions",
        "codepipeline:ListActionTypes",
        "codepipeline:ListPipelines",
        "codepipeline:ListTagsForResource",
        "iam:ListRoles",
        "s3:ListAllMyBuckets",
        "codecommit:ListRepositories",
        "codedeploy:ListApplications",
        "lambda:ListFunctions",
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline"
    },
    {
      "Action": [
        "codepipeline:GetPipeline",
        "codepipeline:GetPipelineState",
        "codepipeline:GetPipelineExecution",
```

```
        "codepipeline:ListPipelineExecutions",
        "codepipeline:ListActionExecutions",
        "codepipeline:ListActionTypes",
        "codepipeline:ListPipelines",
        "codepipeline:ListTagsForResource",
        "iam:ListRoles",
        "s3:GetBucketPolicy",
        "s3:GetObject",
        "s3:ListBucket",
        "codecommit:ListBranches",
        "codedeploy:GetApplication",
        "codedeploy:GetDeploymentGroup",
        "codedeploy:ListDeploymentGroups",
        "elasticbeanstalk:DescribeApplications",
        "elasticbeanstalk:DescribeEnvironments",
        "lambda:GetFunctionConfiguration",
        "opsworks:DescribeApps",
        "opsworks:DescribeLayers",
        "opsworks:DescribeStacks"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "CodeStarNotificationsReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:DescribeNotificationRule"
      ],
      "Resource": "*",
      "Condition": {
        "ArnLike": {
          "codestar-notifications:NotificationsForResource":
 "arn:aws:iam::*:role/Service*"
        }
      }
    }
  ]
}
```

Después de crear esta política, cree el rol de IAM en la cuenta 111222333444 y asocie la política a ese rol. En las relaciones de confianza del rol, debe agregar la AWS cuenta que asumirá este rol. El siguiente ejemplo muestra una política que permite a los usuarios de la *111111111111* AWS cuenta asumir las funciones definidas en la cuenta 111222333444:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

El siguiente ejemplo muestra una política creada en la *111111111111* AWS cuenta que permite a los usuarios asumir el rol nombrado *CrossAccountPipelineViewers* en la cuenta 111222333444:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource":
 "arn:aws:iam::111222333444:role/CrossAccountPipelineViewers"
        }
    ]
}
```

Puede crear políticas de IAM específicas para restringir las llamadas y los recursos a los que los usuarios de su cuenta tienen acceso y, a continuación, asociar esas políticas al usuario administrativo. Para obtener más información sobre cómo crear funciones de IAM y explorar ejemplos de declaraciones de política de IAM, consulte. Ejemplos de políticas administradas por el cliente

## Uso de etiquetas para controlar el acceso a los recursos CodePipeline

Las condiciones de las declaraciones de política de IAM forman parte de la sintaxis que se utiliza para especificar los permisos a los recursos que requieren las CodePipeline acciones. El uso de etiquetas en las condiciones es una manera de controlar el acceso a los recursos y las solicitudes. Para obtener información sobre el etiquetado de CodePipeline recursos, consulte. Etiquetado de recursos En este tema, se explica el control de acceso basado en etiquetas.

Al diseñar políticas de IAM, es posible que se establezcan permisos detallados mediante la concesión de acceso a recursos específicos. A medida que crezca la cantidad de recursos que administra, esta tarea será más complicada. El etiquetado de recursos y el uso de etiquetas en las condiciones de instrucción de política pueden facilitar esta tarea. Puede conceder acceso de forma masiva a cualquier recurso con una determinada etiqueta. A continuación, aplique repetidamente esta etiqueta a los recursos pertinentes durante la creación o después.

Las etiquetas se pueden asociar al recurso o pasarse dentro de la solicitud a los servicios que admiten etiquetado. En CodePipeline, los recursos pueden tener etiquetas y algunas acciones pueden incluirlas. Al crear una política de IAM, puede utilizar las claves de condición de etiqueta para controlar:

- Qué usuarios pueden realizar acciones en un recurso de canalización, basándose en las etiquetas que ya tiene.
- Qué etiquetas se pueden pasar en la solicitud de una acción.
- Si se pueden utilizar claves de etiqueta específicas en una solicitud.

Los operadores de condición de cadena le permiten desarrollar elementos `Condition` que restringen el acceso comparando una clave con el valor de una cadena. Puede agregar `IfExists` al final de cualquier nombre de operador de condición, salvo la condición Null. El objetivo es decir lo siguiente: “Si la clave de la política está presente en el contexto de la solicitud, se debe procesar la clave según se indica en la política. Si la clave no está presente, el elemento de condición se evalúa en verdadero". Por ejemplo, puede utilizar `StringEqualsIfExists` para restringir por condiciones las claves que podrían no estar presentes en otros tipos de recursos.

Para la sintaxis y semántica completas de claves de condición de etiquetas, consulte Control de acceso mediante etiquetas. Para obtener información adicional acerca de las claves de condición, consulte los siguientes recursos. Los ejemplos CodePipeline de políticas de esta sección se ajustan a la siguiente información sobre las claves de condición y la amplían con ejemplos de matices CodePipeline , como la anidación de recursos.

- Operadores de condición de cadena
- Servicios de AWS que funcionan con IAM
- Sintaxis de SCP
- Elementos de la política de JSON de IAM: condición
- fue: RequestTag/tag-key
- Claves de estado para CodePipeline

Los siguientes ejemplos muestran cómo especificar las condiciones de etiqueta en las políticas para los usuarios de CodePipeline .

Example 1: Limitar acciones en función de etiquetas en la solicitud

La política de usuarios `AWSCodePipeline_FullAccess` gestionados otorga a los usuarios permisos ilimitados para realizar cualquier CodePipeline acción en cualquier recurso.

La política siguiente limita esta capacidad y deniega los usuarios no autorizados el permiso para crear canalizaciones donde las etiquetas especificas se incluyen en la solicitud. Para ello, deniega la acción `CreatePipeline` si la solicitud especifica una etiqueta denominada `Project` con uno de los valores `ProjectA` o `ProjectB`. (La clave de condición `aws:RequestTag` se utiliza para controlar qué etiquetas se pueden pasar en una solicitud de IAM).

En el siguiente ejemplo, la intención de la política es denegar a los usuarios no autorizados el permiso para crear una canalización con los valores de etiqueta especificados. Sin embargo, la creación de una canalización requiere acceder a los recursos además de a la propia canalización (por ejemplo, a las acciones y etapas de la canalización). Como lo `'Resource'` especificado en la política es `'*'`, la política se evalúa en función de cada recurso que tenga un ARN y se crea cuando se crea la canalización. Estos recursos adicionales no tienen la clave de condición de etiqueta, por lo que la comprobación `StringEquals` da un error y no se concede al usuario la capacidad para lanzar cualquier tipo de instancia. Para solucionar este problema, utilice en su lugar el operador de condición `StringEqualsIfExists`. De esta forma, la prueba solo se realiza si la clave de condición existe.

Podría leer lo siguiente como: “Si el recurso que se está comprobando tiene una clave de condición `"RequestTag/Project"`, deberá permitirse la acción solo si el valor de clave comienza por `projectA`. Si el recurso que se está comprobando no tiene esta clave de condición, no deberá tenerse en cuenta”.

Además, la política impide que estos usuarios no autorizados manipulen los recursos utilizando la clave de condición `aws:TagKeys` para no permitir que las acciones de modificación de etiquetas incluyan estos mismos valores de etiqueta. El administrador de un cliente debe asociar esta política de IAM a los usuarios de IAM no autorizados, además de la política de usuario administrada.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:CreatePipeline",
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEqualsIfExists": {
          "aws:RequestTag/Project": ["ProjectA", "ProjectB"]
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
```

```
}
```

Example 2: Limitar acciones en función de etiquetas de recursos

La política de usuarios `AWSCodePipeline_FullAccess` administrados otorga a los usuarios permisos ilimitados para realizar cualquier CodePipeline acción en cualquier recurso.

La siguiente política limita esta capacidad y deniega a los usuarios no autorizados el permiso para realizar acciones en canalizaciones específicas del proyecto. Para ello, deniega algunas acciones si el recurso tiene una etiqueta denominada `Project` con el valor `ProjectA` o `ProjectB`. (La clave de condición `aws:ResourceTag` se utiliza para controlar el acceso a los recursos en función de las etiquetas que tengan los recursos). El administrador de un cliente debe asociar esta política de IAM a los usuarios de IAM no autorizados, además de la política de usuario administrada.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": ["ProjectA", "ProjectB"]
        }
      }
    }
  ]
}
```

Example 3: Permitir acciones en función de etiquetas en la solicitud

La política siguiente concede a los usuarios permiso para crear canalizaciones de desarrollo en CodePipeline.

Para ello, permite las acciones `CreatePipeline` y `TagResource` si la solicitud especifica una etiqueta denominada `Project` con el valor `ProjectA`. En otras palabras, la única clave de etiqueta que se puede especificar es`Project`, y su valor debe ser`ProjectA`.

(La clave de condición `aws:RequestTag` se utiliza para controlar qué etiquetas se pueden pasar en una solicitud de IAM). La condición `aws:TagKeys` garantiza la distinción entre mayúsculas y minúsculas de las claves de etiqueta. Esta política es útil para los usuarios o roles que no tienen asociada la política de usuario administrada `AWSCodePipeline_FullAccess`. La política administrada otorga a los usuarios permisos ilimitados para realizar cualquier CodePipeline acción en cualquier recurso.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:CreatePipeline",
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Project": "ProjectA"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

Example 4: Limitar acciones en función de etiquetas de recursos

La política de usuarios `AWSCodePipeline_FullAccess` administrados otorga a los usuarios permisos ilimitados para realizar cualquier CodePipeline acción en cualquier recurso.

La siguiente política limita esta capacidad y deniega a los usuarios no autorizados el permiso para realizar acciones en canalizaciones específicas del proyecto. Para ello, deniega algunas acciones si el recurso tiene una etiqueta denominada `Project` con el valor `ProjectA` o `ProjectB`.

Además, la política impide que estos usuarios no autorizados manipulen los recursos al utilizar la clave de condición `aws:TagKeys` que no permite que las acciones de modificación de etiquetas incluyan estos mismos valores de etiquetas ni eliminen por completo la etiqueta `Project`. El administrador de un cliente debe asociar esta política de IAM a los usuarios de IAM no autorizados, además de la política de usuario administrada.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

## Permisos necesarios para usar la consola

Para usar en la consola de , debe tener un conjunto mínimo de permisos de los siguientes servicios:

- AWS Identity and Access Management
- Amazon Simple Storage Service

Estos permisos le permiten describir otros AWS recursos de su AWS cuenta.

En función del resto de los servicios que integre en sus canalizaciones, es posible que necesite permisos de uno o más de los siguientes servicios:

- AWS CodeCommit
- AWS CodeBuild
- CloudFormation
- AWS CodeDeploy
- AWS Elastic Beanstalk
- AWS Lambda
- AWS OpsWorks

Si crea una política de IAM que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para los usuarios con esa política de IAM. Para asegurarse de que esos usuarios puedan seguir usando la consola, asocie también la política administrada `AWSCodePipeline_ReadOnlyAccess` al usuario, según se explica en AWS políticas gestionadas para AWS CodePipeline.

No es necesario que concedas permisos mínimos de consola a los usuarios que realicen llamadas a la API AWS CLI o a la API.

### Permisos necesarios para ver los registros de procesamiento en la consola

Para ver los registros de la acción de comandos en la CodePipeline consola, el rol de la consola debe tener permisos. Para ver los registros en la consola, añada los permisos `logs:GetLogEvents` al rol de la consola.

En la declaración de las políticas de roles de la consola, limite los permisos al nivel de la canalización como se muestra en el siguiente ejemplo.

```
{
    "Effect": "Allow",
    "Action": [
        "Action": "logs:GetLogEvents"
    ],
    "Resource": "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
}
```

# AWS políticas gestionadas para AWS CodePipeline

Una política AWS administrada es una política independiente creada y administrada por AWS. AWS Las políticas administradas están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y funciones.

Ten en cuenta que es posible que las políticas AWS administradas no otorguen permisos con privilegios mínimos para tus casos de uso específicos, ya que están disponibles para que los usen todos los AWS clientes. Se recomienda definir políticas administradas por el cliente específicas para sus casos de uso a fin de reducir aún más los permisos.

No puedes cambiar los permisos definidos en AWS las políticas administradas. Si AWS actualiza los permisos definidos en una política AWS administrada, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. AWS es más probable que actualice una política AWS administrada cuando Servicio de AWS se lance una nueva o cuando estén disponibles nuevas operaciones de API para los servicios existentes.

Para obtener más información, consulte Políticas administradas por AWS en la Guía del usuario de IAM.

> **Important**
>
> Las políticas administradas de `AWSCodePipelineFullAccess` y `AWSCodePipelineReadOnlyAccess` han sustituido. Utilice las políticas `AWSCodePipeline_FullAccess` y `AWSCodePipeline_ReadOnlyAccess`.

## AWS política gestionada: **`AWSCodePipeline_ FullAccess`**

Esta es una política que otorga acceso total a CodePipeline. Para ver el documento de política de JSON en la consola de IAM, consulte AWSCodePipeline_ FullAccess.

### Detalles de los permisos

Esta política incluye los siguientes permisos.

- `codepipeline`— Otorga permisos a. CodePipeline
- `chatbot`: concede permisos para que las entidades principales administren los recursos de Amazon Q Developer en las aplicaciones de chat.
- `cloudformation`— Otorga permisos para permitir a los directores administrar las pilas de recursos. CloudFormation
- `cloudtrail`— Otorga permisos que permiten a los directores gestionar el inicio de sesión de los recursos. CloudTrail
- `codebuild`— Otorga permisos para permitir a los directores acceder a los recursos del edificio. CodeBuild
- `codecommit`— Otorga permisos para permitir a los directores acceder a los recursos fuente en. CodeCommit
- `codedeploy`— Otorga permisos para permitir a los directores acceder a los recursos de implementación en. CodeDeploy
- `codestar-notifications`— Otorga permisos para permitir a los directores acceder a los recursos de las AWS CodeStar notificaciones.
- `ec2`— Otorga permisos para permitir que las implementaciones CodeCatalyst administren el balanceo de carga elástico en Amazon EC2.
- `ecr`: otorga permisos para permitir el acceso a los recursos de Amazon ECR.
- `elasticbeanstalk`: otorga permisos para permitir a las entidades principales acceder a los recursos de Elastic Beanstalk.
- `iam`: otorga permisos que permiten a las entidades principales administrar las funciones y las políticas en IAM.
- `lambda`: otorga permisos para permitir a las entidades principales administrar los recursos en Lambda.
- `events`— Otorga permisos que permiten a los directores administrar los recursos en Events. CloudWatch
- `opsworks`— Otorga permisos que permiten a los directores gestionar los recursos en ellos. AWS OpsWorks
- `s3`: otorga permisos para permitir a las entidades principales administrar los recursos en Amazon S3.

- `sns`: otorga permisos para que los directores administren los recursos de notificaciones en Amazon SNS.
- `states`— Otorga permisos que permiten a los directores ver las máquinas estatales en ellas. AWS Step Functions Una máquina de estados consiste en un conjunto de estados que administran las tareas y la transición entre estados.

Para ver la política, consulte AWSCodePipeline_ FullAccess.

AWS política gestionada: **`AWSCodePipeline_ ReadOnlyAccess`**

Se trata de una política que concede acceso de solo lectura a. CodePipeline Para ver el documento de política de JSON en la consola de IAM, consulte _. AWSCodePipeline ReadOnlyAccess

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `codepipeline`— Otorga permisos a las acciones en CodePipeline.
- `codestar-notifications`— Otorga permisos para permitir a los directores acceder a los recursos de las AWS CodeStar notificaciones.
- `s3`: otorga permisos para permitir a las entidades principales administrar los recursos en Amazon S3.
- `sns`: otorga permisos para que los directores administren los recursos de notificaciones en Amazon SNS.

Para ver la política, consulte AWSCodePipeline_ ReadOnlyAccess.

AWS política gestionada: **`AWSCodePipelineApproverAccess`**

Se trata de una política que otorga permiso para aprobar o rechazar una acción de aprobación manual. Para ver el documento de política de JSON en la consola de IAM, consulte AWSCodePipelineApproverAccess.

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `codepipeline`— Otorga permisos a las acciones en CodePipeline.

Para ver la política, consulte AWSCodePipelineApproverAccess.

AWS política gestionada: **`AWSCodePipelineCustomActionAccess`**

Se trata de una política que otorga permiso para crear acciones personalizadas en los recursos de Jenkins CodePipeline o integrarlos para crear o probar acciones. Para ver el documento de política de JSON en la consola de IAM, consulte. AWSCodePipelineCustomActionAccess

Detalles de los permisos

Esta política incluye los siguientes permisos.

- `codepipeline`— Otorga permisos a las acciones en CodePipeline.

Para ver la política, consulte AWSCodePipelineCustomActionAccess.

CodePipeline políticas y notificaciones gestionadas

CodePipeline admite las notificaciones, que pueden notificar a los usuarios los cambios importantes en las canalizaciones. Las políticas gestionadas CodePipeline incluyen declaraciones de políticas para la funcionalidad de notificación. Para obtener más información, consulte ¿Qué son las notificaciones?

Permisos relacionados con las notificaciones en políticas administradas de acceso total

Esta política gestionada concede permisos CodeCommit CodeBuild, CodePipeline CodeDeploy además de los servicios y AWS CodeStar notificaciones relacionados. La política también otorga los permisos que necesita para trabajar con otros servicios que se integran con sus canalizaciones, como Amazon S3, Elastic CloudTrail Beanstalk, Amazon EC2 y. CloudFormation Los usuarios que

tienen aplicada esta política administrada también pueden crear y administrar temas de Amazon SNS para notificaciones, suscribir usuarios a temas y cancelar dicha suscripción, mostrar los temas que se pueden elegir como destinos para las reglas de notificación y mostrar Amazon Q Developer en los clientes de aplicaciones de chat configurados para Slack.

La política administrada `AWSCodePipeline_FullAccess` incluye las siguientes instrucciones para permitir el acceso completo a las notificaciones.

```
    {
        "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": "*"
    }
```

Permisos relacionados con las notificaciones en políticas administradas de solo lectura

La política administrada `AWSCodePipeline_ReadOnlyAccess` incluye las siguientes instrucciones para permitir el acceso de solo lectura a las notificaciones. Los usuarios que apliquen esta política pueden consultar notificaciones de recursos, pero no pueden crearlas, administrarlas ni suscribirse a ellas.

```
   {
        "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": "*"
    }
```

Para obtener más información acerca de IAM y las notificaciones consulte Identity and Access Management para AWS CodeStar Notifications.

AWS CodePipeline actualizaciones de AWS políticas administradas

Vea los detalles sobre las actualizaciones de las políticas AWS administradas CodePipeline desde que este servicio comenzó a rastrear estos cambios. Para obtener alertas automáticas sobre los cambios realizados en esta página, suscríbase a la fuente RSS en la CodePipeline Página del historial de revisión de https://docs.aws.amazon.com/codepipeline/latest/userguide/history.html.

| Cambio | Descripción | Fecha |
| --- | --- | --- |
| AWSCodePipeline_ FullAccess — Actualizaciones de la política existente | CodePipeline agregó un permiso a esta política para `ListStacks` respaldarla CloudFormation. | 15 de marzo de 2024 |
| AWSCodePipeline_ FullAccess — Actualizaciones de la política existente | Esta política se ha actualizado para agregar permisos para Amazon Q Developer en aplicaciones de chat. Para obtener más información, consulte CodePipeline políticas y notificaciones gestionadas. | 21 de junio de 2023 |
| AWSCodePipeline_ FullAccess y AWSCodePipeline_ políticas ReadOnlyAccess gestionadas: actualizaciones de la política existente | CodePipeline agregó un permiso a estas políticas para admitir un tipo de notificación adicional utilizando Amazon Q Developer en aplicaciones de | 16 de mayo de 2023 |

| Cambio | Descripción | Fecha |
| --- | --- | --- |
| | chat,`chatbot:ListMicrosoftTeamsChannelConfigurations` . | |
| AWSCodePipelineFullAccess: obsoleta | Esta política ha sido reemplazada por `AWSCodePipeline_FullAccess` .<br><br>Después del 17 de noviembre de 2022, esta política no se podrá adjuntar a ningún usuario, grupo o función nuevos. Para obtener más información, consulte AWS políticas gestionadas para AWS CodePipeline. | 17 de noviembre de 2022 |
| AWSCodePipelineReadOnlyAccess: obsoleta | Esta política ha sido reemplazada por `AWSCodePipeline_ReadOnlyAccess` .<br><br>Después del 17 de noviembre de 2022, esta política no se podrá adjuntar a ningún usuario, grupo o función nuevos. Para obtener más información, consulte AWS políticas gestionadas para AWS CodePipeline. | 17 de noviembre de 2022 |
| CodePipeline comenzó a rastrear los cambios | CodePipeline comenzó a realizar un seguimiento de los cambios de sus políticas AWS gestionadas. | 12 de marzo de 2021 |

## Ejemplos de políticas administradas por el cliente

En esta sección, encontrará ejemplos de políticas de usuario que conceden permisos para diversas acciones de . Estas políticas funcionan cuando utilizas la API, AWS los SDK o el AWS CLI. Cuando se utiliza la consola, debe conceder permisos adicionales específicos a la consola. Para obtener más información, consulte Permisos necesarios para usar la consola.

> **Note**
>
> Todos los ejemplos utilizan la región Oeste de EE. UU. (Oregón) (`us-west-2`) y contienen identificadores de cuenta ficticios.

Ejemplos

- Ejemplo 1: Conceder permisos para obtener el estado de una canalización
- Ejemplo 2: Conceder permisos para activar y desactivar las transiciones entre etapas
- Ejemplo 3: Conceder permisos para obtener una lista de todos los tipos de acciones disponibles
- Ejemplo 4: Conceder permisos para autorizar o rechazar acciones de aprobación manual
- Ejemplo 5: Conceder permisos para sondear trabajos en una acción personalizada
- Ejemplo 6: Adjunte o edite una política para la integración de Jenkins con AWS CodePipeline
- Ejemplo 7: Configurar el acceso entre cuentas en una canalización

### Ejemplo 1: Conceder permisos para obtener el estado de una canalización

En el siguiente ejemplo, se conceden permisos para obtener el estado de la canalización llamada `MyFirstPipeline`:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:GetPipelineState"
```

```
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline"
        }
    ]
}
```

Ejemplo 2: Conceder permisos para activar y desactivar las transiciones entre etapas

En el siguiente ejemplo, se conceden permisos para deshabilitar y habilitar las transiciones entre todas las etapas de la canalización llamada `MyFirstPipeline`:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:DisableStageTransition",
                "codepipeline:EnableStageTransition"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline/*"
        }
    ]
}
```

Para que el usuario pueda deshabilitar o habilitar transiciones en una sola etapa de la canalización, debe indicar la etapa. Por ejemplo, para que el usuario puede habilitar o deshabilitar transiciones en una etapa llamada `Staging` de una canalización denominada `MyFirstPipeline`:

```
"Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/Staging"
```

Ejemplo 3: Conceder permisos para obtener una lista de todos los tipos de acciones disponibles

En el siguiente ejemplo, se conceden permisos para obtener una lista de todos los tipos de acción disponibles para las canalizaciones de la región `us-west-2`:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:ListActionTypes"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:actiontype:*"
        }
    ]
}
```

Ejemplo 4: Conceder permisos para autorizar o rechazar acciones de aprobación manual

En el siguiente ejemplo, se conceden permisos para autorizar o rechazar acciones de aprobación manual en una etapa llamada `Staging` de una canalización denominada `MyFirstPipeline`:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PutApprovalResult"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline/Staging/*"
        }
    ]
}
```

Ejemplo 5: Conceder permisos para sondear trabajos en una acción personalizada

En el siguiente ejemplo, se conceden permisos para sondear trabajos en la acción personalizada llamada `TestProvider`, que es un tipo de acción `Test` en su primera versión, en todas las canalizaciones:

> **Note**
>
> Puede que el proceso de trabajo de una acción personalizada se haya configurado con otra cuenta de AWS o que necesite un rol de IAM específico para que funcione.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PollForJobs"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111222333444:actionType:Custom/Test/TestProvider/1"
            ]
        }
    ]
}
```

Ejemplo 6: Adjunte o edite una política para la integración de Jenkins con AWS CodePipeline

Si configura una canalización para usar Jenkins para compilar o probar, cree una identidad independiente para esa integración y adjunte una política de IAM que tenga los permisos mínimos necesarios para la integración entre Jenkins y. Esta política es la misma que la política administrada `AWSCodePipelineCustomActionAccess`. En el siguiente ejemplo, se muestra una política para la integración de Jenkins:

JSON

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

    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:AcknowledgeJob",
                "codepipeline:GetJobDetails",
                "codepipeline:PollForJobs",
                "codepipeline:PutJobFailureResult",
                "codepipeline:PutJobSuccessResult"
            ],
            "Resource": "*"
        }
    ]
}
```

Ejemplo 7: Configurar el acceso entre cuentas en una canalización

Puede configurar el acceso a canalizaciones para los usuarios y los grupos de otra cuenta de AWS . La forma recomendada es crear un rol en la cuenta donde se creó la canalización. El rol debería permitir a los usuarios de la otra AWS cuenta asumir ese rol y acceder a la canalización. Para obtener más información, consulte Tutorial: Cross-Account Acceso mediante roles.

El siguiente ejemplo muestra una política en la cuenta 80398EXAMPLE que permite a los usuarios ver, pero no cambiar, la canalización nombrada `MyFirstPipeline` en la consola. Esta política se basa en la política administrada `AWSCodePipeline_ReadOnlyAccess`, pero como es específica de la canalización `MyFirstPipeline`, no puede usar la política administrada directamente. Si no desea restringir la política a una canalización específica, considere la posibilidad de usar una de las políticas administradas que crea y mantiene. Para obtener más información, consulte Uso de políticas administradas. Debe asociar esta política a un rol de IAM que cree para obtener acceso (por ejemplo, a un rol denominado `CrossAccountPipelineViewers`:

JSON

```
{
```

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

    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:GetPipeline",
                "codepipeline:GetPipelineState",
                "codepipeline:ListActionTypes",
                "codepipeline:ListPipelines",
                "iam:ListRoles",
                "s3:GetBucketPolicy",
                "s3:GetObject",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "codedeploy:GetApplication",
                "codedeploy:GetDeploymentGroup",
                "codedeploy:ListApplications",
                "codedeploy:ListDeploymentGroups",
                "elasticbeanstalk:DescribeApplications",
                "elasticbeanstalk:DescribeEnvironments",
                "lambda:GetFunctionConfiguration",
                "lambda:ListFunctions"
            ],
            "Resource": "arn:aws:codepipeline:us-
east-2:111122223333:MyFirstPipeline"
        }
    ]
}
```

Después de crear esta política, cree el rol de IAM en la cuenta 80398EXAMPLE y asocie la política a ese rol. En las relaciones de confianza del rol, debe agregar la AWS cuenta que asume este rol.

El siguiente ejemplo muestra una política creada en la *111111111111* AWS cuenta que permite a los usuarios asumir el rol nombrado `CrossAccountPipelineViewers` en la cuenta 80398EXAMPLE:

JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource":
 "arn:aws:iam::111122223333:role/CrossAccountPipelineViewers"
        }
    ]
}
```

## AWS CodePipeline ejemplos de políticas basadas en recursos

Otros servicios, como Amazon S3, también admiten políticas de permisos basadas en recursos. Por ejemplo, puede asociar una política a un bucket de S3 para administrar los permisos de acceso a dicho bucket. Aunque CodePipeline no admite políticas basadas en recursos, sí almacena artefactos para su uso en canalizaciones en buckets S3 versionados.

Example Crear una política para que un bucket de S3 se utilice como almacén de artefactos CodePipeline

Puedes usar cualquier bucket de S3 versionado como almacén de artefactos. CodePipeline Si crea su primera canalización con el asistente para crear canalizaciones, este bucket de S3 se crea de forma automática para garantizar que todos los objetos subidos al almacén de artefactos estén cifrados y las conexiones con el bucket sean seguras. Se recomienda que, si usted crea su propio bucket de S3, añada la siguiente política o sus elementos al bucket. En esta política, el ARN del bucket de S3 es `codepipeline-us-east-2-1234567890`. Sustituya este ARN por el de su bucket de S3:

JSON

```
{
    "Version":"2012-10-17",
    "Id": "SSEAndSSLPolicy",
    "Statement": [
        {
            "Sid": "DenyUnEncryptedObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
```

```
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        },
        {
            "Sid": "DenyInsecureConnections",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": false
                }
            }
        }
    ]
}
```

## Resolución de problemas AWS CodePipeline identidad y acceso

Utilice la siguiente información como ayuda para diagnosticar y solucionar los problemas más comunes que pueden surgir al trabajar con CodePipeline una IAM.

Temas

- No estoy autorizado a realizar ninguna acción en CodePipeline
- No estoy autorizado a realizar lo siguiente: PassRole
- Soy administrador y quiero permitir que otras personas accedan CodePipeline
- Quiero permitir que personas ajenas a mi AWS cuenta para acceder a mis CodePipeline recursos

### No estoy autorizado a realizar ninguna acción en CodePipeline

Si Consola de administración de AWS le indica que no está autorizado a realizar una acción, debe ponerse en contacto con su administrador para obtener ayuda. Su administrador es la persona que le facilitó su nombre de usuario y contraseña.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM de `mateojackson` intenta utilizar la consola para ver detalles sobre una canalización, pero no tiene permisos `codepipeline:GetPipeline`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 codepipeline:GetPipeline on resource: my-pipeline
```

En este caso, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso `my-pipeline` mediante la acción `codepipeline:GetPipeline`.

## No estoy autorizado a realizar lo siguiente: PassRole

Si recibe un error que indica que no está autorizado para llevar a cabo la acción `iam:PassRole`, debe ponerse en contacto con su administrador para recibir ayuda. Su administrador es la persona que le facilitó su nombre de usuario y contraseña. Pida a la persona que actualice sus políticas de forma que pueda transferir un rol a CodePipeline.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio, en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir la función al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta usar la consola para realizar una acción en CodePipeline. Sin embargo, la acción requiere que el servicio tenga permisos otorgados por un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, Mary pide a su administrador que actualice sus políticas para que pueda realizar la acción `iam:PassRole`.

## Soy administrador y quiero permitir que otras personas accedan CodePipeline

Para permitir el acceso de otras personas CodePipeline, debes conceder permiso a las personas o aplicaciones que necesitan acceso. Si usa AWS IAM Identity Center para administrar las personas y las aplicaciones, debe asignar conjuntos de permisos a los usuarios o grupos para definir su nivel de acceso. Los conjuntos de permisos crean políticas de IAM y las asignan a los roles de IAM asociados a la persona o aplicación de forma automática. Para obtener más información, consulte la sección Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .

Si no utiliza IAM Identity Center, debe crear entidades de IAM (usuarios o roles) para las personas o aplicaciones que necesitan acceso. A continuación, debe asociar una política a la entidad que le conceda los permisos correctos en CodePipeline. Una vez concedidos los permisos, proporcione las credenciales al usuario o al desarrollador de la aplicación. Utilizarán esas credenciales para acceder a AWS. Para obtener más información sobre la creación de usuarios, grupos, políticas y permisos de IAM, consulte Identidades de IAM y Políticas y permisos en IAM en la Guía del usuario de IAM.

### Quiero permitir que personas ajenas a mi AWS cuenta para acceder a mis CodePipeline recursos

Se puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Se puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si CodePipeline es compatible con estas funciones, consulteCómo AWS CodePipeline funciona con IAM.
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro usuario de su propiedad Cuenta de AWS en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

## referencia de permisos

Utilice la siguiente tabla como referencia cuando configure políticas de control de acceso y de escritura que pueda asociar a una identidad de IAM (políticas basadas en identidad). En la tabla figuran las operaciones de las API de y las acciones correspondientes para las que puede conceder

permisos para realizar la acción. En el caso de las operaciones que admiten permisos a nivel de recurso, en la tabla se indica el AWS recurso para el que puede conceder los permisos. Las acciones se especifican en el campo `Action` de la política.

Resource-level los permisos son aquellos que permiten especificar los recursos en los que los usuarios pueden realizar acciones. AWS CodePipeline proporciona compatibilidad parcial con los permisos a nivel de recursos. Esto significa que, para algunas llamadas a la AWS CodePipeline API, puedes controlar cuándo se permite a los usuarios usar esas acciones en función de las condiciones que deben cumplirse o qué recursos pueden usar los usuarios. Por ejemplo, puede conceder a los usuarios permiso para generar un listado con información de ejecución de canalizaciones, pero solo para una o varias canalizaciones específicas.

> **Note**
>
> En la columna Recursos, se muestra el recurso necesario para las llamadas a las API que admiten los permisos de nivel de recurso. En el caso de las llamadas a las API que no admiten los permisos de nivel de recurso, puede conceder permisos a los usuarios para que las utilicen, pero tendrá que usar un carácter comodín (*) en el elemento Resource de la instrucción de la política.

Operaciones de API y permisos necesarios para acciones de

AcknowledgeJob

Acción: `codepipeline:AcknowledgeJob`

Se necesita para ver información sobre un trabajo específico y si el proceso de trabajo lo ha recibido. Solo se usa en las acciones personalizadas.

Recursos: solo admite un carácter comodín (*) en el elemento `Resource` de la política.

AcknowledgeThirdPartyJob

Acción: `codepipeline:AcknowledgeThirdPartyJob`

Se necesita para confirmar si el proceso de trabajo ha recibido un trabajo concreto. Solo se usa en las acciones de los socios.

Recursos: solo admite un carácter comodín (*) en el elemento `Resource` de la política.

### CreateCustomActionType

Acción: `codepipeline:CreateCustomActionType`

Necesaria para crear una nueva acción personalizada que se pueda utilizar en todas las canalizaciones asociadas a la AWS cuenta. Solo se usa en las acciones personalizadas.

Recursos:

Tipo de acción

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

### CreatePipeline

Acción: `codepipeline:CreatePipeline`

Se necesita para crear una canalización.

Recursos:

Canalización

`arn:aws:codepipeline:region:account:pipeline-name`

### DeleteCustomActionType

Acción: `codepipeline:DeleteCustomActionType`

Se necesita para marcar una acción personalizada como eliminada. `PollForJobs` para la acción personalizada produce un error una vez marcada la acción para su eliminación. Solo se usa en las acciones personalizadas.

Recursos:

Tipo de acción

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

### DeletePipeline

Acción: `codepipeline:DeletePipeline`

Se necesita para eliminar una canalización.

Recursos:

Canalización

`arn:aws:codepipeline:region:account:pipeline-name`

DeleteWebhook

Acción: `codepipeline:DeleteWebhook`

Se necesita para eliminar un webhook.

Recursos:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

DeregisterWebhookWithThirdParty

Acción: `codepipeline:DeregisterWebhookWithThirdParty`

Antes de eliminar un webhook, es necesario eliminar la conexión entre el webhook creado por él CodePipeline y la herramienta externa con los eventos que se van a detectar. Actualmente, solo se admiten los webhooks que se dirigen a un tipo de acción de. GitHub

Recursos:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

DisableStageTransition

Acción: `codepipeline:DisableStageTransition`

Se necesita para impedir que los artefactos de una canalización vayan a la siguiente etapa de la canalización.

Recursos:

Canalización

`arn:aws:codepipeline:region:account:pipeline-name`

EnableStageTransition

Acción: `codepipeline:EnableStageTransition`

Se necesita para que los artefactos de una canalización vayan a la siguiente etapa de la canalización.

Recursos:

Canalización

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

GetJobDetails

Acción: `codepipeline:GetJobDetails`

Se necesita para recuperar información sobre un trabajo. Solo se usa en las acciones personalizadas.

Recursos: no se requieren recursos.

GetPipeline

Acción: `codepipeline:GetPipeline`

Se necesita para recuperar la estructura, las etapas, las acciones y los metadatos de una canalización, incluido el ARN de canalización.

Recursos:

Canalización

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

GetPipelineExecution

Acción: `codepipeline:GetPipelineExecution`

Se necesita para recuperar información sobre la ejecución de una canalización, como los detalles de los artefactos, el ID de ejecución de la canalización y el nombre, la versión y el estado de la canalización.

Recursos:

Canalización

`arn:aws:codepipeline:region:account:pipeline-name`

GetPipelineState

Acción: `codepipeline:GetPipelineState`

Se necesita para recuperar información sobre el estado de una canalización, como las etapas y las acciones.

Recursos:

Canalización

`arn:aws:codepipeline:region:account:pipeline-name`

GetThirdPartyJobDetails

Acción: `codepipeline:GetThirdPartyJobDetails`

Se necesita para solicitar los detalles de un trabajo en una acción de terceros. Solo se usa en las acciones de los socios.

Recursos: solo admite un carácter comodín (*) en el elemento `Resource` de la política.

ListActionTypes

Acción: `codepipeline:ListActionTypes`

Necesario para generar un resumen de todos los tipos de CodePipeline acciones asociados a tu cuenta.

Recursos:

Tipo de acción

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versior`

ListPipelineExecutions

Acción: `codepipeline:ListPipelineExecutions`

Se requiere para generar un resumen de las ejecuciones más recientes para una canalización.

Recursos:

Canalización

`arn:aws:codepipeline:region:account:pipeline-name`

ListPipelines

Acción: `codepipeline:ListPipelines`

Se necesita para generar un resumen de todas las canalizaciones asociadas a su cuenta.

Recursos:

ARN de canalización con comodín (no se admiten permisos de nivel de recursos a nivel de nombre de canalización)

`arn:aws:codepipeline:region:account:*`

ListTagsForResource

Acción: `codepipeline:ListTagsForResource`

Se necesita para obtener la lista de las etiquetas de un recurso específico.

Recursos:

Tipo de acción

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versio`

Canalización

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

ListWebhooks

Acción: `codepipeline:ListWebhooks`

Se necesita para mostrar todos los webhooks en la cuenta de esa región.

Recursos:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

PollForJobs

Acciones: `codepipeline:PollForJobs`

Necesario para recuperar información sobre cualquier puesto de trabajo con el CodePipeline que poder actuar.

Recursos:

Tipo de acción

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versio`

PollForThirdPartyJobs

Acción: `codepipeline:PollForThirdPartyJobs`

Se necesita para determinar si hay trabajos de terceros en los que un proceso de trabajo deba actuar. Solo se usa en las acciones de los socios.

Recursos: solo admite un carácter comodín (*) en el elemento `Resource` de la política.

PutActionRevision

Acción: `codepipeline:PutActionRevision`

Necesario para informar a una fuente CodePipeline sobre nuevas revisiones.

Recursos:

Action

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

PutApprovalResult

Acción: `codepipeline:PutApprovalResult`

Se necesita para notificar la respuesta sobre una solicitud de aprobación manual a CodePipeline. Las respuestas válidas son `Approved` y `Rejected`.

Recursos:

Action

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

> **Note**
>
> Esta llamada al API admite parcialmente permisos de nivel de recurso. Sin embargo, es posible que encuentre un error si utiliza el generador de políticas o la consola de IAM para crear políticas con `"codepipeline:PutApprovalResult"` que especifiquen el ARN de un recurso. Si encuentra un error, puede utilizar la pestaña JSON en la consola de IAM o la CLI para crear una política.

PutJobFailureResult

Acción: `codepipeline:PutJobFailureResult`

Se necesita para notificar un error en un trabajo y que un proceso de trabajo lo ha devuelto a la canalización. Solo se usa en las acciones personalizadas.

Recursos: solo admite un carácter comodín (*) en el elemento `Resource` de la política.

PutJobSuccessResult

Acción: `codepipeline:PutJobSuccessResult`

Se necesita para notificar que un proceso de trabajo ha devuelto un trabajo a la canalización correctamente. Solo se usa en las acciones personalizadas.

Recursos: solo admite un carácter comodín (*) en el elemento `Resource` de la política.

PutThirdPartyJobFailureResult

Acción: `codepipeline:PutThirdPartyJobFailureResult`

Se necesita para notificar un error en un trabajo de terceros que un proceso de trabajo ha devuelto a la canalización. Solo se usa en las acciones de los socios.

Recursos: solo admite un carácter comodín (*) en el elemento `Resource` de la política.

PutThirdPartyJobSuccessResult

Acción: `codepipeline:PutThirdPartyJobSuccessResult`

Se necesita para notificar que un proceso de trabajo ha devuelto un trabajo de terceros a la canalización correctamente. Solo se usa en las acciones de los socios.

Recursos: solo admite un carácter comodín (*) en el elemento `Resource` de la política.

PutWebhook

Acción: `codepipeline:PutWebhook`

Se necesita para crear un webhook.

Recursos:

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RegisterWebhookWithThirdParty

Acción: `codepipeline:RegisterWebhookWithThirdParty`

Recursos:

Una vez creado un webhook, se necesita para configurar terceros compatibles para llamar a la URL de webhook generada.

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RetryStageExecution

Acción: `codepipeline:RetryStageExecution`

Se necesita para reanudar la ejecución en la canalización reintentando las últimas acciones erróneas de una etapa.

Recursos:

Canalización

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

StartPipelineExecution

Acción: `codepipeline:StartPipelineExecution`

Se necesita para iniciar la canalización especificada (en concreto, para iniciar el procesamiento de la última confirmación en la ubicación de origen especificada como parte de la canalización).

Recursos:

Canalización

`arn:aws:codepipeline:region:account:pipeline-name`

TagResource

Acción: `codepipeline:TagResource`

Se necesita para etiquetar el recurso especificado.

Recursos:

Tipo de acción

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

Canalización

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UntagResource

Acción: `codepipeline:UntagResource`

Se necesita para eliminar las etiquetas del recurso especificado.

Recursos:

Tipo de acción

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

Canalización

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UpdatePipeline

Acción: `codepipeline:UpdatePipeline`

Se necesita para actualizar una canalización específica con cambios o modificaciones en su estructura.

Recursos:

Canalización

`arn:aws:codepipeline:region:account:pipeline-name`

## Administre la función CodePipeline de servicio

La función de servicio se configura con una o más políticas que controlan el acceso a los AWS recursos utilizados por la canalización. Puede que desee adjuntar más políticas a esta función, editar la política asociada a la función o configurar políticas para otras funciones de servicio en AWS. También puede asociar una política a un rol al configurar el acceso entre cuentas en su canalización.

**Important**

Si se modifica la instrucción de una política o se asocia otra política al rol, es posible que las canalizaciones dejen de funcionar. Asegúrese de conocer bien las consecuencias antes de modificar el rol de servicio de . Asegúrese de probar las canalizaciones después de implementar cualquier cambio en el rol de servicio.

**Note**

En la consola, los roles de servicio creados antes de septiembre de 2018 se crean con el nombre `oneClick_AWS-CodePipeline-Service_ID-Number`.
Los roles de servicio creados después de septiembre de 2018 usan el formato de nombre de rol de servicio `AWSCodePipelineServiceRole-Region-Pipeline_Name`. Por ejemplo,

en el caso de una canalización denominada `MyFirstPipeline` en `eu-west-2`, la consola asigna un nombre al rol y a la política `AWSCodePipelineServiceRole-eu-west-2-MyFirstPipeline`.

## CodePipeline política de rol de servicio

La declaración de política de la función de CodePipeline servicio contiene los permisos mínimos para gestionar las canalizaciones. Puede editar la declaración del rol de servicio para quitar el acceso a los recursos que no use. Consulte la referencia de acción correspondiente para conocer los permisos mínimos requeridos CodePipeline para cada acción.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowS3BucketAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketVersioning",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::[[pipeArtifactBucketNames]]"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "{{accountId}}"
        }
      }
    },
    {
      "Sid": "AllowS3ObjectAccess",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl",
        "s3:GetObject",
```

```
        "s3:GetObjectVersion",
        "s3:PutObjectTagging",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::[[pipeArtifactBucketNames]]/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "{{accountId}}"
        }
      }
    }
  ]
}
```

> **Note**
>
> En la política, se requieren los siguientes permisos cuando los objetos de S3 del bucket de origen contienen etiquetas:
>
> ```
> s3:PutObjectTagging
> s3:GetObjectTagging
> s3:GetObjectVersionTagging
> ```

## Elimine los permisos de la función CodePipeline de servicio

Puede editar la instrucción del rol de servicio para quitar el acceso a los recursos que no use. Por ejemplo, si ninguna de sus canalizaciones incluye Elastic Beanstalk, puede editar la instrucción de política para quitar la sección que concede acceso a los recursos de Elastic Beanstalk.

Del mismo modo, si ninguna de tus canalizaciones lo incluye CodeDeploy, puedes editar la declaración de política para eliminar la sección que concede acceso a CodeDeploy los recursos:

```
{
"Action": [
    "codedeploy:CreateDeployment",
    "codedeploy:GetApplicationRevision",
```

```
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:RegisterApplicationRevision"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
```

## Agrega permisos al rol de CodePipeline servicio

Debe actualizar su instrucción de política del rol de servicio con los permisos de Servicio de AWS que no se hayan incluido en la instrucción de política predeterminada del rol de servicio para poder usarlos en las canalizaciones.

Esto es especialmente importante si la función de servicio que utilizas para tus canalizaciones se creó antes de añadir el soporte a una Servicio de AWS.

En la siguiente tabla se muestra cuándo se añadió la compatibilidad con otros Servicios de AWS.

| Servicio de AWS | CodePipeline fecha de soporte |
| --- | --- |
| CodePipeline se agregó el soporte para invocar acciones. Consulte Permisos de política de roles de servicio para la acción de CodePipeline invocación. | 14 de marzo de 2025 |
| Se ha agregado compatibilidad con la acción de EC2. Consulte Permisos de la política de rol de servicio para la acción de implementación de EC2. | 21 de febrero de 2025 |
| Se ha agregado compatibilidad con la acción de EKS. Consulte Permisos para las políticas de roles de servicio. | 20 de febrero de 2025 |
| Se ha agregado compatibilidad con la acción `ECRBuildAndPublish` de Amazon Elastic Container Registry. Consulte Permisos del rol de servicio: acción `ECRBuildAndPublish`. | 22 de noviembre de 2024 |

| Servicio de AWS | CodePipeline fecha de soporte |
| --- | --- |
| Se ha agregado compatibilidad con la acción `InspectorScan` de Amazon Inspector. Consulte Permisos del rol de servicio: acción `InspectorScan`. | 22 de noviembre de 2024 |
| Se ha agregado compatibilidad con la acción de comandos. Consulte Permisos del rol de servicio: acción de Comandos. | 3 de octubre de 2024 |
| CloudFormation se agregó soporte para acciones. Consulte Permisos del rol de servicio: acción de `CloudFormationStackSet` y Permisos del rol de servicio: acción `CloudFormationStackInstances`. | 30 de diciembre de 2020 |
| CodeCommit Se agregó soporte para acciones en formato de artefacto de salida de clones completos. Consulte Permisos de rol de servicio: CodeCommit acción. | 11 de noviembre de 2020 |
| AWS CodeBuild Se agregó el soporte para acciones de construcción por lotes. Consulte Permisos de rol de servicio: CodeCommit acción. | 30 de julio de 2020 |
| AWS AppConfig se agregó soporte para acciones. Consulte Permisos del rol de servicio: acción de `AppConfig`. | 22 de junio de 2020 |
| AWS Step Functions soporte de acción agregado. Consulte Permisos del rol de servicio: acción `StepFunctions`. | 27 de mayo de 2020 |

| Servicio de AWS | CodePipeline fecha de soporte |
| --- | --- |
| AWS CodeStar Se agregó el soporte para acciones de conexiones. Consulte Permisos del rol de servicio: AWS CodeConnections acción. | 18 de diciembre de 2019 |
| Se ha agregado compatibilidad con la acción de implementación de S3. Consulte Permisos del rol de servicio: acción de implementación de S3. | 16 de enero de 2019 |
| Se ha agregado compatibilidad con la acción `CodeDeployToECS` . Consulte Permisos del rol de servicio: acción de `CodeDeployToECS` . | 27 de noviembre de 2018 |
| Se ha agregado compatibilidad con la acción de Amazon ECR. Consulte Permisos del rol de servicio: acción de Amazon ECR. | 27 de noviembre de 2018 |
| Se ha agregado compatibilidad con la acción de Service Catalog. Consulte Permisos del rol de servicio: acción de Service Catalog. | 16 de octubre de 2018 |
| AWS Device Farm Se agregó soporte para acciones. Consulte Permisos de rol de servicio: AWS Device Farm acción. | 19 de julio de 2018 |
| Se ha agregado compatibilidad con la acción de Amazon ECS. Consulte Permisos del rol de servicio: acción estándar de Amazon ECS. | 12 de diciembre de 2017 /Actualización para optar por la autorización de etiquetado el 21 de julio de 2017 |
| CodeCommit soporte de acción agregado. Consulte Permisos de rol de servicio: CodeCommit acción. | 18 de abril de 2016 |
| AWS OpsWorks soporte de acción agregado. Consulte Permisos de rol de servicio: AWS OpsWorks acción. | 2 de junio de 2016 |

| Servicio de AWS | CodePipeline fecha de soporte |
| --- | --- |
| CloudFormation soporte de acción agregado. Consulte Permisos de rol de servicio: CloudFormation acción. | 3 de noviembre de 2016 |
| AWS CodeBuild soporte de acción agregado. Consulte Permisos de rol de servicio: CodeBuild acción. | 1 de diciembre de 2016 |
| Se ha agregado compatibilidad con la acción de Elastic Beanstalk. Consulte Permisos del rol de servicio: acción de implementación `ElasticBeanstalk` . | Lanzamiento del servicio inicial |
| CodeDeploy soporte de acción agregado. Consulte Permisos de rol de servicio: AWS CodeDeploy acción. | Lanzamiento del servicio inicial |
| Se ha agregado compatibilidad con la acción de origen de S3. Consulte Permisos del rol de servicio: acción de origen de S3. | Lanzamiento del servicio inicial |

Siga estos pasos para añadir permisos a un servicio compatible:

1. Inicie sesión en la consola de IAM Consola de administración de AWS y ábrala en https://console.aws.amazon.com/iam/.
2. En la consola de IAM, en el panel de navegación, elija Roles y seleccione su rol `AWS-CodePipeline-Service` en la lista de roles.
3. En la pestaña Permisos, en Políticas en línea, en la fila de su política de rol de servicio, elija Editar política.
4. Añada los permisos necesarios en el cuadro Documento de política.

> **Note**
>
> Al crear políticas de IAM, siga los consejos de seguridad estándar de concesión de privilegios mínimos, es decir, conceder solo los permisos necesarios para realizar

una tarea. Algunas llamadas a la API admiten los permisos basados en recursos y permiten limitar el acceso. Por ejemplo, en este caso, para limitar los permisos cuando se llama a `DescribeTasks` y `ListTasks`, puede sustituir el carácter comodín (*) por un ARN de recurso o por un ARN de recurso que contenga un carácter comodín (*). Para obtener más información acerca de la creación de un política que concede acceso con privilegios mínimos, consulte https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege.

5. Elija Revisar política para asegurarse de que la política no contiene errores. Cuando la política no tenga errores, elija Aplicar política.

## Inicio de sesión y supervisión CodePipeline

Puede utilizar las funciones de inicio de sesión AWS para determinar las acciones que los usuarios han realizado en su cuenta y los recursos que se han utilizado. Los archivos de registro muestran:

- La fecha y la hora de las acciones.
- La dirección IP de origen de una acción.
- Las acciones que han fallado debido a permisos inadecuados.

Las características de registro están disponibles en los siguientes Servicios de AWS:

- AWS CloudTrail se puede usar para registrar las llamadas a la AWS API y los eventos relacionados realizados por o en nombre de un Cuenta de AWS. Para obtener más información, consulte Registro de llamadas a la CodePipeline API con AWS CloudTrail.
- Amazon CloudWatch Events se puede utilizar para supervisar Nube de AWS los recursos y las aplicaciones en las que se ejecuta AWS. Puede crear alertas en Amazon CloudWatch Events en función de las métricas que defina. Para obtener más información, consulte Supervisión de CodePipeline eventos.

## Validación de conformidad para AWS CodePipeline

Nuestra nueva experiencia AWS de registro no está diseñada para cargas de trabajo reguladas. Si está utilizando nuestra nueva experiencia de AWS registro, pero quiere utilizarla AWS para cargas de trabajo reguladas, puede suscribirse a una experiencia AWS (avanzada) o activar funciones avanzadas para su entorno. AWS

Para saber si un programa de cumplimiento Servicio de AWS se encuentra dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de cumplimiento Servicios de AWS y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros mediante Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. Para obtener más información sobre su responsabilidad de conformidad al utilizarlos Servicios de AWS, consulte AWS la documentación de seguridad.

## Resiliencia en AWS CodePipeline

La infraestructura AWS global se basa en AWS regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Para obtener más información sobre AWS las regiones y las zonas de disponibilidad, consulte Infraestructura global.AWS

## Seguridad de la infraestructura en AWS CodePipeline

Como servicio gestionado, AWS CodePipeline está protegido por la seguridad de la red AWS global. Para obtener información sobre los servicios AWS de seguridad y cómo se AWS protege la infraestructura, consulte Seguridad AWS en la nube. Para diseñar su AWS entorno utilizando las mejores prácticas de seguridad de la infraestructura, consulte Protección de infraestructuras en un marco de buena  AWS arquitectura basado en el pilar de la seguridad.

Utiliza las llamadas a la API AWS publicadas para acceder a CodePipeline través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.

- Paquetes de cifrado con perfecto secreto directo (PFS), como el DHE (Ephemeral) o el ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

# Prácticas recomendadas de seguridad

CodePipeline proporciona una serie de características de seguridad que debe tener en cuenta a la hora de desarrollar e implementar sus propias políticas de seguridad. Las siguientes prácticas recomendadas son directrices generales y no constituyen una solución de seguridad completa. Puesto que es posible que estas prácticas recomendadas no sean adecuadas o suficientes para el entorno, considérelas como consideraciones útiles en lugar de como normas.

Utilice el cifrado y la autenticación para los repositorios de origen que se conectan a sus canalizaciones. Estas son las CodePipeline mejores prácticas de seguridad:

- Si crea una configuración de canalización o acción que debe incluir secretos, como tokens o contraseñas, no introduzca los secretos directamente en la configuración de la acción ni los valores predeterminados de las variables definidas a nivel de canalización o CloudFormation configuración, ya que la información se mostrará en los registros. Utilice Secrets Manager para configurar y almacenar los secretos y, a continuación, utilice el secreto al que se hace referencia en la configuración de la canalización y la acción, tal y como se describe en Uso AWS Secrets Manager para rastrear las contraseñas de las bases de datos o las claves de API de terceros.
- Si crea una canalización que utiliza un bucket de origen de S3, configure el cifrado del lado del servidor para los artefactos almacenados en Amazon S3 CodePipeline mediante la administración AWS KMS keys, tal y como se describe en. Configurar el cifrado del lado del servidor para los artefactos almacenados en Amazon S3 para CodePipeline
- Cuando utilice un proveedor de compilación de Jenkins para la acción de compilación o prueba de la canalización, instale Jenkins en una instancia EC2 y configure un perfil de instancia EC2 distinto. Asegúrese de que el perfil de instancia conceda a Jenkins solo los AWS permisos necesarios para realizar las tareas del proyecto, como la recuperación de archivos de Amazon S3. Para saber cómo crear la función para su perfil de instancia de Jenkins, consulte los pasos que se describen en Creación de un rol de IAM para usar en la integración de Jenkins.

# CodePipeline referencia de estructura de tuberías

Puede utilizarla CodePipeline para estructurar un conjunto CI/CD de pasos automatizados que lleven a cabo las tareas de creación, prueba e implementación del código fuente de la aplicación. En esta sección de referencia, se proporcionan detalles sobre la estructura y los parámetros de JSON de la canalización. Para obtener una lista general de conceptos que describen cómo se utilizan las canalizaciones, consulte CodePipeline conceptos.

- Cuando creas una canalización, eliges una acción fuente y un proveedor disponibles, como un bucket de S3, un CodeCommit repositorio, un repositorio de Bitbucket o un GitHub repositorio que contenga tu código fuente e inicie la canalización cuando confirmas un cambio en el código fuente. Esta sección de referencia proporciona información de referencia sobre los orígenes disponibles para su canalización. Para obtener más información acerca de cómo trabajar con acciones de origen, consulte Inicie una canalización en CodePipeline.
- Puede elegir las acciones y los proveedores de prueba, compilación e implementación que desee incluir automáticamente cuando se ejecute la canalización. Esta sección de referencia proporciona información de referencia sobre las acciones disponibles y cómo encajan en el JSON de la canalización.
- La canalización finalizada constará de una etapa de origen junto con etapas adicionales en las que se configurarán las acciones para implementar y probar la aplicación. Para ver un ejemplo conceptual de una DevOps canalización que implementa tu aplicación, consulta. DevOps ejemplo de canalización

De forma predeterminada, cualquier canalización en la que se cree correctamente AWS CodePipeline tiene una estructura válida. Sin embargo, si creas o editas manualmente un archivo JSON para crear una canalización o actualizas una canalización desde el AWS CLI, puedes crear inadvertidamente una estructura que no sea válida. La siguiente referencia puede ayudarle a entender mejor los requisitos de estructura de su canalización y cómo solucionar los problemas. Consulte las restricciones en Cuotas en AWS CodePipeline, que se aplican a todas las canalizaciones.

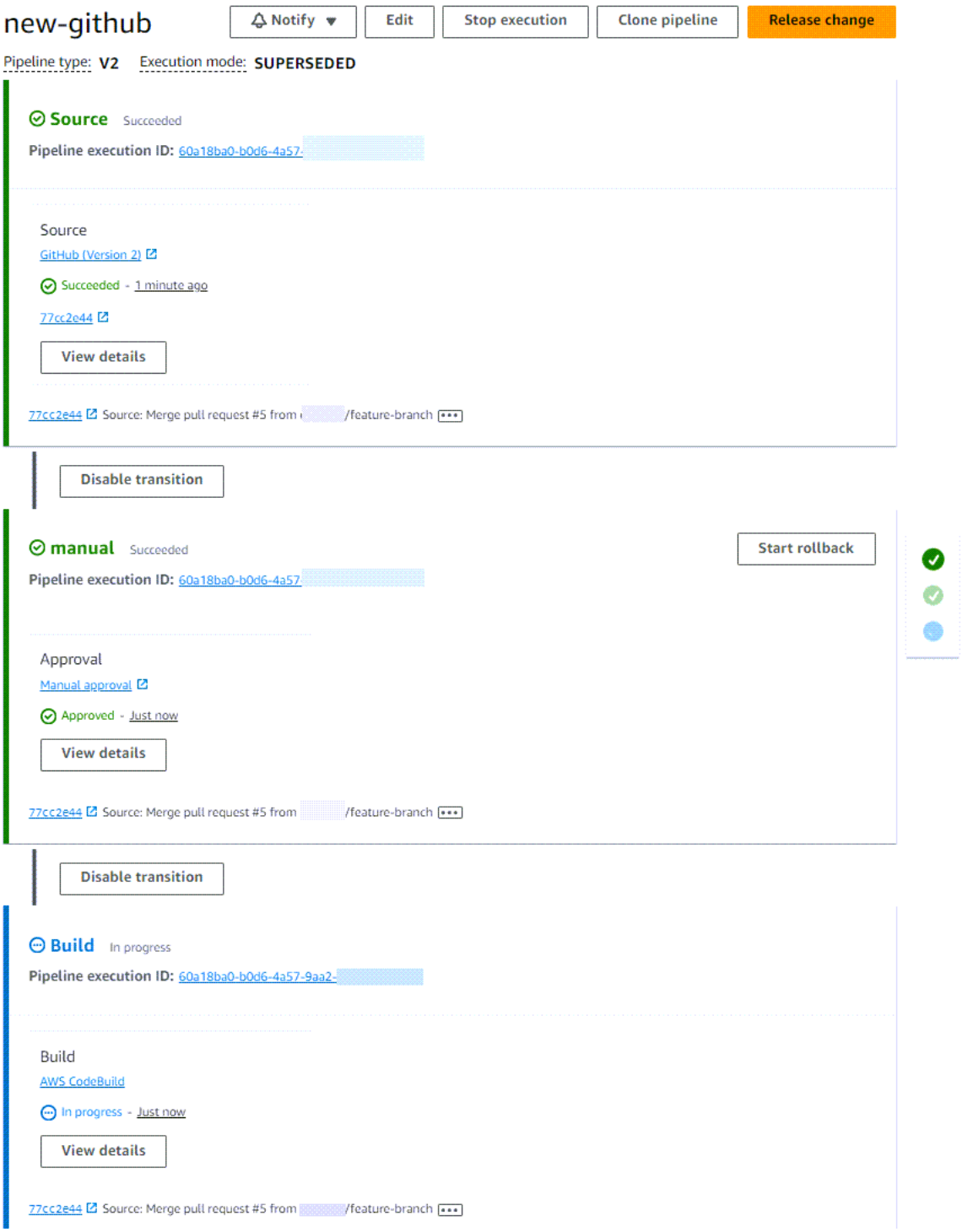
En las siguientes secciones, se proporcionan parámetros de alto nivel y su posición en la estructura de canalización. Los requisitos de estructura de la canalización se detallan en cada sección para los siguientes tipos de componentes de canalización:

- Referencia de campo para Declaración de canalización

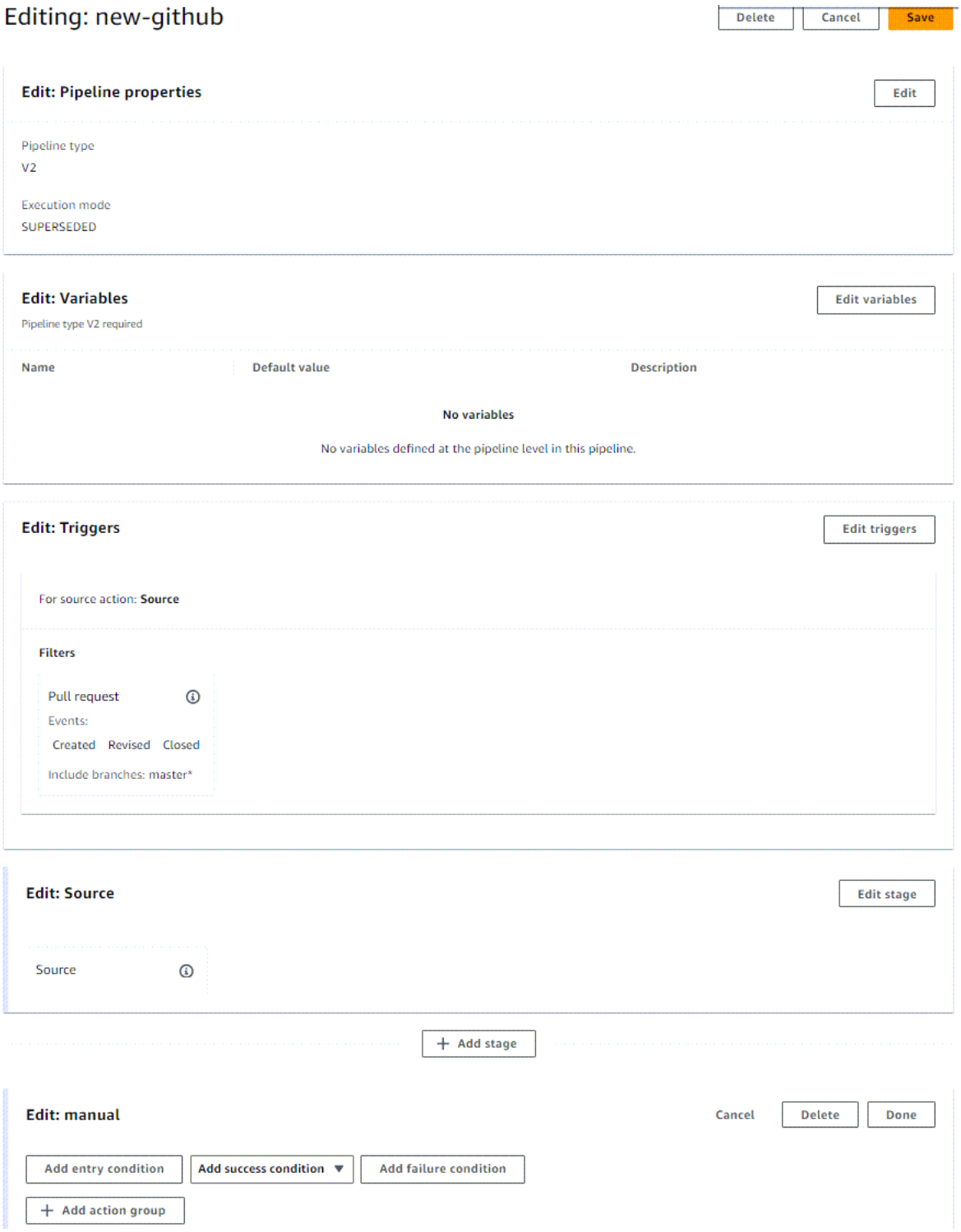
- Referencia de campo para Declaración de etapas
- Referencia de campo para Declaración de acciones
- Lista de Proveedores de acciones válidos en CodePipeline por tipo de acción
- Referencia de la para
- Referencia de la para
- Lista de enlaces para Parámetros de configuración válidos para cada tipo de proveedor

Para obtener más información, consulta el PipelineDeclaration objeto en la CodePipeline Guía de API.

El siguiente ejemplo de vista de consola de canalización muestra la canalización denominada new-github, las etapas denominadas `Source` `manualBuild`, y las acciones de GitHub (a través de la GitHub aplicación), la aprobación manual y los proveedores de CodeBuild acciones.




El modo de edición de canalizaciones, visto en el diagrama de la consola, permite editar las anulaciones, desencadenadores y acciones de origen, tal y como se muestra en el siguiente ejemplo.




## Temas

- Declaración de canalización
- Declaración de etapas
- Declaración de acciones

- Proveedores de acciones válidos en CodePipeline
- Configuración válida para el parámetro PollForSourceChanges
- Artefactos de entrada y salida para cada tipo de acción
- Parámetros de configuración válidos para cada tipo de proveedor

## Declaración de canalización

La canalización y el nivel de metadatos de una canalización tienen una estructura básica que incluye los siguientes parámetros y sintaxis. El parámetro de canalización representa la estructura de las acciones y etapas que se van a realizar en la canalización.

Para obtener más información, consulte el PipelineDeclaration objeto en la Guía de CodePipeline API.

El siguiente ejemplo muestra la canalización y el nivel de metadatos de la estructura de la canalización tanto en JSON como en YAML para una canalización de tipo V2.

YAML

```
pipeline:
  name: MyPipeline
  roleArn: >-
    arn:aws:iam::ACCOUNT_ID:role/service-role/AWSCodePipelineServiceRole-us-west-2-
MyPipeline
  artifactStore:
    type: S3
    location: amzn-s3-demo-bucket
  stages:
    ...
  version: 6
  executionMode: SUPERSEDED
  pipelineType: V2
  variables:
  - name: MyVariable
    defaultValue: '1'
  triggers:
  - providerType: CodeStarSourceConnection
    gitConfiguration:
      sourceActionName: Source
      push:
```

```
      - branches:
          includes:
          - main
          excludes:
          - feature-branch
        pullRequest:
        - events:
          - CLOSED
          branches:
            includes:
            - main*
metadata:
  pipelineArn: 'arn:aws:codepipeline:us-west-2:ACCOUNT_ID:MyPipeline'
  created: '2019-12-12T06:49:02.733000+00:00'
  updated: '2020-09-10T06:34:07.447000+00:00'
  pollingDisabledAt: '2020-09-10T06:34:07.447000+00:00'
```

JSON

```
{
    "pipeline": {
        "name": "MyPipeline",
        "roleArn": "arn:aws:iam::ACCOUNT_ID:role/service-role/
AWSCodePipelineServiceRole-us-west-2-MyPipeline",
        "artifactStore": {
            "type": "S3",
            "location": "amzn-s3-demo-bucket"
        },
        "stages": {
            ...
    },
        "version": 6,
        "executionMode": "SUPERSEDED",
                "pipelineType": "V2",
        "variables": [
            {
                "name": "MyVariable",
                "defaultValue": "1"
            }
        ],
        "triggers": [
            {
                "providerType": "CodeStarSourceConnection",
```

```
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "branches": {
                                "includes": [
                                    "main"
                                ],
                                "excludes": [
                                    "feature-branch"
                                ]
                            }
                        }
                    ],
                    "pullRequest": [
                        {
                            "events": [
                                "CLOSED"
                            ],
                            "branches": {
                                "includes": [
                                    "main*"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    },
    "metadata": {
        "pipelineArn": "arn:aws:codepipeline:us-west-2:ACCOUNT_ID:MyPipeline",
        "created": "2019-12-12T06:49:02.733000+00:00",
        "updated": "2020-09-10T06:34:07.447000+00:00",
        "pollingDisabledAt": "2020-09-10T06:34:07.447000+00:00"
    }
}
```

## name

El nombre de la canalización. El nombre de una canalización no se puede modificar al editarla o actualizarla.

> **Note**
>
> Si desea cambiar el nombre de una canalización existente, puede utilizar el comando `get-pipeline` de la CLI para crear un archivo JSON que incluya la estructura de la canalización. A continuación, puede utilizar el comando `create-pipeline` de la CLI para crear una canalización con esa estructura y asignarla un nombre nuevo.

## `roleArn`

El ARN de IAM para la función de CodePipeline servicio, como arn:aws:iam: :80398 _. EXAMPLE:role/CodePipeline Service_Role

Para usar la consola a fin de ver el ARN del rol de servicio de la canalización en lugar de la estructura JSON, elija su canalización en la consola y, a continuación, seleccione Configuración. En la pestaña General, aparecerá el campo ARN del rol de servicio.

## `ArtifactStore` O `ArtifactStores`

El `artifactStore` campo contiene el tipo de depósito de artefactos y la ubicación de una canalización con todas las acciones en la misma región. AWS Si añades acciones en una región diferente a la que estás en proceso, el `artifactStores` mapa se usa para mostrar el grupo de artefactos de cada AWS región en la que se ejecutan las acciones. Al crear o editar una canalización, debe tener un bucket de artefactos en la región de la canalización, así como un bucket de artefactos por cada región en la que tiene previsto ejecutar una acción.

> **Note**
>
> En la estructura de la canalización, debe incluir `artifactStore` o `artifactStores` en su canalización, pero no puede usar ambos. Si crea una acción entre regiones en la canalización, debe utilizar `artifactStores`.

En el siguiente ejemplo, se muestra la estructura básica de una canalización con acciones entre regiones que utiliza el parámetro `artifactStores`:

```
    "pipeline": {
        "name": "YourPipelineName",
```

```
        "roleArn": "CodePipeline_Service_Role",
        "artifactStores": {
            "us-east-1": {
                "type": "S3",
                "location": "S3 artifact bucket name, such as amzn-s3-demo-bucket"
            },
            "us-west-2": {
                "type": "S3",
                "location": "S3 artifact bucket name, such as amzn-s3-demo-bucket"
            }
        },
        "stages": [
            {

...
```

### type

El tipo de ubicación del bucket de artefactos, especificado como Amazon S3.

### ubicación

El nombre del bucket de Amazon S3 que se generó automáticamente la primera vez que creó una canalización con la consola, como codepipeline-us-east-2-1234567890, o cualquier bucket de Amazon S3 que aprovisione para este fin

## etapas

Este parámetro contiene el nombre de cada etapa de la canalización. Para obtener más información sobre los parámetros y la sintaxis en el nivel de fase de la estructura de canalización, consulta el StageDeclaration objeto en la guía de la CodePipeline API.

La estructura de la canalización para las etapas debe cumplir estos requisitos:

- La canalización debe tener dos etapas como mínimo.
- La primera fase de una canalización debe incluir al menos una acción de origen. Solo puede incluir acciones de origen.
- La primera etapa de la canalización es la única que puede incluir acciones de origen.
- Al menos una etapa de cada canalización debe contener una acción que no sea de origen.
- Los nombres de las etapas de una canalización deben ser diferentes.

- Los nombres de las etapas no se pueden editar en la CodePipeline consola. Si editas el nombre de una etapa con la AWS CLI y la etapa contiene una acción con uno o más parámetros secretos (como un token de OAuth), el valor de esos parámetros secretos no se conserva. Tendrá que escribir manualmente el valor de los parámetros (que están enmascarados con cuatro asteriscos en el archivo JSON que devuelve la AWS CLI) e incluirlos en la estructura JSON.

> **Important**
>
> Las canalizaciones que estén inactivas durante más de 30 días tendrán deshabilitado el sondeo para la canalización. Para obtener más información, consulte el sondeo DisabledAt en la referencia a la estructura de canalización. Consulte los pasos para migrar una canalización que usa sondeos para que use la detección de cambios basada en eventos en Métodos de detección de cambios.

## `versión`

El número de versión de una canalización se genera automáticamente y se actualiza cada vez que se actualiza la canalización.

## `executionMode`

Puede configurar el modo de ejecución de la canalización para poder especificar el comportamiento de la canalización para ejecuciones consecutivas, como poner en cola, reemplazar o ejecutar en modo paralelo. Para obtener más información, consulte Configuración o cambio del modo de ejecución de una canalización.

> **Important**
>
> Para las canalizaciones en modo PARALELO, la reversión de etapas no está disponible. Del mismo modo, las condiciones de error con un tipo de resultado de reversión no se pueden agregar a una canalización en modo PARALELO.

## `Tipo de tubería`

El tipo de canalización especifica la estructura y las características disponibles en la canalización, como en el caso de una canalización de tipo V2. Para obtener más información, consulte Tipos de canalización.

## `variables`

Las variables a nivel de canalización se definen cuando la canalización se crea y se resuelven en el tiempo de ejecución de la canalización. Para obtener más información, consulte Referencia de variables. Para ver un tutorial con una variable a nivel de canalización que se transfiere en el momento de la ejecución de la canalización, consulte Tutorial: Uso de variables a nivel de canalización.

## `desencadenadores`

Los desencadenadores permiten configurar la canalización para que se inicie con un tipo de evento concreto o filtrado, por ejemplo, cuando se detecta un cambio en una ramificación específica o en una solicitud de extracción. Los activadores se pueden configurar para las acciones de origen con conexiones que utilizan la `CodeStarSourceConnection` acción en CodePipeline, por ejemplo, Bitbucket GitHub y. GitLab Para obtener más información acerca de las acciones de origen que utilizan conexiones, consulte Agrega proveedores de fuentes de terceros a las canalizaciones mediante AWS CodeConnections.

Para obtener más información y ejemplos detallados, consulte Automatización del inicio de las canalizaciones mediante desencadenadores y filtrado.

Para el filtrado, se admiten patrones de expresiones regulares en formato glob, tal y como se detalla en Trabajar con patrones glob en la sintaxis.

> **Note**
>
> Las acciones de origen CodeCommit y de S3 requieren un recurso de detección de cambios configurado (una EventBridge regla) o utilizar la opción para consultar el repositorio en busca de cambios en el origen. En el caso de las canalizaciones con una acción fuente de Bitbucket o GitHub Enterprise Server, no es necesario configurar un webhook ni realizar sondeos de forma predeterminada. GitHub La acción de conexiones administra la detección de cambios por usted.

> **Important**
>
> Las canalizaciones que estén inactivas durante más de 30 días tendrán deshabilitado el sondeo para la canalización. Para obtener más información, consulta el sondeo DisabledAt en la referencia sobre la estructura de canalizaciones. Consulte los pasos para migrar una canalización que usa sondeos para que use la detección de cambios basada en eventos en Métodos de detección de cambios.

### `Campos de configuración` de Git

La configuración de Git para el desencadenador, incluidos los tipos de eventos y cualquier parámetro para filtrar por ramas, rutas de archivos, etiquetas o eventos de solicitud de extracción.

Los campos de la estructura JSON se definen de la siguiente manera:

- `sourceActionName`: el nombre de la acción de origen de la canalización con la configuración de Git.
- `push`: eventos de inserción con filtrado. Estos eventos utilizan una operación OR entre distintos filtros de inserción y una operación AND en los filtros.
  - `branches`: las ramificaciones se van a filtrar. Las ramificaciones utilizan una operación AND entre inclusiones y exclusiones.
    - `includes`: patrones para filtrar las ramificaciones que se incluirán. Incluye el uso de una operación OR.
    - `excludes`: patrones para filtrar las ramificaciones que se excluirán. Excluye el uso de una operación OR.
  - `filePaths`: los nombres de las rutas de archivo que se van a filtrar.
    - `includes`: patrones para filtrar las rutas de archivo que se incluirán. Incluye el uso de una operación OR.
    - `excludes`: patrones para filtrar las rutas de archivo que se excluirán. Excluye el uso de una operación OR.
  - `tags`: los nombres de las etiquetas que se van a filtrar.
    - `includes`: patrones para filtrar las etiquetas que se incluirán. Incluye el uso de una operación OR.
    - `excludes`: patrones para filtrar las etiquetas que se excluirán. Excluye el uso de una operación OR.

- `pullRequest`: eventos de solicitud de extracción con filtrado de eventos de solicitud de extracción y filtros de solicitud de extracción.
  - `events`: filtra los eventos de solicitud de extracción que se abran, actualicen o cierren según se especifique.
  - `branches`: las ramificaciones se van a filtrar. Las ramificaciones utilizan una operación AND entre inclusiones y exclusiones.
    - `includes`: patrones para filtrar las ramificaciones que se incluirán. Incluye el uso de una operación OR.
    - `excludes`: patrones para filtrar las ramificaciones que se excluirán. Excluye el uso de una operación OR.
  - `filePaths`: los nombres de las rutas de archivo que se van a filtrar.
    - `includes`: patrones para filtrar las rutas de archivo que se incluirán. Incluye el uso de una operación OR.
    - `excludes`: patrones para filtrar las rutas de archivo que se excluirán. Excluye el uso de una operación OR.

El siguiente es un ejemplo de la configuración del desencadenador para los tipos de eventos de solicitud de inserción y extracción.

```
"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"
                            ]
                        }
                    }
```

```
                    ]
                }
            }
        ],
```

### **Campos** push de tipo evento para incluir y excluir

El comportamiento de inclusión y exclusión de los niveles de los campos de configuración de Git para los tipos de eventos de inserción se muestra en la siguiente lista:

```
push (OR operation is used between push and pullRequest or multiples)
    filePaths (AND operation is used between filePaths, branches, and tags)
        includes (AND operation is used between includes and excludes)
            **/FILE.md, **/FILE2 (OR operation is used between file path names)
        excludes (AND operation is used between includes and excludes)
            **/FILE.md, **/FILE2 (OR operation is used between file path names)
    branches (AND operation is used between filePaths, branches, and tags)
        includes (AND operation is used between includes and excludes)
            BRANCH/**", "BRANCH2/** (OR operation is used between branch names)
        excludes (AND operation is used between includes and excludes)
            BRANCH/**", "BRANCH2/** (OR operation is used between branch names)
    tags (AND operation is used between filePaths, branches, and tags)
         includes (AND operation is used between includes and excludes)
            TAG/**", "TAG2/** (OR operation is used between tag names)
         excludes (AND operation is used between includes and excludes)
            TAG/**", "TAG2/** (OR operation is used between tag names)
```

### **Campos de solicitud de** extracción de tipo de evento para incluir y excluir

El comportamiento de inclusión y exclusión de los niveles de los campos de configuración de Git para los tipos de eventos de solicitud de extracción se muestra en la siguiente lista:

```
pullRequest (OR operation is used between push and pullRequest or multiples)
    events (AND operation is used between events, filePaths, and branches). Includes/
excludes are N/A for pull request events.
    filePaths (AND operation is used between events, filePaths, and branches)
        includes (AND operation is used between includes and excludes)
            **/FILE.md, **/FILE2 (OR operation is used between file path names)
        excludes (AND operation is used between includes and excludes)
            **/FILE.md, **/FILE2 (OR operation is used between file path names)
    branches (AND operation is used between events, filePaths, and branches)
        includes (AND operation is used between includes and excludes)
```

```
            BRANCH/**", "BRANCH2/** (OR operation is used between branch names)
        excludes (AND operation is used between includes and excludes)
            BRANCH/**", "BRANCH2/** (OR operation is used between branch names)
```

## metadatos

Los campos de metadatos de la canalización son distintos de la estructura de la canalización y no se pueden editar. Al actualizar una canalización, la fecha del campo de metadatos `updated` cambia automáticamente.

### pipelineArn

El nombre de recurso de Amazon (ARN) de la canalización.

Para usar la consola a fin de ver el ARN de la canalización en lugar de la estructura JSON, elija su canalización en la consola y, a continuación, seleccione Configuración. En la pestaña General, aparecerá el campo ARN de la canalización.

### created

Fecha y hora en que se creó la canalización.

### updated

Fecha y hora en que se actualizó por última vez la canalización.

### sondeo DisabledAt

La fecha y la hora en las que se deshabilitó el sondeo, en el caso de una canalización configurada para el sondeo con fines de detección de cambios.

Las canalizaciones que estén inactivas durante más de 30 días tendrán deshabilitado el sondeo para la canalización.

- Si no se llevan a cabo ejecuciones en 30 días, se deshabilitará el sondeo en las canalizaciones inactivas.
- Las canalizaciones que utilicen EventBridge CodeStar conexiones o webhooks no se verán afectadas.
- Las canalizaciones activas no se verán afectadas.

Para obtener más información, consulta el `pollingDisabledAt` parámetro situado bajo el PipelineMetadata objeto en la Guía de CodePipeline API. Consulte los pasos para migrar una canalización que usa sondeos para que use la detección de cambios basada en eventos en Métodos de detección de cambios.

## Declaración de etapas

El nivel de etapa de una canalización tiene una estructura básica que incluye los siguientes parámetros y sintaxis. Para obtener más información, consulte el StageDeclaration objeto en la Guía de CodePipeline API.

En el siguiente ejemplo se muestra el nivel de etapa de la estructura de canalización en JSON y YAML. El ejemplo muestra dos etapas denominadas `Source` y `Build`. El ejemplo contiene dos condiciones, una para `onSuccess` y otra para `beforeEntry`.

YAML

```
pipeline:
  name: MyPipeline
  roleArn: >-
    arn:aws:iam::ACCOUNT_ID:role/service-role/AWSCodePipelineServiceRole-us-west-2-
MyPipeline
  artifactStore:
    type: S3
    location: amzn-s3-demo-bucket
  stages:
    - name: Source
      actions:
        - name: Source
          ...
    - name: Build
      actions:
        - name: Build
          ...
      onSuccess:
        conditions:
        - result: ROLLBACK
          rules:
          - name: DeploymentWindowRule
           ...
      beforeEntry:
        conditions:
```

```
        - result: FAIL
          rules:
          - name: MyLambdaRule
          ...
  version: 6
metadata:
  pipelineArn: 'arn:aws:codepipeline:us-west-2:ACCOUNT_ID:MyPipeline'
  created: '2019-12-12T06:49:02.733000+00:00'
  updated: '2020-09-10T06:34:07.447000+00:00'
```

JSON

```
{
    "pipeline": {
        "name": "MyPipeline",
        "roleArn": "arn:aws:iam::ACCOUNT_ID:role/service-role/
AWSCodePipelineServiceRole-us-west-2-MyPipeline",
        "artifactStore": {
            "type": "S3",
            "location": "amzn-s3-demo-bucket"
        },
        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "name": "Source",
                        ...
                    }
                ]
            },
            {
                "name": "Build",
                "actions": [
                    {
                        "name": "Build",
                        ...
                    }
                ],
                "onSuccess": {
                    "conditions": [
                        {
                            "result": "ROLLBACK",
```

```
                            "rules": [
                                {
                                    "name": "DeploymentWindowRule",
                                    ...
                                }
                            ]
                        }
                    ]
                },
                "beforeEntry": {
                    "conditions": [
                        {
                            "result": "FAIL",
                            "rules": [
                                {
                                    "name": "MyLambdaRule",
                                     ...
                                }
                            ]
                        }
                    ]
                }
            }
        ],

            }
        ],
        "version": 6
    },
    "metadata": {
        "pipelineArn": "arn:aws:codepipeline:us-west-2:ACCOUNT_ID:MyPipeline",
        "created": "2019-12-12T06:49:02.733000+00:00",
        "updated": "2020-09-10T06:34:07.447000+00:00"
    }
}
```

## name

El nombre de la etapa de .

### actions

El nivel de acción de una canalización tiene una estructura básica que incluye los siguientes parámetros y sintaxis. Para ver los parámetros y ejemplos, consulte Declaración de acciones.

### condiciones

Las condiciones contienen una o más reglas que están disponibles en una lista de reglas en CodePipeline. Si todas las reglas de una condición se realizan correctamente, se cumple la condición. Puede configurar las condiciones para que, cuando no se cumplan los criterios, se active el resultado especificado.

Es posible configurar los siguientes tipos de condiciones:

- `beforeEntry`
- `onFailure`
- `onSuccess`

Para obtener más información y ejemplos, consulta Configuración de las condiciones de una etapa.

### reglas

Cada condición tiene un conjunto de reglas, en el que las reglas se ordenan y se evalúan juntas. Por lo tanto, si una regla falla en la condición, la condición también fallará. Puede anular las condiciones de la regla en el tiempo de ejecución de la canalización.

Las reglas disponibles se proporcionan en la Referencia de reglas. Para obtener más información, consulte la Referencia de la estructura de reglas en Referencia de estructura de las reglas.

## Declaración de acciones

El nivel de acción de una canalización tiene una estructura básica que incluye los siguientes parámetros y sintaxis. Para obtener más información, consulte el ActionDeclaration objeto en la Guía de CodePipeline API.

En el siguiente ejemplo se muestra el nivel de acción de la estructura de canalización tanto en JSON como en YAML.

YAML

```
. . .

  stages:
    - name: Source
      actions:
        - name: Source
          actionTypeId:
            category: Source
            owner: AWS
            provider: S3
            version: '1'
          runOrder: 1
          configuration:
            PollForSourceChanges: 'false'
            S3Bucket: amzn-s3-demo-bucket
            S3ObjectKey: codedeploy_linux.zip
          outputArtifacts:
            - name: SourceArtifact
          inputArtifacts: []
          region: us-west-2
          namespace: SourceVariables
    - name: Build
      actions:
        - name: Build
          actionTypeId:
            category: Build
            owner: AWS
            provider: CodeBuild
            version: '1'
          runOrder: 1
          configuration:
            EnvironmentVariables: >-
              [{"name":"ETag","value":"#{SourceVariables.ETag}","type":"PLAINTEXT"}]
            ProjectName: my-project
          outputArtifacts:
            - name: BuildArtifact
          inputArtifacts:
            - name: SourceArtifact
          region: us-west-2
          namespace: BuildVariables
          runOrder: 1
```

```
          configuration:
            CustomData: >-
              Here are the exported variables from the build action: S3 ETAG:
              #{BuildVariables.ETag}
          outputArtifacts: []
          inputArtifacts: []
          region: us-west-2
```

JSON

```
. . .

        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "provider": "S3",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "PollForSourceChanges": "false",
                            "S3Bucket": "amzn-s3-demo-bucket",
                            "S3ObjectKey": "aws-codepipeline-s3-aws-
codedeploy_linux.zip"
                        },
                        "outputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "inputArtifacts": [],
                        "region": "us-west-2",
                        "namespace": "SourceVariables"
                    }
                ]
            },
```

```
            {
                "name": "Build",
                "actions": [
                    {
                        "name": "Build",
                        "actionTypeId": {
                            "category": "Build",
                            "owner": "AWS",
                            "provider": "CodeBuild",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "EnvironmentVariables": "[{\"name\":\"ETag\",\"value\":
\"#{SourceVariables.ETag}\",\"type\":\"PLAINTEXT\"}]",
                            "ProjectName": "my-build-project"
                        },
                        "outputArtifacts": [
                            {
                                "name": "BuildArtifact"
                            }
                        ],
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "region": "us-west-2",
                        "namespace": "BuildVariables"
                    }
                ]

. . .
```

Para obtener una lista de ejemplos de detalles de `configuration` apropiados para el tipo de proveedor, consulte Parámetros de configuración válidos para cada tipo de proveedor.

La estructura de la acción debe cumplir estos requisitos:

- Los nombres de las acciones de una etapa deben ser diferentes.
- Se requiere una acción de origen para cada canalización.

- Las acciones de origen que no utilizan una conexión se pueden configurar para que detecten los cambios, o bien para desactivar la opción de detección de cambios. Consulte Métodos de detección de cambios.
- Esto es así para todas las acciones, ya estén en la misma etapa o en etapas posteriores, pero el artefacto de entrada no debe ser necesariamente la siguiente acción en de una secuencia estricta cuyo origen es la acción que proporcionó el artefacto de salida. Las acciones en paralelo pueden declarar distintos paquetes de artefactos de salida que, a su vez, consumen las siguientes y distintas acciones.
- Cuando se utiliza un bucket de Amazon S3 como ubicación de implementación, también se especifica una clave de objeto. Una clave de objeto puede ser un nombre de archivo (objeto) o una combinación de un prefijo (ruta de carpeta) y el nombre del archivo. Puede utilizar variables para especificar el nombre de la ubicación que desea que utilice la canalización. Las acciones de implementación de Amazon S3 admiten el uso de las siguientes variables en las claves de objeto de Amazon S3.

  Uso de variables en Amazon S3

| Variable | Ejemplo de entrada de la consola | Output |
| --- | --- | --- |
| datetime | js-application/{datetime}.zip | Marca temporal UTC con este formato: <AAAA>-<MM>-DD>_<HH>-<MM>-<SS><br><br>Ejemplo:<br><br>js- application/2019 -01-10_07-39-57.zip |
| uuid | js-application/{uuid}.zip | El UUID es un identificador único global diferente de cualquier otro identific ador. El UUID tiene este formato (todos los dígitos en formato hexadecimal): <8 dígitos>-<4 dígitos>-4 dígitos>-<4 dígitos>-<12 dígitos><br><br>Ejemplo: |

| Variable | Ejemplo de entrada de la consola | Output |
| --- | --- | --- |
| | | js- application/54a60075 -b96a-4bf 3-9013-db3a9 EXAMPLE.zip |

## `name`

El nombre de la acción.

## `region`

Para las acciones en las que el proveedor es un, el del recurso. Servicio de AWS Región de AWS

Cross-Region las acciones utilizan el `Region` campo para designar el Región de AWS lugar donde se van a crear las acciones. Los AWS recursos creados para esta acción deben crearse en la misma región proporcionada en el `region` campo. No puede crear acciones entre regiones para los siguientes tipos de acciones:

- Acciones de código fuente
- Acciones de proveedores de terceros
- Acciones de proveedores personalizados

## `roleArn`

El ARN del rol de servicio de IAM que realizará la acción declarada. Esto se asume a través del roleArn que se especifica a nivel de la canalización.

## `namespace`

Las acciones se pueden configurar con variables. Utilice el campo `namespace` para establecer el espacio de nombres y la información de variables para las variables de ejecución. Para obtener información de referencia acerca de las variables de ejecución y las variables de salida de acción, consulte Referencia de variables.

> **Note**
>
> En el caso de Amazon ECR, Amazon S3 o CodeCommit las fuentes, también puede crear una anulación de fuente utilizando la entrada de transformación de entrada EventBridge para usar la entrada `revisionValue` en el evento de canalización, donde `revisionValue` se deriva de la variable de evento de origen de su clave de objeto, confirmación o ID de imagen. Para obtener más información, consulte el paso opcional para la entrada de transformación de entrada, que se incluye en los procedimientos de Acciones y recursos de origen de Amazon ECR EventBridge, Conexión a acciones de origen de Amazon S3 con un origen habilitado para eventos o CodeCommit acciones de origen y EventBridge.

## acción TypeId

El ID del tipo de acción se identifica como una combinación de los cuatro campos siguientes.

### categoría

El tipo de acción o paso de la canalización, como una acción de origen. Cada tipo de acción tiene un conjunto específico de proveedores de acción válidos. Para obtener una lista de los proveedores válidos según el tipo de acción, consulte la Referencia de la estructura de acciones.

Estas son las `actionTypeId` categorías (tipos de acción) válidas para CodePipeline:

- `Source`
- `Build`
- `Approval`
- `Deploy`
- `Test`
- `Invoke`
- `Compute`

### owner

La única cadena de versión válida para todos los tipos de acción admitidos actualmente es `AWS`, `ThirdParty` o `Custom`. Para ver la cadena de propietario válida para una acción específica, consulte la Referencia de la estructura de acciones.

Para obtener más información, consulte la Referencia de la API de CodePipeline .

### `versión`

La versión de la acción.

### `proveedor`

El proveedor de la acción, como CodeBuild.

- Los tipos de proveedor válidos para una categoría de acción dependen de la categoría. Por ejemplo, para una categoría de acción de origen, un tipo de proveedor válido sería `S3`, `CodeStarSourceConnection`, `CodeCommit` o `Amazon ECR`. Este ejemplo muestra la estructura de una acción de código fuente con `S3` como proveedor:

```
"actionTypeId": {
  "category": "Source",
  "owner": "AWS",
  "version": "1",
  "provider": "S3"},
```

## `InputArtifacts`

Este campo contiene la estructura del artefacto de entrada, si es compatible con la categoría de acción. El artefacto de entrada de una acción debe coincidir exactamente con el artefacto de salida declarado en una acción anterior. Por ejemplo, si una acción anterior incluye la siguiente declaración:

```
"outputArtifacts": [
    {
    "MyApp"
    }
],
```

y no hay otros artefactos de salida, el artefacto de entrada de una acción posterior debe ser:

```
"inputArtifacts": [
    {
    "MyApp"
    }
```

```
],
```

Por ejemplo, una acción de origen no puede tener artefactos de entrada porque es la primera acción de la canalización. Sin embargo, una acción de origen siempre tendrá artefactos de salida que se procesen mediante la siguiente acción. Los artefactos de salida de una acción de origen son los archivos de aplicación del repositorio de origen, comprimidos y proporcionados a través del depósito de artefactos, que se procesan mediante la siguiente acción, por ejemplo, una CodeBuild acción que actúa sobre los archivos de la aplicación mediante comandos de compilación.

Las acciones de implementación son un claro ejemplo de acciones que no pueden tener artefactos de salida, ya que estas acciones suelen ser la última acción de una canalización.

### `name`

El nombre de artefacto para los artefactos de entrada de la acción.

## `Artefacto de salida`

Los nombres de los artefactos de salida deben ser diferentes en una canalización. Por ejemplo, una canalización puede incluir una acción con un artefacto de salida que se llame `"MyApp"` y otra acción con un artefacto de salida llamado `"MyBuiltApp"`. Sin embargo, una canalización no puede incluir dos acciones que tengan un artefacto de salida denominado `"MyApp"`.

Este campo contiene la estructura del artefacto de salida, si es compatible con la categoría de acción. El artefacto de salida de una acción debe coincidir exactamente con el artefacto de salida declarado en una acción anterior. Por ejemplo, si una acción anterior incluye la siguiente declaración:

```
"outputArtifacts": [
    {
    "MyApp"
    }
],
```

y no hay otros artefactos de salida, el artefacto de entrada de una acción posterior debe ser:

```
"inputArtifacts": [
    {
    "MyApp"
    }
```

```
],
```

Por ejemplo, una acción de origen no puede tener artefactos de entrada porque es la primera acción de la canalización. Sin embargo, una acción de origen siempre tendrá artefactos de salida que se procesen mediante la siguiente acción. Los artefactos de salida de una acción de origen son los archivos de aplicación del repositorio de origen, comprimidos y proporcionados a través del depósito de artefactos, que se procesan mediante la siguiente acción, por ejemplo, una CodeBuild acción que actúa sobre los archivos de la aplicación mediante comandos de compilación.

Las acciones de implementación son un claro ejemplo de acciones que no pueden tener artefactos de salida, ya que estas acciones suelen ser la última acción de una canalización.

#### `name`

El nombre de artefacto para los artefactos de salida de la acción.

## `configuración` (por proveedor de acciones)

La configuración de la acción contiene detalles y parámetros adecuados para el tipo de proveedor. En la siguiente sección, los parámetros de configuración de la acción de ejemplo son específicos de la acción de origen de S3.

La configuración de la acción y los límites de los input/output artefactos pueden variar según el proveedor de la acción. Para ver una lista de ejemplos de configuración de acciones según el proveedor de acción, consulte Referencia de la estructura de acciones y la tabla en Parámetros de configuración válidos para cada tipo de proveedor. La tabla proporciona un enlace a la referencia de acción para cada tipo de proveedor, en la que se enumeran los parámetros de configuración de cada acción en detalle. Para ver una tabla con los límites de artefactos de entrada y salida para cada proveedor de acción, consulte Artefactos de entrada y salida para cada tipo de acción.

Al trabajar con acciones se deben tener en cuenta los siguientes aspectos:

- Las acciones de origen no tienen artefactos de entrada, y las acciones de implementación no tienen artefactos de salida.
- En el caso de los proveedores de acciones de origen que no utilizan ninguna conexión, como S3, debe usar el parámetro `PollForSourceChanges` para especificar si desea que la canalización se inicie automáticamente cuando se detecte algún cambio. Consulte Configuración válida para el parámetro `PollForSourceChanges`.

- Para configurar la detección de cambios automática a fin de iniciar la canalización o deshabilitar la detección de cambios, consulte Acciones de origen y métodos de detección de cambios.
- Para configurar los desencadenadores con filtrado, utilice la acción de origen para las conexiones y, a continuación, consulte Automatización del inicio de las canalizaciones mediante desencadenadores y filtrado.
- Para ver las variables de salida de cada acción, consulte Referencia de variables.

> **Note**
>
> En el caso de Amazon ECR, Amazon S3 o CodeCommit las fuentes, también puede crear una anulación de fuente utilizando la entrada de transformación de entrada EventBridge para usar la entrada `revisionValue` en el evento de canalización, donde `revisionValue` se deriva de la variable de evento de origen de su clave de objeto, confirmación o ID de imagen. Para obtener más información, consulte el paso opcional para la entrada de transformación de entrada, que se incluye en los procedimientos de Acciones y recursos de origen de Amazon ECR EventBridge, Conexión a acciones de origen de Amazon S3 con un origen habilitado para eventos o CodeCommit acciones de origen y EventBridge.

> **Important**
>
> Las canalizaciones que estén inactivas durante más de 30 días tendrán deshabilitado el sondeo para la canalización. Para obtener más información, consulte el sondeo DisabledAt en la referencia sobre la estructura de canalizaciones. Consulte los pasos para migrar una canalización que usa sondeos para que use la detección de cambios basada en eventos en Métodos de detección de cambios.

> **Note**
>
> Las acciones de origen CodeCommit y de S3 requieren un recurso de detección de cambios configurado (una EventBridge regla) o utilizar la opción para consultar el repositorio en busca de cambios en el origen. En el caso de las canalizaciones con una acción fuente de Bitbucket o GitHub Enterprise Server, no es necesario configurar un webhook ni realizar sondeos de

forma predeterminada. GitHub La acción de conexiones administra la detección de cambios por usted.

## RunOrder

Un entero positivo que indica el orden de ejecución de la acción en la etapa. Las acciones en paralelo de la etapa se muestran con el mismo número entero. Por ejemplo, dos acciones con un orden de ejecución de dos se ejecutarán en paralelo después de que se ejecute la primera acción de la etapa.

El valor predeterminado `runOrder` para una acción es 1. El valor deber ser un entero positivo (número natural). No se pueden usar fracciones, decimales, números negativos ni el número cero Para especificar una secuencia de acciones en serie, utilice el número más pequeño para la primera acción y números mayores para las acciones subsiguientes. Para especificar acciones en paralelo, utilice el mismo entero para cada acción que quiera ejecutar en paralelo. En la consola, puede especificar una secuencia en serie para una acción seleccionando Añadir grupo de acciones en el nivel de la etapa en la que desea que se ejecute, o puede especificar una secuencia paralela seleccionando Añadir acción. Grupo de acciones hace referencia al orden de ejecución de una o más acciones en el mismo nivel.

Por ejemplo, para que tres acciones se ejecuten en secuencia en una etapa, asigne a la primera acción el valor `runOrder` 1, a la segunda acción el valor `runOrder` 2 y a la tercera el valor `runOrder` 3. Si quiere que la segunda y tercera acciones se ejecuten en paralelo, asigne a la primera acción el valor `runOrder` 1 y a la segunda y tercera el valor `runOrder` 2.

> **Note**
>
> La numeración de las acciones en serie no tiene que ser en una secuencia estricta. Por ejemplo, si tiene tres acciones en una secuencia y decide eliminar la segunda, no tiene que reasignar el número del valor `runOrder` de la tercera. Como el valor `runOrder` de dicha acción (3) es mayor que el valor `runOrder` de la primera acción (1), se ejecuta en serie después de la primera acción de la etapa.

# Proveedores de acciones válidos en CodePipeline

El formato de la estructura de la canalización se utiliza para compilar acciones y etapas en una canalización. Un tipo de acción consta de una categoría de acción y un tipo de proveedor.

Cada categoría de acción tiene una lista válida de proveedores de acción. Para hacer referencia a los proveedores de acción válidos para cada categoría de acción, consulte la Referencia de la estructura de acciones.

Cada categoría de acción tiene un conjunto de proveedores designado. Cada proveedor de acción, como Amazon S3, tiene un nombre de proveedor, por ejemplo `S3`, que debe utilizarse en el campo `Provider` en la categoría de acción de la estructura de la canalización.

Hay tres valores válidos para el campo `Owner` en la sección de categoría de acción de la estructura de canalización: `AWS`, `ThirdParty` y `Custom`.

Para encontrar el nombre del proveedor y la información del propietario de su proveedor de acción, consulte la Referencia de la estructura de acciones o Artefactos de entrada y salida para cada tipo de acción.

En esta tabla, se muestran los proveedores válidos para cada tipo de acción.

> **Note**
>
> Para ver las acciones de Bitbucket o GitHub Enterprise Server, consulta el tema de referencia sobre las CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas acciones. GitHub

Proveedores de acciones válidos por tipo de acción

| Categoría de la acción | Proveedores válidos de la acción | Tipo de canalización compatible | Referencia de la acción |
|---|---|---|---|
| origen | Amazon S3 | V1, V2 | Referencia sobre la acción de origen de Amazon S3 |

| Categoría de la acción | Proveedores válidos de la acción | Tipo de canalización compatible | Referencia de la acción |
|---|---|---|---|
| | Amazon ECR | V1, V2 | Referencia de acciones de origen de Amazon ECR |
| | CodeCommit | V1, V2 | CodeCommit referencia de acción de origen |
| | CodeStarSourceConnection (para Bitbucket GitHub, las acciones de GitHub Enterprise Server) | V1, V2 | CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas |
| Build | Acción `ECRBuildAndPublish` de Amazon ECR | V2 únicamente | Referencia de la acción de compilación `ECRBuildAndPublish` |
| | CodeBuild | V1, V2 | AWS CodeBuild referencia de acciones de creación y prueba |
| | Acción de comandos (consulte Computación) | V2 únicamente | |
| | Personalizado CloudBees | V1, V2 | Artefactos de entrada y salida para cada tipo de acción |
| | Custom Jenkins | V1, V2 | Artefactos de entrada y salida para cada tipo de acción |

| Categoría de la acción | Proveedores válidos de la acción | Tipo de canalización compatible | Referencia de la acción |
|---|---|---|---|
| | Personalizado TeamCity | V1, V2 | Artefactos de entrada y salida para cada tipo de acción |
| Test | CodeBuild | V1, V2 | AWS CodeBuild referencia de acciones de creación y prueba |
| | AWS Device Farm | V1, V2 | Artefactos de entrada y salida para cada tipo de acción |
| | Personalizado BlazeMeter | V1, V2 | Artefactos de entrada y salida para cada tipo de acción |
| | ThirdParty GhostInspector | | Artefactos de entrada y salida para cada tipo de acción |
| | Custom Jenkins | | Artefactos de entrada y salida para cada tipo de acción |
| | ThirdParty StormRunner Carga de microenfoque | | Artefactos de entrada y salida para cada tipo de acción |
| | ThirdParty Nouvola | | Artefactos de entrada y salida para cada tipo de acción |
| | ThirdParty Runscope | | Artefactos de entrada y salida para cada tipo de acción |

| Categoría de la acción | Proveedores válidos de la acción | Tipo de canalización compatible | Referencia de la acción |
|---|---|---|---|
| Implementación | Amazon S3 | | Referencia de acción de implementación de Amazon S3 |
| | CloudFormation | | Referencia de la acción de implementación de CloudFormation |
| | CodeDeploy | | Artefactos de entrada y salida para cada tipo de acción |
| | Acción de implementación de EC2 | V2 únicamente | Referencia de la acción de Amazon EC2 |
| | Amazon ECS | | Artefactos de entrada y salida para cada tipo de acción |
| | Amazon ECS (Blue/Green) (esta es la `CodeDeployToECS` acción) | | Artefactos de entrada y salida para cada tipo de acción |
| | Acción de Amazon EKS | V2 únicamente | Referencia de acción de implementación de Amazon Elastic Kubernetes Service EKS |
| | Elastic Beanstalk | | Artefactos de entrada y salida para cada tipo de acción |
| | AWS AppConfig | | Referencia de la acción de implementación de AWS AppConfig |

| Categoría de la acción | Proveedores válidos de la acción | Tipo de canalización compatible | Referencia de la acción |
|---|---|---|---|
| | OpsWorks | | Artefactos de entrada y salida para cada tipo de acción |
| | Service Catalog | | Artefactos de entrada y salida para cada tipo de acción |
| | Amazon Alexa | | Artefactos de entrada y salida para cada tipo de acción |
| | Personalizado XebiaLabs | | Artefactos de entrada y salida para cada tipo de acción |
| Aprobación | Manual | | Artefactos de entrada y salida para cada tipo de acción |
| Invocación | CodePipelineInvocar la acción | | AWS CodePipeline invocar referencia a la acción |
| | AWS Lambda | | AWS Lambda invocar referencia a la acción |
| | AWS Step Functions | | AWS Step Functions invocar referencia de acción |
| | InspectorScan | | Referencia de la acción de invocación de `InspectorScan` de Amazon Inspector |

| Categoría de la acción | Proveedores válidos de la acción | Tipo de canalización compatible | Referencia de la acción |
|---|---|---|---|
| Computación | Acción de Comandos | | Referencia de la acción de Comandos |

Algunos tipos de acción solo CodePipeline están disponibles en AWS regiones seleccionadas. Es posible que un tipo de acción esté disponible en una AWS región, pero no AWS haya un proveedor para ese tipo de acción.

Para obtener más información sobre cada uno de los proveedores de acciones, consulte Integraciones con tipos de CodePipeline acciones.

## Configuración válida para el parámetro `PollForSourceChanges`

El valor predeterminado del parámetro `PollForSourceChanges` lo determina el método utilizado para crear la canalización, tal y como se describe en la tabla siguiente. En muchos casos, el valor predeterminado del parámetro `PollForSourceChanges` es true y se debe deshabilitar.

Si el valor predeterminado del parámetro `PollForSourceChanges` es true, haga lo siguiente:

- Agregue el parámetro `PollForSourceChanges` al archivo JSON o la plantilla de CloudFormation .
- Cree recursos de detección de cambios (regla de CloudWatch eventos, según corresponda).
- Establecer el parámetro `PollForSourceChanges` en false.

> **Note**
>
> Si creas una regla de CloudWatch eventos o un webhook, debes establecer el parámetro en falso para evitar que se active la canalización más de una vez.

El parámetro `PollForSourceChanges` no se utiliza para las acciones de origen de Amazon ECR.

- Valores predeterminados del parámetro **`PollForSourceChanges`**

| origen | Método de creación | Ejemplo de salida de la estructura JSON de "configuration" |
| --- | --- | --- |
| CodeCommit | La canalización se crea con la consola (y los recursos de detección de cambios los crea la consola). El parámetro se muestra en la salida de la estructura de la canalización y tiene el valor predeterminado `false`. | `BranchName": "main", "PollForSourceChanges": "false", "RepositoryName": "my-repo"` |
| | La canalización se crea con la CLI o CloudFormation, y el `PollForSourceChanges` parámetro no se muestra en la salida JSON, pero se establece en .² `true` | `BranchName": "main", "RepositoryName": "my-repo"` |
| Amazon S3 | La canalización se crea con la consola (y los recursos de detección de cambios los crea la consola). El parámetro se muestra en la salida de la estructura de la canalización y tiene el valor predeterminado `false`. | `"S3Bucket": "my-bucket", "S3ObjectKey": "object.zip", "PollForSourceChanges": "false"` |
| | La canalización se crea con la CLI o CloudFormation, y el `PollForSourceChanges` parámetro no se muestra en la salida JSON, pero se establece en .² `true` | `"S3Bucket": "my-bucket", "S3ObjectKey": "object.zip"` |
| GitHub | La canalización se crea con la consola (y los recursos de detección de cambios los crea la consola). El parámetro se muestra en la salida de la estructura de la canalización y tiene el valor predeterminado `false`. | `"Owner": "MyGitHubAccountName ", "Repo": " MyGitHubRepositoryName " "PollForSourceChanges": "false", "Branch": " main" "OAuthToken": " ****"` |

| origen | Método de creación | Ejemplo de salida de la estructura JSON de "configuration" |
| --- | --- | --- |
| | La canalización se crea con la CLI o CloudFormation, y el `PollForSourceChanges` parámetro no se muestra en la salida JSON, pero se establece en .² `true` | `"Owner": "MyGitHubAccountName ", "Repo": "MyGitHubRepositoryName ", "Branch": " main", "OAuthToken": " ****"` |

² Si se `PollForSourceChanges` ha agregado en algún momento a la estructura JSON o a la CloudFormation plantilla, se muestra como se muestra a continuación:

```
"PollForSourceChanges": "true",
```

Para obtener más información sobre los recursos de detección de cambios que se aplican a cada proveedor de origen, consulte Métodos de detección de cambios.

## Artefactos de entrada y salida para cada tipo de acción

Según el proveedor y el tipo de acción, puede tener la cantidad de artefactos de entrada y salida que se indican a continuación.

Restricciones de artefactos por tipo de acción

| Propietario | Tipo de acción | Proveedor | Número válido de artefactos de entrada | Número válido de artefactos de salida |
| --- | --- | --- | --- | --- |
| AWS | origen | S3 | 0 | 1 |
| AWS | origen | CodeCommit | 0 | 1 |
| AWS | origen | ECR | 0 | 1 |

| Propietario | Tipo de acción | Proveedor | Número válido de artefactos de entrada | Número válido de artefactos de salida |
|---|---|---|---|---|
| ThirdParty | origen | CodeStarSourceConnection | 0 | 1 |
| AWS | Build | CodeBuild | De 1 a 5 | De 0 a 5 |
| AWS | Test | CodeBuild | De 1 a 5 | De 0 a 5 |
| AWS | Test | DeviceFarm | 1 | 0 |
| AWS | Aprobación | ThirdParty | 0 | 0 |
| AWS | Implementación | S3 | 1 | 0 |
| AWS | Implementación | CloudFormation | De 0 a 10 | De 0 a 1 |
| AWS | Implementación | CodeDeploy | 1 | 0 |
| AWS | Implementación | ElasticBeanstalk | 1 | 0 |
| AWS | Implementación | OpsWorks | 1 | 0 |
| AWS | Implementación | ECS | 1 | 0 |
| AWS | Implementación | ServiceCatalog | 1 | 0 |
| AWS | Invocación | Lambda | De 0 a 5 | De 0 a 5 |
| ThirdParty | Implementación | AlexaSkillsKit | De 1 a 2 | 0 |
| Custom | Build | Jenkins | De 0 a 5 | De 0 a 5 |

| Propietario | Tipo de acción | Proveedor | Número válido de artefactos de entrada | Número válido de artefactos de salida |
|---|---|---|---|---|
| `Custom` | Test | `Jenkins` | De 0 a 5 | De 0 a 5 |
| `Custom` | Cualquier categoría compatible | Según se indique en la acción personalizada | De 0 a 5 | De 0 a 5 |

## Parámetros de configuración válidos para cada tipo de proveedor

En esta sección, se incluyen los parámetros de la sección `configuration` válidos para cada proveedor de acción.

Cada acción debe tener una configuración de acción válida que depende del tipo de proveedor de dicha acción. En la tabla siguiente se incluyen los elementos de configuración de acción necesarios para cada tipo de proveedor válido:

Propiedades de configuración de la acción de los tipos de proveedores

| Nombre del proveedor | Nombre del proveedor del tipo de acción | Propiedades de configuración | Required/Optional |
|---|---|---|---|
| Amazon S3 (proveedor de acciones de implementación) | Para obtener más información, incluidos ejemplos relacionados con los parámetros de acción de implementación de Amazon S3, consulte Referencia de acción de implementación de Amazon S3. | | |
| Amazon S3 (proveedor de acciones de fuente) | Para obtener más información, incluidos ejemplos relacionados con los parámetros de acción de fuente de Amazon S3, consulte Referencia sobre la acción de origen de Amazon S3. | | |

| Nombre del proveedor | Nombre del proveedor del tipo de acción | Propiedades de configuración | Required/Optional |
|---|---|---|---|
| Amazon ECR | Para obtener más información, incluidos ejemplos relacionados con los parámetros de Amazon ECR, consulte Referencia de acciones de origen de Amazon ECR. | | |
| CodeCommit | Para obtener más información, incluidos ejemplos relacionados con CodeCommit los parámetros, consulteCodeCommit referencia de acción de origen. | | |
| CodeStarSourceConnection action for Bitbucket GitHub (mediante una GitHub aplicación), GHES y GitLab | Para obtener más información, incluidos ejemplos de la configuración de la acción, consulte Parámetros de configuración. | | |
| GitHub (a través de la aplicación OAuth) | Para obtener más información, incluidos ejemplos relacionados con los GitHub parámetros, consulte. GitHub (a través de la aplicación OAuth): referencia de la acción fuente Esta es la GitHub acción de la versión 1. | | |
| AWS CloudFormation | Para obtener más información, incluidos ejemplos relacionados con AWS CloudFormation los parámetros, consulteReferencia de la acción de implementación de CloudFormation. | | |
| CodeBuild | Para obtener más descripción y ejemplos relacionados con CodeBuild los parámetros, consulteAWS CodeBuild referencia de acciones de creación y prueba. | | |
| CodeDeploy | Para obtener más descripción y ejemplos relacionados con CodeDeploy los parámetros, consulteReferencia de la acción de implementación de AWS CodeDeploy. | | |

| Nombre del proveedor | Nombre del proveedor del tipo de acción | Propiedades de configuración | Required/Optional |
|---|---|---|---|
| AWS Device Farm | Para obtener más descripción y ejemplos relacionados con AWS Device Farm los parámetros, consulteAWS Device Farm referencia de acción de prueba. | | |
| AWS Elastic Beanstalk | `ElasticBeanstalk` | `ApplicationName` | Obligatorio |
| | | `EnvironmentName` | Obligatorio |
| AWS Lambda | Para obtener más información, incluidos ejemplos relacionados con AWS Lambda los parámetros, consulteAWS Lambda invocar referencia a la acción. | | |
| AWS OpsWorks Stacks | `OpsWorks` | `Stack` | Obligatorio |
| | | `Layer` | Opcional |
| | | `App` | Obligatorio |
| Amazon ECS | Para ver más descripciones y ejemplos relacionados con los parámetros de Amazon ECS, consulte Referencia de acción de implementación de Amazon Elastic Container Service. | | |
| Amazon ECS y CodeDeploy (Blue/Green) | Para obtener más descripciones y ejemplos relacionados con Amazon ECS y sus CodeDeploy blue/green parámetros, consulteReferencia de acciones de implementación de Amazon Elastic Container Service y CodeDeploy azul-verde. | | |
| Service Catalog | `ServiceCatalog` | `TemplateFilePath` | Obligatorio |
| | | `ProductVersionName` | Obligatorio |
| | | `ProductType` | Obligatorio |
| | | `ProductVersionDescription` | Opcional |
| | | `ProductId` | Obligatorio |

| Nombre del proveedor | Nombre del proveedor del tipo de acción | Propiedades de configuración | Required/Optional |
|---|---|---|---|
| Alexa Skills Kit | `AlexaSkillsKit` | `ClientId` | Obligatorio |
| | | `ClientSecret` | Obligatorio |
| | | `RefreshToken` | Obligatorio |
| | | `SkillId` | Obligatorio |
| Jenkins | El nombre de la acción que especificó en el CodePipeline complemento para Jenkins (por ejemplo, *`MyJenkinsProviderName`*) | `ProjectName` | Obligatorio |
| Aprobación manual | `Manual` | `CustomData` | Opcional |
| | | `ExternalEntityLink` | Opcional |
| | | `NotificationArn` | Opcional |

En el ejemplo siguiente, se muestra una configuración válida para una acción de implementación que utiliza Alexa Skills Kit:

```
"configuration": {
  "ClientId": "amzn1.application-oa2-client.aadEXAMPLE",
  "ClientSecret": "****",
  "RefreshToken": "****",
  "SkillId": "amzn1.ask.skill.22649d8f-0451-4b4b-9ed9-bfb6cEXAMPLE"
}
```

En el siguiente ejemplo, se muestra una configuración válida para una aprobación manual:

```
"configuration": {
```

```
  "CustomData": "Comments on the manual approval",
  "ExternalEntityLink": "http://my-url.com",
  "NotificationArn": "arn:aws:sns:us-west-2:12345EXAMPLE:Notification"
}
```

# Referencia de la estructura de acciones

En esta sección se hace referencia únicamente a la configuración de acciones. Para obtener información general conceptual sobre la estructura de la canalización, consulte CodePipeline referencia de estructura de tuberías.

Cada proveedor de acciones CodePipeline utiliza un conjunto de campos de configuración obligatorios y opcionales en la estructura de la canalización. En esta sección se proporciona la siguiente información de referencia según el proveedor de acciones:

- Valores válidos para los campos `ActionType` incluidos en el bloque de acción de estructura de canalización, como, por ejemplo, `Owner` y `Provider`.
- Descripciones y otra información de referencia sobre los parámetros `Configuration` (obligatorios y opcionales) incluidos en la sección de acción de estructura de canalización.
- Campos de acción JSON y YAML de ejemplo válidos.

Esta sección se actualiza periódicamente con más proveedores de acciones. La información de referencia está disponible actualmente para los siguientes proveedores de acciones:

Temas


- Referencia de la acción de Amazon EC2
- Referencia de acciones de origen de Amazon ECR
- Referencia de la acción de compilación ECRBuildAndPublish
- Referencia de acciones de implementación de Amazon Elastic Container Service y CodeDeploy azul-verde
- Referencia de acción de implementación de Amazon Elastic Container Service
- Referencia de acción de implementación de Amazon Elastic Kubernetes Service EKS
- AWS Lambda Referencia de la acción de implementación de
- Referencia de acción de implementación de Amazon S3
- Referencia sobre la acción de origen de Amazon S3
- Referencia de la acción de implementación de AWS AppConfig
- Referencia de la acción de implementación de CloudFormation
- CloudFormation Referencia de la acción de implementación de StackSets
- AWS CodeBuild referencia de acciones de creación y prueba

- AWS CodePipeline invocar referencia a la acción
- CodeCommit referencia de acción de origen
- Referencia de la acción de implementación de AWS CodeDeploy
- CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas
- Referencia de la acción de Comandos
- AWS Device Farm referencia de acción de prueba
- Referencia de la acción de implementación de Elastic Beanstalk
- Referencia de la acción de invocación de InspectorScan de Amazon Inspector
- AWS Lambda invocar referencia a la acción
- AWS OpsWorks Referencia de la acción de implementación de
- AWS Service Catalog Referencia de la acción de implementación de
- AWS Step Functions invocar referencia de acción

## Referencia de la acción de Amazon EC2

Una acción EC2 de Amazon EC2 se utiliza para implementar el código de aplicación en su flota de implementación. Su flota de implementación puede estar formada por instancias Linux o SSM-managed nodos Linux de Amazon EC2. Las instancias deben tener instalado SSM Agent.

> **Note**
>
> Esta acción admite solo tipos de instancias de Linux. El tamaño de flota máximo admitido son 500 instancias.

La acción seleccionará un número de instancias en función de un número máximo especificado. Se elegirán primero las instancias anteriores que hayan fallado. La acción omitirá la implementación en determinadas instancias en caso de que la instancia ya haya recibido la implementación del mismo artefacto de entrada, como en el caso de que la acción hubiese fallado anteriormente.

> **Note**
>
> Esta acción solo se admite para canalizaciones de tipo V2.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Permisos de la política de rol de servicio para la acción de implementación de EC2
- Referencia del archivo de especificaciones de implementación
- Declaración de acciones
- Ejemplo de declaración de acción con especificaciones de implementación
- Véase también


## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `EC2`
- Versión: `1`

## Parámetros de configuración

InstanceTagKey

Obligatorio: sí

La clave de etiqueta de las instancias que creó en Amazon EC2, como `Name`.

InstanceTagValue

Obligatorio: no

El valor de etiqueta de las instancias que creó en Amazon EC2, como `my-instances`.

Si no se especifica este valor, se InstanceTagKey compararán todas las instancias con.

InstanceType

Obligatorio: sí

El tipo de instancias o nodos de SSM creados en Amazon EC2. Los valores válidos son `EC2` y `SSM_MANAGED_NODE`.

Debe haber creado, etiquetado e instalado SSM Agent en todas las instancias.

> **Note**
>
> Al crear la instancia, se crea un rol de instancia de EC2 o se utiliza uno existente. Para evitar `Access Denied` errores, debes añadir los permisos del bucket de S3 al rol de la instancia para conceder a la instancia los permisos del bucket del CodePipeline artefacto. Cree un rol predeterminado o actualice el rol actual con el permiso `s3:GetObject` limitado al bucket de artefactos correspondiente a la región de su canalización.

TargetDirectory

Obligatorio: sí (en caso de que se especifique script)

El directorio que se debe usar en la instancia de Amazon EC2 para ejecutar scripts.

DeploySpec

Obligatorio: sí (en caso de que se definan especificaciones de implementación)

El archivo que se utilizará para configurar los eventos de instalación y ciclo de vida de la implementación. Para obtener información y las descripciones de los campos de especificaciones de implementación, consulte Referencia del archivo de especificaciones de implementación. Para ver la configuración de una acción con el archivo de especificaciones de implementación indicado, consulte el ejemplo de Ejemplo de declaración de acción con especificaciones de implementación.

MaxBatch

Obligatorio: no

El número máximo de instancias que se permite implementar en paralelo.

MaxError

Obligatorio: no

El número máximo de errores de instancia permitidos durante la implementación.

TargetGroupNameList

Obligatorio: no

La lista de nombres de grupos de destino para la implementación. Los grupos de destino deben estar ya creados.

Los grupos de destino proporcionan un conjunto de instancias para procesar solicitudes específicas. Si se especifica el grupo de destino, las instancias se eliminarán del grupo de destino antes de la implementación y se volverán a agregar al grupo de destino después de la implementación.

PreScript

Obligatorio: no

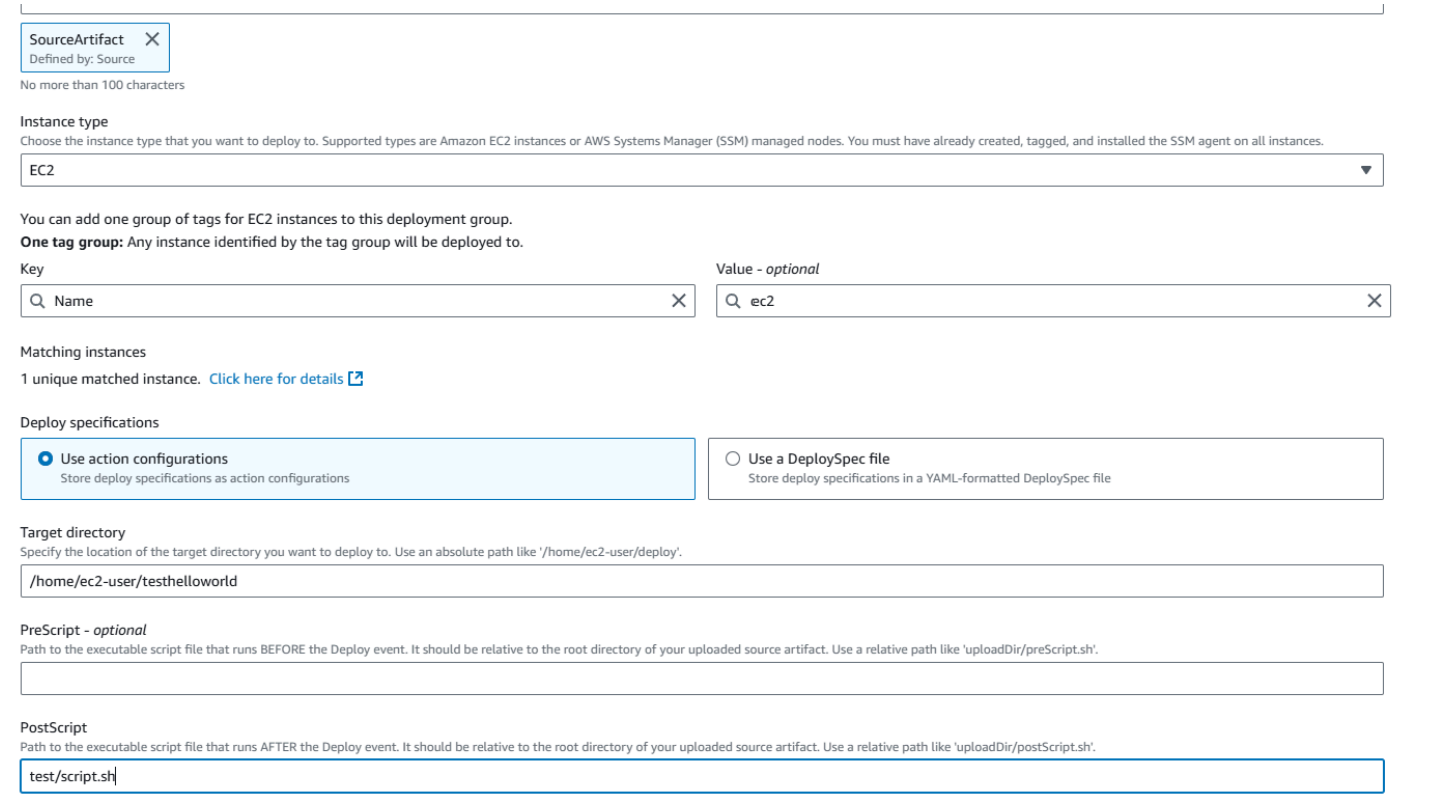
El script que se ejecutará antes de la fase de implementación de la acción.

PostScript

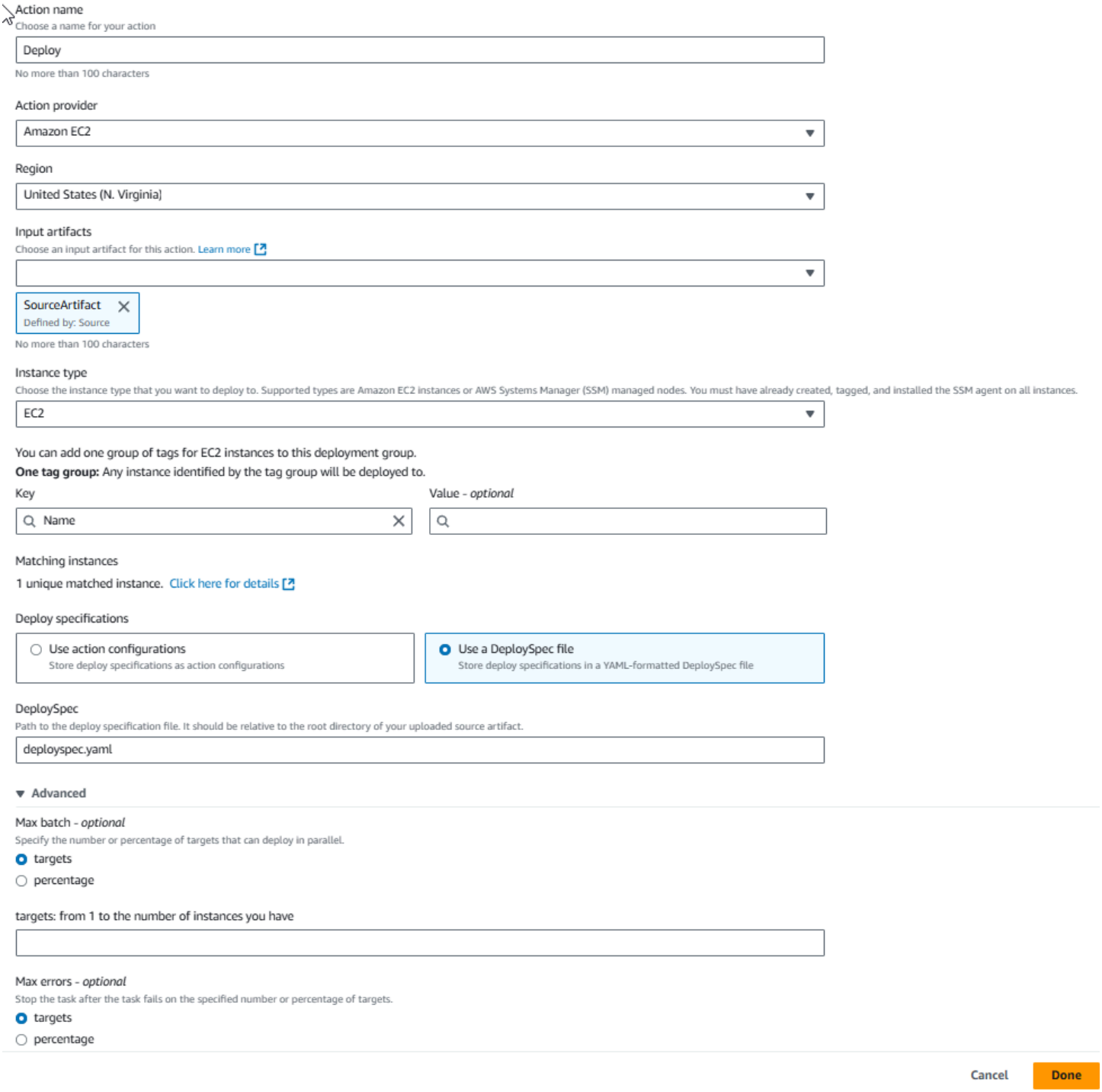
Obligatorio: sí

El script que se ejecutará después de la fase de implementación de la acción.

La siguiente imagen muestra un ejemplo de la página Editar de la acción en la que se elige Usar la configuración de la acción.




La siguiente imagen muestra un ejemplo de la página de edición de la acción en la que se elige Usar un DeploySpec archivo.




# Artefactos de entrada

- Número de artefactos: 1
- Descripción: los archivos proporcionados (en caso de haberlos) para respaldar las acciones del script durante la implementación.

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Permisos de la política de rol de servicio para la acción de implementación de EC2

Cuando CodePipeline se ejecuta la acción, el rol de CodePipeline servicio requiere los siguientes permisos, con un alcance adecuado para el acceso con el mínimo privilegio.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "StatementWithAllResource",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "elasticloadbalancing:DescribeTargetGroupAttributes",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth",
                "ssm:CancelCommand",
                "ssm:DescribeInstanceInformation",
                "ssm:ListCommandInvocations"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "StatementForLogs",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
```

```
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:/aws/codepipeline/
{{pipelineName}}:*"
            ]
        },
        {
            "Sid": "StatementForElasticloadbalancing",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": [
                "arn:aws:elasticloadbalancing:us-east-1:111122223333:targetgroup/
[[targetGroupName]]/*"
            ]
        },
        {
            "Sid": "StatementForSsmOnTaggedInstances",
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ec2:us-east-1:111122223333:instance/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/{{tagKey}}": "{{tagValue}}"
                }
            }
        },
        {
            "Sid": "StatementForSsmApprovedDocuments",
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1::document/AWS-RunPowerShellScript",
                "arn:aws:ssm:us-east-1::document/AWS-RunShellScript"
            ]
        }
    ]
```

```
}
```

## Registra los grupos de tu canalización en los CloudWatch registros

Cuando CodePipeline se ejecuta la acción, CodePipeline crea un grupo de registros con el nombre de la canalización de la siguiente manera. Esto permite reducir los permisos para registrar los recursos mediante el nombre de la canalización.

```
/aws/codepipeline/MyPipelineName
```

Los siguientes permisos de registro se incluyen en las actualizaciones anteriores para el rol de servicio.

- registros: CreateLogGroup
- registros: CreateLogStream
- registros: PutLogEvents

Para ver los registros en la consola mediante la página del cuadro de diálogo de detalles de la acción, se debe agregar el permiso para ver los registros al rol de la consola. Para obtener más información, consulte el ejemplo de política de permisos para consolas en Permisos necesarios para ver los registros de procesamiento en la consola.

## Política de funciones de servicio: permisos para CloudWatch los registros

Cuando CodePipeline se ejecuta la acción, CodePipeline crea un grupo de registros con el nombre de la canalización de la siguiente manera. Esto permite reducir los permisos para registrar los recursos mediante el nombre de la canalización.

```
/aws/codepipeline/MyPipelineName
```

Para ver los registros en la consola mediante la página del cuadro de diálogo de detalles de la acción, se debe agregar el permiso para ver los registros al rol de la consola. Para obtener más información, consulte el ejemplo de política de permisos para consolas en Permisos necesarios para ver los registros de procesamiento en la consola.

# Referencia del archivo de especificaciones de implementación

Cuando CodePipeline se ejecuta la acción, puedes especificar un archivo de especificaciones para configurar la implementación en tus instancias. El archivo de especificaciones de implementación especifica qué se debe instalar y qué enlaces de eventos del ciclo de vida se deben ejecutar como respuesta a los eventos del ciclo de vida de implementación. El archivo de especificaciones de implementación siempre está ahí. YAML-formatted El archivo de especificaciones de implementación se utiliza para:

- Asignar los archivos de origen de la revisión de la aplicación a sus destinos en la instancia.
- Especificar permisos personalizados para los archivos implementados.
- Especificar los scripts que se van a ejecutar en cada instancia en las distintas fases del proceso de implementación.

El archivo de especificaciones de implementación admite los parámetros de configuración de implementación específicos que admite el CodeDeploy AppSpec archivo. Puede usar el AppSpec archivo existente directamente y se ignorarán los parámetros no admitidos. Para obtener más información sobre el AppSpec archivo en CodeDeploy, consulte la referencia al archivo de especificación de la aplicación en la Guía del CodeDeploy usuario.

Los parámetros de implementación del archivo se especifican como se indica a continuación.

- `files`: el archivo de especificaciones de implementación designa los valores de `source:` y `destination:` para los archivos de implementación.
- `scripts`: los eventos incluidos en los scripts para la implementación. Se admiten dos eventos: `BeforeDeploy` y `AfterDeploy`.
- `hooks`: los enlaces de ciclo de vida del evento. Se admiten los siguientes enlaces: `ApplicationStop`, `BeforeInstall`, `AfterInstall`, `ApplicationStart` y `ValidateService`.

> **Note**
>
> El parámetro hooks está disponible para garantizar la AppSpec compatibilidad con la versión 0.0 (AppSpec formato) CodeDeploy y solo está disponible en ella. Para este formato, CodePipeline realizará un mapeo de los eventos haciendo un mejor esfuerzo.

El archivo de especificaciones debe emplear los espacios como requiere YAML. Se producirá un error si las posiciones y el número de los espacios del archivo de especificaciones de implementación no son correctos. Para obtener más información acerca del espaciado, consulte la especificación de YAML.

A continuación, se muestra un ejemplo de archivo de especificaciones de implementación.

```
version: 0.1
files:
  - source: /index.html
    destination: /var/www/html/
scripts:
  BeforeDeploy:
    - location: scripts/install_dependencies
      timeout: 300
      runas: myuser
  AfterDeploy:
    - location: scripts/start_server
      timeout: 300
      runas: myuser
```

Para ver la configuración de una acción con el archivo de especificaciones de implementación indicado, consulte el ejemplo de Ejemplo de declaración de acción con especificaciones de implementación.

## Declaración de acciones

YAML

```
name: DeployEC2
actions:
- name: EC2
  actionTypeId:
    category: Deploy
    owner: AWS
    provider: EC2
    version: '1'
  runOrder: 1
  configuration:
    InstanceTagKey: Name
    InstanceTagValue: my-instances
    InstanceType: EC2
```

```
    PostScript: "test/script.sh",
    TargetDirectory: "/home/ec2-user/deploy"
  outputArtifacts: []
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
```

JSON

```
{
    "name": "DeployEC2",
    "actions": [
        {
            "name": "EC2Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "EC2",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "InstanceTagKey": "Name",
                "InstanceTagValue": "my-instances",
                "InstanceType": "EC2",
                "PostScript": "test/script.sh",
                "TargetDirectory": "/home/ec2-user/deploy"
            },
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1"
        }
    ]
},
```

# Ejemplo de declaración de acción con especificaciones de implementación

## YAML

```
name: DeployEC2
actions:
- name: EC2
  actionTypeId:
    category: Deploy
    owner: AWS
    provider: EC2
    version: '1'
  runOrder: 1
  configuration:
    DeploySpec: "deployspec.yaml"
    InstanceTagKey: Name
    InstanceTagValue: my-instances
    InstanceType: EC2
  outputArtifacts: []
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
```

## JSON

```
{
    "name": "DeployEC2",
    "actions": [
        {
            "name": "EC2Deploy",
            "actionTypeId": {
                "category": "Deploy",
                "owner": "AWS",
                "provider": "EC2",
                "version": "1"
            },
            "runOrder": 1,
            "configuration": {
                "DeploySpec": "deployspec.yaml",
                "InstanceTagKey": "Name",
                "InstanceTagValue": "my-instances",
                "InstanceType": "EC2"
            },
```

```
            "outputArtifacts": [],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1"
        }
    ]
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Tutorial: Implemente en instancias de Amazon EC2 con CodePipeline— En este tutorial se explica cómo crear instancias de EC2 en las que se desplegará un archivo de script, además de cómo crear la canalización mediante la acción EC2.
- La acción de despliegue de EC2 falla y muestra el mensaje de error `No existe tal archivo`. En este tema, se ofrecen soluciones para los problemas relacionados con errores de archivos no encontrados con la acción de EC2.

# Referencia de acciones de origen de Amazon ECR

Activa la canalización cuando se envía una nueva imagen al repositorio de Amazon ECR. Esta acción proporciona un archivo de definiciones de imagen que hace referencia al URI de la imagen que se ha enviado a Amazon ECR. Esta acción de origen se utiliza a menudo junto con otra acción de origen, como CodeCommit, para permitir una ubicación de origen para todos los demás artefactos de origen. Para obtener más información, consulte Tutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR.

Cuando usas la consola para crear o editar tu canalización, CodePipeline crea una EventBridge regla que inicia tu canalización cuando se produce un cambio en el repositorio.

> **Note**
>
> En el caso de Amazon ECR, Amazon S3 o CodeCommit las fuentes, también puedes crear una anulación de fuente utilizando la entrada de transformación de entrada EventBridge para

usar la entrada `revisionValue` en el evento de canalización, donde `revisionValue` se deriva de la variable de evento de origen de tu clave de objeto, confirmación o ID de imagen. Para obtener más información, consulte el paso opcional para la entrada de transformación de entrada, que se incluye en los procedimientos de Acciones y recursos de origen de Amazon ECR EventBridge, Conexión a acciones de origen de Amazon S3 con un origen habilitado para eventos o CodeCommit acciones de origen y EventBridge.

Ya debe haber creado un repositorio de Amazon ECR y enviado una imagen antes de conectar la canalización a través de una acción de Amazon ECR.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Permisos del rol de servicio: acción de Amazon ECR
- Declaración de acción (ejemplo de Amazon ECR)
- Véase también


## Tipo de acción

- Categoría: `Source`
- Propietario: `AWS`
- Proveedor: `ECR`
- Versión: `1`

## Parámetros de configuración

RepositoryName

Obligatorio: sí

El nombre del repositorio de Amazon ECR al que se envió la imagen.

ImageTag

Obligatorio: no

La etiqueta utilizada para la imagen.

> **Note**
>
> Si no se especifica un valor para `ImageTag`, el valor predeterminado es `latest`.

## Artefactos de entrada

- Número de artefactos: `0`
- Descripción: los artefactos de entrada no se aplican a este tipo de acción.

## Artefactos de salida

- Número de artefactos: `1`
- Descripción: esta acción produce un artefacto que contiene un archivo `imageDetail.json` que contiene el URI de la imagen que desencadenó la ejecución de la canalización. Para obtener más información sobre el archivo `imageDetail.json`, consulte Detail.json archivo de imagen para las acciones de blue/green implementación de Amazon ECS.

## Variables de salida

Cuando se configura, esta acción produce variables a las que se puede hacer referencia mediante la configuración de acción de una acción descendente en la canalización. Esta acción produce variables que se pueden ver como variables de salida, incluso si la acción no tiene un espacio de nombres. Configure una acción con un espacio de nombres para que esas variables estén disponibles para la configuración de las acciones posteriores.

Para obtener más información, consulte Referencia de variables.

RegistryId

El ID de AWS cuenta asociado al registro que contiene el repositorio.

RepositoryName

El nombre del repositorio de Amazon ECR al que se envió la imagen.

ImageTag

La etiqueta utilizada para la imagen.

> **Note**
>
> La variable `ImageTag` de salida no se genera cuando se anula la revisión de origen.

ImageDigest

El resumen `sha256` del manifiesto de la imagen.

ImageURI

El URI de la imagen.

## Permisos del rol de servicio: acción de Amazon ECR

Para admitir Amazon ECR, añada lo siguiente a su instrucción de política:

```
{
    "Effect": "Allow",
    "Action": [
        "ecr:DescribeImages"
    ],
    "Resource": "resource_ARN"
},
```

Para obtener más información sobre esta acción, consulte Referencia de acciones de origen de Amazon ECR.

## Declaración de acción (ejemplo de 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"
        }
    ]
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Tutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR— Este tutorial proporciona un AppSpec archivo de ejemplo y un grupo de CodeDeploy aplicaciones y despliegues de ejemplo para crear una canalización con una CodeCommit fuente de Amazon ECR que se implemente en las instancias de Amazon ECS.

# Referencia de la acción de compilación **ECRBuildAndPublish**

Esta acción de compilación permite automatizar la compilación e inserción de una nueva imagen cuando se produce un cambio en el origen. Esta acción se basa en una ubicación específica de Dockerfile y empuja la imagen. Esta acción de creación no es lo mismo que la acción de código fuente de Amazon ECR CodePipeline, que activa la canalización cuando se produce un cambio en el repositorio de código fuente de Amazon ECR. Para obtener información acerca de esa acción, consulte Referencia de acciones de origen de Amazon ECR.

No se trata de una acción de origen que vaya a desencadenar la canalización. Esta acción compila una imagen y la inserta en el repositorio de imágenes de Amazon ECR.

Debes haber creado ya un repositorio de Amazon ECR y haber añadido un Dockerfile a tu repositorio de código fuente, por ejemplo GitHub, antes de añadir la acción a tu canalización.

> **Important**
>
> Esta acción usa CodeBuild computación CodePipeline gestionada para ejecutar comandos en un entorno de compilación. Si ejecuta la acción de Comandos, se le cobrarán cargos por separado en AWS CodeBuild.

> **Note**
>
> Esta acción solo está disponible para canalizaciones de tipo V2.

Temas

- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Permisos del rol de servicio: acción ECRBuildAndPublish
- Declaración de acciones
- Véase también


## Tipo de acción

- Categoría: `Build`
- Propietario: `AWS`
- Proveedor: `ECRBuildAndPublish`
- Versión: `1`

## Parámetros de configuración

ECRRepositoryName

Obligatorio: sí

Nombre del repositorio de Amazon ECR donde se inserta la imagen.

DockerFilePath

Obligatorio: no

La ubicación del Dockerfile utilizado para crear la imagen. Si lo desea, puede proporcionar una ubicación de Dockerfile alternativa si no está en el nivel raíz.

> **Note**
>
> Si no se especifica ningún valor para `DockerFilePath`, el valor predeterminado es el nivel raíz del repositorio de origen.

ImageTags

Obligatorio: no

Etiquetas utilizadas para la imagen. Puede introducir varias etiquetas como una lista de cadenas delimitadas por comas.

> **Note**
>
> Si no se especifica un valor para `ImageTags`, el valor predeterminado es `latest`.

RegistryType

Obligatorio: no

Especifica si el repositorio es público o privado. Los valores válidos son `private | public`.

> **Note**
>
> Si no se especifica un valor para `RegistryType`, el valor predeterminado es `private`.

## Artefactos de entrada

- Número de artefactos: `1`
- Descripción: el artefacto generado por la acción de origen que contiene el Dockerfile necesario para compilar la imagen.

## Artefactos de salida

- Número de artefactos: `0`

## Variables de salida

Cuando se configura, esta acción produce variables a las que se puede hacer referencia mediante la configuración de acción de una acción descendente en la canalización. Esta acción produce

variables que se pueden ver como variables de salida, incluso si la acción no tiene un espacio de nombres. Configure una acción con un espacio de nombres para que esas variables estén disponibles para la configuración de las acciones posteriores.

Para obtener más información, consulte Referencia de variables.

ECRImageDigestId

El resumen `sha256` del manifiesto de la imagen.

ECRRepositoryName

El nombre del repositorio de Amazon ECR al que se envió la imagen.

## Permisos del rol de servicio: acción **`ECRBuildAndPublish`**

Para la compatibilidad con la acción `ECRBuildAndPublish`, agregue lo siguiente a su declaración de política:

```
{
    "Statement": [
         {
            "Sid": "ECRRepositoryAllResourcePolicy",
            "Effect": "Allow",
            "Action": [
                "ecr:DescribeRepositories",
                "ecr:GetAuthorizationToken",
                "ecr-public:DescribeRepositories",
                "ecr-public:GetAuthorizationToken"
            ],
        "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:PutImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchCheckLayerAvailability"
```

```
            ],
            "Resource": "PrivateECR_Resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr-public:GetAuthorizationToken",
                "ecr-public:DescribeRepositories",
                "ecr-public:InitiateLayerUpload",
                "ecr-public:UploadLayerPart",
                "ecr-public:CompleteLayerUpload",
                "ecr-public:PutImage",
                "ecr-public:BatchCheckLayerAvailability",
                "sts:GetServiceBearerToken"
            ],
            "Resource": "PublicECR_Resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sts:GetServiceBearerToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Además, si aún no lo has agregado para la `Commands` acción, agrega los siguientes permisos a tu rol de servicio para ver CloudWatch los registros.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Reduzca los permisos al nivel de recursos de la canalización a través de los permisos basados en recursos de la declaración de la política del rol de servicio.

Para obtener más información sobre esta acción, consulte Referencia de la acción de compilación `ECRBuildAndPublish`.

## Declaración de acciones

YAML

```
name: ECRBuild
actionTypeId:
  category: Build
  owner: AWS
  provider: ECRBuildAndPublish
  version: '1'
runOrder: 1
configuration:
  ECRRepositoryName: actions/my-imagerepo
outputArtifacts: []
inputArtifacts:
- name: SourceArtifact
region: us-east-1
namespace: BuildVariables
```

JSON

```
{
    "name": "ECRBuild",
    "actionTypeId": {
        "category": "Build",
        "owner": "AWS",
        "provider": "ECRBuildAndPublish",
        "version": "1"
    },
    "runOrder": 1,
    "configuration": {
        "ECRRepositoryName": "actions/my-imagerepo"
```

```
    },
    "outputArtifacts": [],
    "inputArtifacts": [
        {
            "name": "SourceArtifact"
        }
    ],
    "region": "us-east-1",
    "namespace": "BuildVariables"
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Tutorial: Cree y envíe una imagen de Docker a Amazon ECR con CodePipeline (tipo V2). Este tutorial proporciona un ejemplo de Dockerfile e instrucciones para crear una canalización que inserta la imagen en ECR con un cambio en el repositorio de origen y, a continuación, la implementa en Amazon ECS.

# Referencia de acciones de implementación de Amazon Elastic Container Service y CodeDeploy azul-verde

Puede configurar una canalización para implementar aplicaciones de contenedor mediante una blue/green implementación. AWS CodePipeline En una blue/green implementación, puede lanzar una nueva versión de la aplicación junto con la versión anterior y probar la nueva versión antes de redirigir el tráfico hacia ella. También puede monitorizar el proceso de implementación y revertirlo rápidamente si hay algún problema.

La canalización completa detecta los cambios en las imágenes o en el archivo de definición de tareas y los utiliza CodeDeploy para enrutar e implementar el tráfico en un clúster y un balanceador de cargas de Amazon ECS. CodeDeploy crea un nuevo receptor en su balanceador de cargas que puede dirigir su nueva tarea a través de un puerto especial. También puede configurar la canalización para que utilice una ubicación de origen, como un CodeCommit repositorio, donde se almacena la definición de la tarea de Amazon ECS.

Antes de crear la canalización, debe haber creado los recursos de Amazon ECS, los CodeDeploy recursos y el balanceador de cargas y el grupo objetivo. Ya debe haber etiquetado y almacenado la

imagen en su repositorio de imágenes y haber subido la definición de la tarea y el AppSpec archivo a su repositorio de archivos.

> **Note**
>
> En este tema se describe la acción de CodeDeploy blue/green implementación de Amazon ECS CodePipeline. Para obtener información de referencia sobre las acciones de implementación estándar de Amazon ECS en CodePipeline, consulteReferencia de acción de implementación de Amazon Elastic Container Service.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Permisos del rol de servicio: acción de CodeDeployToECS
- Declaración de acciones
- Véase también


## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `CodeDeployToECS`
- Versión: `1`

## Parámetros de configuración

ApplicationName

Obligatorio: sí

El nombre de la aplicación en CodeDeploy. Antes de crear tu canalización, debes haber creado la aplicación en CodeDeploy.

DeploymentGroupName

Obligatorio: sí

El grupo de implementación especificado para los conjuntos de tareas de Amazon ECS que creó para su CodeDeploy aplicación. Antes de crear la canalización, debe haber creado el grupo de implementación en CodeDeploy.

TaskDefinitionTemplateArtifact

Obligatorio: sí

El nombre del artefacto de entrada que proporciona el archivo de definición de tareas a la acción de implementación. Este es generalmente el nombre del artefacto de salida de la acción de origen. Cuando utiliza la consola, el nombre por defecto del artefacto de salida de la acción de origen es `SourceArtifact`.

AppSpecTemplateArtifact

Obligatorio: sí

El nombre del artefacto de entrada que proporciona el AppSpec archivo para la acción de implementación. Este valor se actualiza cuando se ejecuta la canalización. Este es generalmente el nombre del artefacto de salida de la acción de origen. Cuando utiliza la consola, el nombre por defecto del artefacto de salida de la acción de origen es `SourceArtifact`. Si está `TaskDefinition` en AppSpec un archivo, puede conservar el texto del `<TASK_DEFINITION>` marcador de posición tal y como se muestra aquí.

AppSpecTemplatePath

Obligatorio: no

El nombre del archivo almacenado en la ubicación del AppSpec archivo fuente de la canalización, como el CodeCommit repositorio de la canalización. El nombre del archivo predeterminado es `appspec.yaml`. Si tu AppSpec archivo tiene el mismo nombre y está almacenado en el nivel raíz de tu repositorio de archivos, no necesitas proporcionar el nombre del archivo. Si la ruta no es la predeterminada, introdúzcala con el nombre del archivo.

TaskDefinitionTemplatePath

Obligatorio: no

El nombre de archivo de la definición de tarea almacenada en la ubicación de origen del archivo de canalización, como el CodeCommit repositorio de tu canalización. El nombre del archivo

predeterminado es `taskdef.json`. Si el archivo de definición de tarea tiene el mismo nombre y está almacenado en el nivel raíz del repositorio de archivos, no es necesario que proporcione el nombre del archivo. Si la ruta no es la predeterminada, introdúzcala con el nombre del archivo.

Imagen <Number>ArtifactName

Obligatorio: no

El nombre del artefacto de entrada que proporciona la imagen a la acción de implementación. Por lo general, se trata del artefacto de salida del repositorio de imágenes, como el resultado de la acción de origen de Amazon ECR.

Los valores disponibles para `<Number>` son del 1 al 4.

Imagen <Number>ContainerName

Obligatorio: no

El nombre de la imagen disponible en el repositorio de imágenes, como el repositorio de origen de Amazon ECR.

Los valores disponibles para `<Number>` son del 1 al 4.

## Artefactos de entrada

- Número de artefactos: `1 to 5`
- Descripción: La `CodeDeployToECS` acción busca primero el archivo de definición de tareas y el AppSpec archivo en el repositorio de archivos de origen, luego busca la imagen en el repositorio de imágenes, genera dinámicamente una nueva revisión de la definición de tareas y, por último, ejecuta los AppSpec comandos para implementar el conjunto de tareas y el contenedor en el clúster.

  La acción `CodeDeployToECS` busca un archivo `imageDetail.json` que asigne el URI de la imagen a la imagen. Cuando se confirma un cambio en el repositorio de imágenes de Amazon ECR, la acción de origen de ECR de la canalización crea un archivo `imageDetail.json` para dicha confirmación. También puede añadir manualmente un archivo `imageDetail.json` para una canalización en la que la acción no esté automatizada. Para obtener más información sobre el archivo `imageDetail.json`, consulte Detail.json archivo de imagen para las acciones de blue/green implementación de Amazon ECS.

La acción `CodeDeployToECS` genera de forma dinámica una nueva revisión de la definición de la tarea. En esta fase, esta acción sustituye los marcadores de posición del archivo de definición de tareas por el URI de la imagen recuperada de Detail.json los archivos de imagen. Por ejemplo, si establece IMAGE1_NAME como Image1ContainerName parámetro, debe especificar el marcador de posición <IMAGE1_NAME>como el valor del campo de imagen en su archivo de definición de tareas. En este caso, la CodeDeployToECS acción reemplaza el marcador de posición por el <IMAGE1_NAME>URI real de la imagen recuperada de la imagen del artefacto que Detail.json especifiques. Image1ArtifactName

En el caso de las actualizaciones de definiciones de tareas, el CodeDeploy `AppSpec.yaml` archivo contiene la `TaskDefinition` propiedad.

```
TaskDefinition: <TASK_DEFINITION>
```

La acción `CodeDeployToECS` actualizará esta propiedad mediante la acción una vez creada la nueva definición de tarea.

Para el valor del campo `TaskDefinition`, el texto del marcador de posición debe ser <TASK_DEFINITION>. La acción `CodeDeployToECS` reemplaza este marcador de posición por el ARN real de la definición de tarea generada dinámicamente.

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Permisos del rol de servicio: acción de **`CodeDeployToECS`**

Para la `CodeDeployToECS` acción (blue/green despliegues), los siguientes son los permisos mínimos necesarios para crear canalizaciones con una acción de blue/green implementación CodeDeploy hacia Amazon ECS.

JSON

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

```
    "Statement": [
        {
            "Sid": "AllowCodeDeployDeploymentActions",
            "Action": [
                "codedeploy:CreateDeployment",
                "codedeploy:GetDeployment"
            ],
            "Resource": [
                "arn:aws:codedeploy:*:111122223333:deploymentgroup:
[[ApplicationName]]/*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "AllowCodeDeployApplicationActions",
            "Action": [
                "codedeploy:GetApplication",
                "codedeploy:GetApplicationRevision",
                "codedeploy:RegisterApplicationRevision"
            ],
            "Resource": [
                "arn:aws:codedeploy:*:111122223333:application:
[[ApplicationName]]",
                "arn:aws:codedeploy:*:111122223333:application:
[[ApplicationName]]/*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "AllowCodeDeployDeploymentConfigAccess",
            "Action": [
                "codedeploy:GetDeploymentConfig"
            ],
            "Resource": [
                "arn:aws:codedeploy:*:111122223333:deploymentconfig:*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "AllowECSRegisterTaskDefinition",
            "Action": [
                "ecs:RegisterTaskDefinition"
            ],
            "Resource": [
```

```
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "AllowPassRoleToECS",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::111122223333:role/[[PassRoles]]"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ecs.amazonaws.com",
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Puede optar por utilizar la autorización de etiquetado en Amazon ECS. Al suscribirse, debe otorgar los siguientes permisos: `ecs:TagResource`. Para obtener más información sobre cómo suscribirse y determinar si el permiso es obligatorio y si se aplica la autorización de etiquetas, consulte el Plazos de la autorización de etiquetado en la Guía para desarrolladores de Amazon Elastic Container Service.

También debe añadir los permisos `iam:PassRole` para utilizar los rones de IAM en las tareas. Para obtener más información, consulte Rol de ejecución de tareas de Amazon ECS y Roles de IAM para tareas.

También puede añadir `ecs-tasks.amazonaws.com` a la lista de servicios con la condición `iam:PassedToService`, como se muestra en el ejemplo anterior.

## Declaración de acciones

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"
        }
    ]
}
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Tutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR— En este tutorial se explica cómo crear los recursos de Amazon ECS CodeDeploy y los recursos de Amazon ECS que necesita para una blue/green implementación. El tutorial muestra cómo insertar una imagen de Docker en Amazon ECR y cómo crear una definición de tarea de Amazon ECS que incluya el nombre de la imagen de Docker, el nombre del contenedor, el nombre del servicio de Amazon ECS y la configuración del equilibrador de carga. A continuación, el tutorial le explica cómo crear el AppSpec archivo y la canalización para su implementación.

> **Note**
>
> En este tema y en este tutorial se describe la CodeDeploy/ECS blue/green acción para CodePipeline. Para obtener información sobre las acciones estándar de ECS en CodePipeline, consulte Tutorial: Continuous Deployment with CodePipeline.

- AWS CodeDeploy Guía del usuario: para obtener información sobre cómo usar el balanceador de cargas, el detector de producción, los grupos objetivo y la aplicación de Amazon ECS en una blue/green implementación, consulte el tutorial: Implementación de un servicio de Amazon ECS. Esta información de referencia de la guía del AWS CodeDeploy usuario proporciona una descripción general de las blue/green implementaciones con Amazon ECS y. AWS CodeDeploy
- Guía para desarrolladores de Amazon Elastic Container Service: para obtener información sobre cómo trabajar con imágenes y contenedores de Docker, servicios y clústeres de ECS y conjuntos de tareas de ECS, consulte ¿Qué es Amazon ECS?

# Referencia de acción de implementación de Amazon Elastic Container Service

Puede utilizar una acción de Amazon ECS para implementar un conjunto de tareas y un servicio de Amazon ECS. Un servicio de Amazon ECS es una aplicación de contenedor que se implementa en un clúster de Amazon ECS. Un clúster de Amazon ECS es una colección de instancias que alojan su aplicación de contenedores en la nube. La implementación requiere una definición de tarea que cree en Amazon ECS y un archivo de definiciones de imágenes que se CodePipeline utilice para implementar la imagen.

> **Important**
>
> La acción de implementación estándar de Amazon ECS para CodePipeline crea su propia revisión de la definición de la tarea en función de la revisión utilizada por el servicio Amazon ECS. Si crea nuevas revisiones para la definición de la tarea sin actualizar el servicio Amazon ECS, la acción de implementación ignorará esas revisiones.

Antes de crear tu canalización, debes haber creado los recursos de Amazon ECS, haber etiquetado y almacenado la imagen en tu repositorio de imágenes y haber subido el archivo buildspec a tu repositorio de archivos.

> **Note**
>
> En este tema de referencia se describe la acción de implementación estándar de Amazon ECS para. CodePipeline Para obtener información de referencia sobre Amazon ECS para

CodeDeploy blue/green implementar acciones CodePipeline, consulteReferencia de acciones de implementación de Amazon Elastic Container Service y CodeDeploy azul-verde.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Permisos del rol de servicio: acción estándar de Amazon ECS
- Declaración de acciones
- Véase también


## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `ECS`
- Versión: `1`

## Parámetros de configuración

ClusterName

Obligatorio: sí

El clúster de Amazon ECS en Amazon ECS.

ServiceName

Obligatorio: sí

El servicio de Amazon ECS que creó en Amazon ECS.

FileName

Obligatorio: no

El nombre de su archivo de definiciones de imagen, el archivo JSON que describe el nombre del contenedor de su servicio y la imagen y la etiqueta. Este archivo se utiliza para las implementaciones estándar de ECS. Para obtener más información, consulte Artefactos de entrada y Archivo imagedefinitions.json para las acciones de implementación estándar de.

DeploymentTimeout

Obligatorio: no

El tiempo de espera de la acción de implementación de Amazon ECS en minutos. El tiempo de espera se puede configurar hasta el tiempo de espera predeterminado máximo para esta acción. Por ejemplo:

```
"DeploymentTimeout": "15"
```

## Artefactos de entrada

- Número de artefactos: 1
- Descripción: la acción busca un archivo `imagedefinitions.json` en el repositorio de archivos de origen para la canalización. Un documento de definiciones de imágenes es un archivo JSON que describe el nombre del contenedor de Amazon ECS, así como la imagen y la etiqueta. CodePipeline usa el archivo para recuperar la imagen de su repositorio de imágenes, como Amazon ECR. Puede añadir manualmente un archivo `imagedefinitions.json` para una canalización en la que la acción no esté automatizada. Para obtener más información sobre el archivo `imagedefinitions.json`, consulte Archivo imagedefinitions.json para las acciones de implementación estándar de.

  La acción requiere una imagen existente que ya se haya subido a su repositorio de imágenes. Como la asignación de imágenes la proporciona el archivo `imagedefinitions.json`, la acción no requiere que el origen de Amazon ECR se incluya como acción de origen en el proceso.

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

# Permisos del rol de servicio: acción estándar de Amazon ECS

A continuación se indican los permisos mínimos necesarios en Amazon ECS para crear canalizaciones con una acción de implementación de Amazon ECS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "TaskDefinitionPermissions",
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeTaskDefinition",
                "ecs:RegisterTaskDefinition"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "ECSServicePermissions",
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeServices",
                "ecs:UpdateService"
            ],
            "Resource": [
                "arn:aws:ecs:*:111122223333:service/[[clusters]]/*"
            ]
        },
        {
            "Sid": "ECSTagResource",
            "Effect": "Allow",
            "Action": [
                "ecs:TagResource"
            ],
            "Resource": [
                "arn:aws:ecs:*:111122223333:task-definition/
[[taskDefinitions]]:*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": [
                        "RegisterTaskDefinition"
                    ]
                }
            }
        },
        {
            "Sid": "IamPassRolePermissions",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::111122223333:role/[[passRoles]]"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ecs.amazonaws.com",
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Puede optar por utilizar la autorización de etiquetado en Amazon ECS. Al suscribirse, debe otorgar los siguientes permisos: `ecs:TagResource`. Para obtener más información sobre cómo suscribirse y determinar si el permiso es obligatorio y si se aplica la autorización de etiquetas, consulte el Plazos de la autorización de etiquetado en la Guía para desarrolladores de Amazon Elastic Container Service.

También debe añadir los permisos `iam:PassRole` para utilizar los roles de IAM en las tareas. Para obtener más información, consulte Rol de ejecución de tareas de Amazon ECS y Roles de IAM para tareas. Utilice la siguiente política:

# Declaración de acciones

## 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"
        }
    ]
```

```
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Consulte Tutorial: Cree y envíe una imagen de Docker a Amazon ECR con CodePipeline (tipo V2) un tutorial que muestra cómo utilizar la ECRBuildandPublish acción para insertar una imagen y, a continuación, utilizar la acción estándar de ECS para implementarla en Amazon ECS.
- Tutorial: implementación continua con CodePipeline: este tutorial le muestra cómo crear un Dockerfile para almacenarlo en un repositorio de archivos fuente, por ejemplo. CodeCommit A continuación, el tutorial muestra cómo incorporar un archivo CodeBuild buildspec que compila y envía la imagen de Docker a Amazon ECR y crea el archivo imagedefinitions.json. Por último, se crea una definición de tareas y servicios de Amazon ECS y, a continuación, se crea la canalización con una acción de implementación de Amazon ECS.

  > **Note**
  >
  > En este tema y en este tutorial se describe la acción de implementación estándar de Amazon ECS para. CodePipeline Para obtener información sobre Amazon ECS para CodeDeploy blue/green implementar acciones CodePipeline, consulteTutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR.

- Guía para desarrolladores de Amazon Elastic Container Service: para obtener información sobre cómo trabajar con imágenes y contenedores de Docker, los servicios y clústeres de Amazon ECS y los conjuntos de tareas de Amazon ECS, consulte ¿Qué es Amazon ECS?

# Referencia de acción de implementación de Amazon Elastic Kubernetes Service **EKS**

Puede usar la acción `EKSDeploy` para implementar un servicio de Amazon EKS. La implementación requiere un archivo de manifiesto de Kubernetes que CodePipeline se utilice para implementar la imagen.

Antes de crear la canalización, debe haber creado los recursos de Amazon ECS y haber almacenado la imagen en su repositorio de imágenes. Opcionalmente, puede proporcionar información de VPC para su clúster.

> **Important**
>
> Esta acción usa CodeBuild procesamiento CodePipeline administrado para ejecutar comandos en un entorno de compilación. Si ejecuta la acción de Comandos, se le cobrarán cargos por separado en AWS CodeBuild.

> **Note**
>
> La acción de implementación de EKS solo está disponible para canalizaciones de tipo V2.

La acción de EKS admite clústeres de EKS públicos y privados. Los clústeres privados son el tipo recomendado por EKS, pero se admiten ambos.

La acción de EKS es compatible con las acciones en varias cuentas. Para agregar una acción de EKS en varias cuentas, agregue `actionRoleArn` en su cuenta de destino en la declaración de la acción.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de entorno
- Variables de salida
- Permisos para las políticas de roles de servicio
- Declaración de acciones
- Véase también

## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `EKS`
- Versión: `1`

## Parámetros de configuración

ClusterName

Obligatorio: sí

El clúster de Amazon EKS en Amazon EKS.

Opciones en Helm

Las siguientes opciones están disponibles cuando Helm es la herramienta de implementación seleccionada.

HelmReleaseName

Obligatorio: sí (solo para el tipo Helm)

El nombre de la versión de la implementación.

HelmChartLocation

Obligatorio: sí (solo para el tipo Helm)

La ubicación del gráfico para la implementación.

HelmValuesFiles

Obligatorio: no (opcional solo para el tipo Helm)

Para anular los archivos de valores de Helm, introduzca los archivos de valores de Helm separados por comas en la ubicación del gráfico de Helm.

Opciones en Kubectl

Las siguientes opciones están disponibles cuando Kubectl es la herramienta de implementación seleccionada.

ManifestFiles

Obligatorio: sí (solo para el tipo Kubectl)

El nombre del archivo de manifiesto, el archivo de texto que describe el nombre del contenedor del servicio y la imagen y la etiqueta. Este archivo se usa para parametrizar el URI de la imagen y otra información. Puede usar una variable de entorno para este propósito.

Este archivo se almacena en el repositorio de origen de la canalización.

Namespace

Obligatorio: no

El espacio de nombres de Kubernetes que se va a utilizar en los comandos o. `kubectl` `helm`

Subredes

Obligatorio: no

Las subredes de VPC para el clúster. Forman parte de la misma VPC que está conectada al clúster. También puede proporcionar subredes que aún no estén conectadas al clúster y especificarlas aquí.

SecurityGroupIds

Obligatorio: no

Los grupos de seguridad de la VPC del clúster. Forman parte de la misma VPC que está conectada al clúster. También puede proporcionar grupos de seguridad que aún no estén conectados al clúster y especificarlos aquí.

## Artefactos de entrada

- Número de artefactos: 1
- Descripción: la acción busca el archivo de manifiesto de Kubernetes o gráfico de Helm en el repositorio de archivos de origen para la canalización. Si quieres usar los gráficos de Helm en formato.tgz almacenados en un bucket de S3, puedes hacerlo configurando el S3 como tu acción de origen. Bucket/Key Por ejemplo, la clave de objeto proporcionada sería `my-chart-0.1.0.tgz`.

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Variables de entorno

Se usa para reemplazar variables como los repositorios de imágenes o las etiquetas de imágenes en los archivos de manifiesto o en los archivos de valores de los gráficos de Helm.

Key

La clave de un par de variables de entorno de clave-valor, como `$IMAGE_TAG`.

Valor

El valor del par clave-valor, como `v1.0`. El valor se puede parametrizar con variables de salida de acciones o variables de canalización. Por ejemplo, Pipeline puede tener una ECRBuildAndPublish acción con la que crear una imagen ECR`${codepipeline.PipelineExecutionId}`, y la acción EKS puede usar esta imagen `${codepipeline.PipelineExecutionId}` como valor de la variable de entorno.

## Variables de salida

EKSClusterName

El clúster de Amazon EKS en Amazon EKS.

## Permisos para las políticas de roles de servicio

Para ejecutar esta acción, los siguientes permisos deben estar disponibles en la política de rol de servicio de la canalización.

- Acciones de EC2:  cuando se CodePipeline ejecuta la acción, se requieren permisos de instancia de EC2. Tenga en cuenta que no es lo mismo que el rol de instancia de EC2 que se requiere al crear el clúster de EKS.

  Si utiliza un rol de servicio existente, para utilizar esta acción tendrá que agregar los siguientes permisos para el rol de servicio.

- ec2: CreateNetworkInterface
- ec2: DescribeDhcpOptions
- ec2: DescribeNetworkInterfaces
- ec2: DeleteNetworkInterface
- ec2: DescribeSubnets
- ec2: DescribeSecurityGroups
- ec2: DescribeVpcs

- Acciones de EKS:  cuando se CodePipeline ejecuta la acción, se requieren los permisos del clúster de EKS. Tenga en cuenta que no es lo mismo que el rol de clúster de EKS de IAM que se requiere al crear el clúster de EKS.

  Si utiliza un rol de servicio existente, para utilizar esta acción tendrá que agregar el siguiente permiso para el rol de servicio.

  - Eks: DescribeCluster
- Registrar las acciones del flujo:  cuando CodePipeline se ejecuta la acción, CodePipeline crea un grupo de registros con el nombre de la canalización de la siguiente manera. Esto permite reducir los permisos para registrar los recursos mediante el nombre de la canalización.

  ```
  /aws/codepipeline/MyPipelineName
  ```

  Si utiliza un rol de servicio existente, para utilizar esta acción tendrá que agregar los siguientes permisos para el rol de servicio.

  - registros: CreateLogGroup
  - registros: CreateLogStream
  - registros: PutLogEvents

En la declaración de la política de rol de servicio, limite los permisos al nivel de recurso, como se muestra en el siguiente ejemplo.

JSON

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

```
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "arn:aws:eks:*:111122223333:cluster/YOUR_CLUSTER_NAME"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcs",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:111122223333:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME",
                "arn:aws:logs:*:111122223333:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
            ]
        }
    ]
}
```

Para ver los registros en la consola mediante la página del cuadro de diálogo de detalles de la acción, se debe agregar el permiso para ver los registros al rol de la consola. Para obtener más

información, consulte el ejemplo de política de permisos para consolas en Permisos necesarios para ver los registros de procesamiento en la consola.

### Agregación del rol de servicio como entrada de acceso al clúster

Una vez que los permisos estén disponibles en la política de funciones de servicio de tu canalización, puedes configurar los permisos del clúster añadiendo la función de CodePipeline servicio como entrada de acceso para tu clúster.

También puede usar un rol de acción que tenga los permisos actualizados. Para obtener más información, consulte el tutorial de ejemplo en Paso 4: Cree una entrada de acceso para la función CodePipeline de servicio.

## Declaración de acciones

YAML

```
Name: DeployEKS
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: EKS
  Version: '1'
RunOrder: 2
Configuration:
  ClusterName: my-eks-cluster
  ManifestFiles: ManifestFile.json
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
```

JSON

```
{
    "Name": "DeployECS",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "EKS",
        "Version": "1"
    },
    "RunOrder": 2,
```

```
    "Configuration": {
        "ClusterName": "my-eks-cluster",
        "ManifestFiles": "ManifestFile.json"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ]
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Consulte en Tutorial: Implemente en Amazon EKS con CodePipeline un tutorial que muestra cómo crear un clúster de EKS y un archivo de manifiesto de Kubernetes para agregar la acción a la canalización.

# AWS Lambda Referencia de la acción de implementación de

Utiliza una acción de AWS Lambda implementación para administrar la implementación del código de la aplicación para la implementación sin servidor. Puede implementar una función y utilizar estrategias de implementación para la implementación del tráfico de la siguiente manera:

- Implementaciones lineales y canarias para el cambio del tráfico
- Implementaciones Todo a la vez

> **Note**
>
> Esta acción solo se admite para canalizaciones de tipo V2.

Temas

- Tipo de acción
- Parámetros de configuración

- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Permisos de la política de rol de servicio para la acción de implementación de Lambda
- Declaración de acciones
- Véase también

## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `Lambda`
- Versión: `1`

## Parámetros de configuración

FunctionName

Obligatorio: sí

En nombre de la función que creó en Lambda, como `MyLambdaFunction`.

Ya debe haber creado una versión.

FunctionAlias

Obligatorio: no

El alias de la función que creó en Lambda y es la función en la que se va a implementar, por ejemplo, `live`. El alias debe existir y tener una versión posterior cuando comience la ejecución de la acción. (Será la versión de destino de la reversión).

Si no se proporciona, la acción implementa el artefacto de origen en `$LATEST` y crea una nueva versión. En este caso de uso, las opciones de estrategia de implementación y versión de destino no están disponibles.

PublishedTargetVersion

Obligatorio: no

La versión de la función Lambda deseada en la que se va a FunctionAlias implementar. Pueden ser variables de canalización o de nivel de acción, como `#{variables.lambdaTargetVersion}`. La versión debe publicarse cuando comience la ejecución de la acción.

Obligatorio si no se proporciona ningún artefacto de entrada.

DeployStrategy

Obligatorio: no (el valor predeterminado es `AllAtOnce`)

Determina la velocidad que adopta la acción de implementación de Lambda para transferir el tráfico de la versión original de la función de Lambda a la nueva versión. FunctionAlias Las estrategias de implementación disponibles son canaria o lineal. Formatos aceptados:

- `AllAtOnce` -

  Desvía todo el tráfico a las funciones Lambda actualizadas a la vez.

  Si no se especifica, el valor predeterminado es `AllAtOnce`.
- `Canary10Percent5Minutes`: cambia el 10 por ciento del tráfico en el primer incremento. El 90 por ciento restante se implementa cinco minutos más tarde.

  Se pueden cambiar los valores tanto del porcentaje como de los minutos.
- `Linear10PercentEvery1Minute`: cambia el 10 por ciento del tráfico cada minuto hasta que se cambia todo el tráfico.

  Se pueden cambiar los valores tanto del porcentaje como de los minutos.

Se deben tener en cuenta las siguientes consideraciones para este campo:

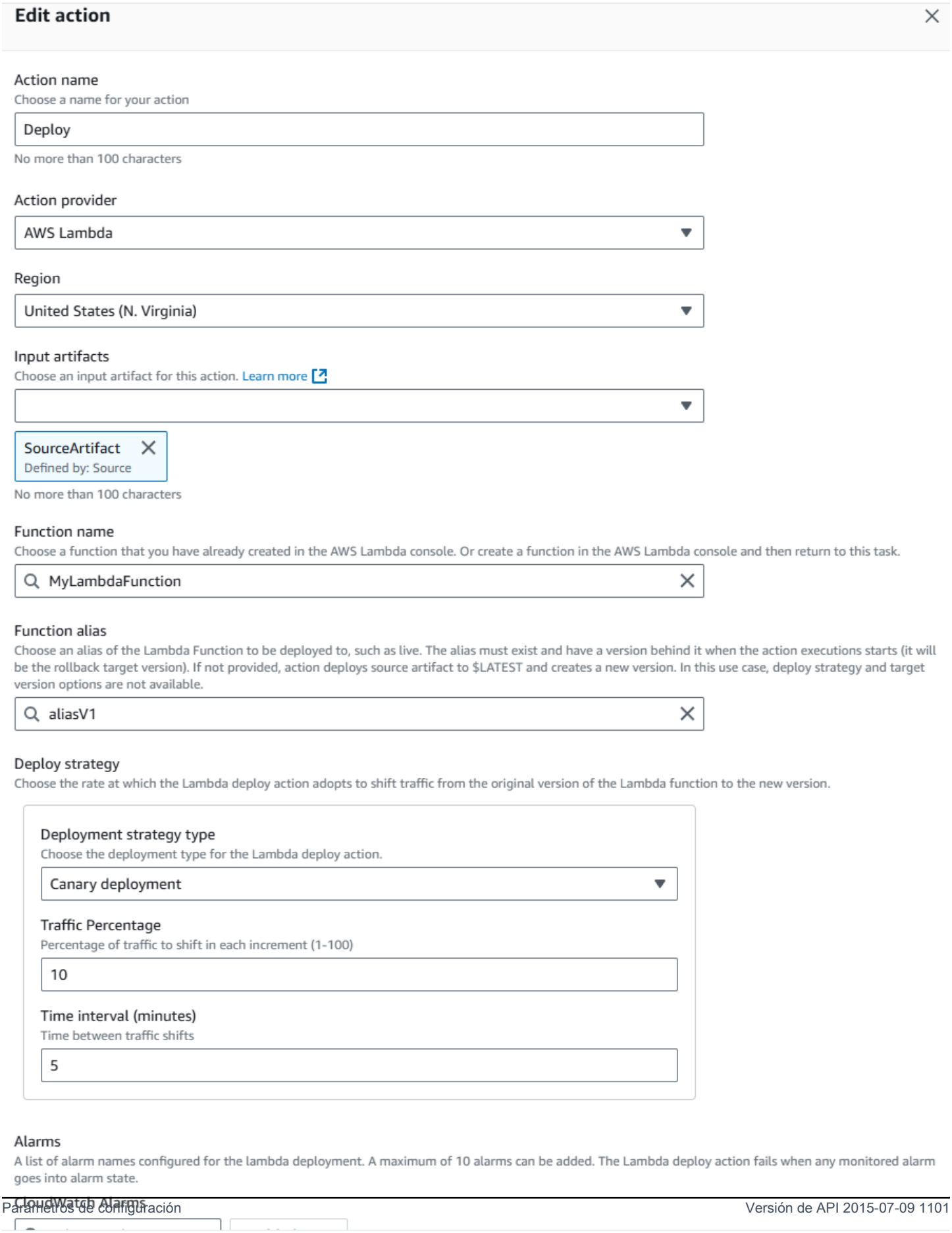
- El tiempo de espera total máximo es de dos días.
- Solo está disponible cuando FunctionAlias se proporciona.

Alarmas

Obligatorio: no

Una lista separada por comas con los nombres de las alarmas configuradas para la implementación de Lambda. Se puede agregar un máximo de 10 alarmas. La acción falla cuando las alarmas monitorizadas pasan al estado de ALARMA.

En la siguiente imagen, se muestra un ejemplo de la página Editar de la acción.


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

## Artefactos de entrada

- Número de artefactos: 1
- Descripción: los archivos proporcionados (en caso de haberlos) para respaldar las acciones del script durante la implementación.

## Artefactos de salida

- Número de artefactos: 0
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Variables de salida

Cuando se configura, esta acción produce variables a las que se puede hacer referencia mediante la configuración de acción de una acción descendente en la canalización. Esta acción produce variables que se pueden ver como variables de salida, incluso si la acción no tiene un espacio de nombres. Configure una acción con un espacio de nombres para que esas variables estén disponibles para la configuración de las acciones posteriores.

Para obtener más información, consulte Referencia de variables.

FunctionVersion

La nueva versión de la función de Lambda que se implementó.

## Permisos de la política de rol de servicio para la acción de implementación de Lambda

Cuando CodePipeline se ejecuta la acción, el rol de CodePipeline servicio requiere los siguientes permisos, con un alcance adecuado para el acceso con el mínimo privilegio.

JSON

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

```
            "Sid": "StatementForLambda",
            "Effect": "Allow",
            "Action": [
                "lambda:GetAlias",
                "lambda:GetFunctionConfiguration",
                "lambda:GetProvisionedConcurrencyConfig",
                "lambda:PublishVersion",
                "lambda:UpdateAlias",
                "lambda:UpdateFunctionCode"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:111122223333:function:
{{FunctionName}}",
                "arn:aws:lambda:us-east-1:111122223333:function:
{{FunctionName}}:*"
            ]
        },
        {
            "Sid": "StatementForCloudWatch",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": [
                "arn:aws:cloudwatch:us-east-1:111122223333:alarm:{{AlarmNames}}"
            ]
        },
        {
            "Sid": "StatementForLogs1",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:/us-east-1/
codepipeline/{{pipelineName}}",
                "arn:aws:logs:us-east-1:111122223333:log-group:/us-east-1/
codepipeline/{{pipelineName}}:*"
            ]
        },
        {
            "Sid": "StatementForLogs2",
            "Effect": "Allow",
            "Action": [
```

```
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:/us-east-1/
codepipeline/{{pipelineName}}:log-stream:*"
            ]
        }
    ]
}
```

## Declaración de acciones

### YAML

```
name: Deploy
actionTypeId:
  category: Deploy
  owner: AWS
  provider: Lambda
  version: '1'
runOrder: 1
configuration:
  DeployStrategy: Canary10Percent5Minutes
  FunctionAlias: aliasV1
  FunctionName: MyLambdaFunction
outputArtifacts: []
inputArtifacts:
- name: SourceArtifact
region: us-east-1
namespace: DeployVariables
```

### JSON

```
{
    "name": "Deploy",
    "actionTypeId": {
        "category": "Deploy",
        "owner": "AWS",
        "provider": "Lambda",
        "version": "1"
    },
```

```
    "runOrder": 1,
    "configuration": {
        "DeployStrategy": "Canary10Percent5Minutes",
        "FunctionAlias": "aliasV1",
        "FunctionName": "MyLambdaFunction"
    },
    "outputArtifacts": [],
    "inputArtifacts": [
        {
            "name": "SourceArtifact"
        }
    ],
    "region": "us-east-1",
    "namespace": "DeployVariables"
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Tutorial: Implementaciones de funciones Lambda con CodePipeline. En este tutorial, se explica cómo crear una función de Lambda de ejemplo en la que se crean un alias y una versión, se agrega la función de Lambda comprimida a la ubicación de origen y se ejecuta la acción de Lambda en la canalización.

# Referencia de acción de implementación de Amazon S3

Utiliza una acción de implementación de Amazon S3 para implementar archivos en un bucket de Amazon S3 para hospedar o archivar sitios web estáticos. Puede especificar si desea extraer los archivos de implementación antes de subirlos a su bucket.

> **Note**
>
> En este tema de referencia se describe la acción de implementación de Amazon S3 CodePipeline cuando la plataforma de implementación es un bucket de Amazon S3 configurado para el alojamiento. Para obtener información de referencia sobre la acción de origen de Amazon S3 en CodePipeline, consulteReferencia sobre la acción de origen de Amazon S3.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Permisos del rol de servicio: acción de implementación de S3
- Ejemplo de configuración de una acción
- Véase también


## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `S3`
- Versión: `1`

## Parámetros de configuración

BucketName

Obligatorio: sí

El nombre del bucket de Amazon S3 en el que se implementan los archivos.

Extract

Obligatorio: sí

Si es verdadero, especifica que los archivos se deben extraer antes de cargarlos. De lo contrario, los archivos de la aplicación permanecen comprimidos para su carga, como en el caso de un sitio web estático alojado. Si es falso, entonces `ObjectKey` es obligatoria.

ObjectKey

Condicional. Obligatorio si `Extract` = false

El nombre de la clave del objeto de Amazon S3 que identifica de forma exclusiva el objeto en el bucket de S3.

KMSEncryptionKeyARN

Obligatorio: no

El ARN de la clave de AWS KMS cifrado del bucket de host. El parámetro `KMSEncryptionKeyARN` cifra los artefactos cargados con la AWS KMS key proporcionada. Si se trata de una clave de KMS, puede utilizar el ID de la clave, el ARN de la clave o el ARN del alias.

> **Note**
>
> Los alias se reconocen únicamente en la cuenta que ha creado la clave de KMS. Para las acciones entre cuentas, solo puedes usar el ID de la clave o el ARN de la clave para identificar la clave. Cross-account las acciones implican usar el rol de la otra cuenta (AccountB), por lo que al especificar el ID de la clave se utilizará la clave de la otra cuenta (AccountB).

> **Important**
>
> CodePipeline solo admite claves KMS simétricas. No utilice una clave de KMS asimétrica para cifrar los datos en el bucket de S3.

CannedACL

Obligatorio: no

El parámetro `CannedACL` aplica la ACL predefinida especificada a objetos implementados en Amazon S3. Esto sobreescribe cualquier ACL existente aplicado al objeto.

CacheControl

Obligatorio: no

El `CacheControl` parámetro controla el comportamiento de almacenamiento en caché de requests/responses los objetos del bucket. Para una lista de valores válidos, consulte el `Cache-Control`campo del encabezado para las operaciones HTTP Para introducir varios valores en `CacheControl`, utilice una coma entre cada valor. Puede añadir un espacio después de cada coma (opcional), tal y como se muestra en este ejemplo para la CLI:

```
"CacheControl": "public, max-age=0, no-transform"
```

## Artefactos de entrada

- Número de artefactos: 1
- Descripción:  Los archivos para la implementación o el archivado se obtienen del repositorio de origen, se comprimen y se cargan en él. CodePipeline

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Permisos del rol de servicio: acción de implementación de S3

Para la compatibilidad con la acción de S3, agregue lo siguiente a su declaración de política:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl",
        "s3:GetBucketVersioning",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::[[s3DeployBuckets]]",
        "arn:aws:s3:::[[s3DeployBuckets]]/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
```

```
    }
  ]
}
```

Para admitir la acción de implementación de S3, si sus objetos de S3 tienen etiquetas, también debe agregar los siguientes permisos a su declaración de política:

```
"s3:GetObjectTagging",
"s3:GetObjectVersionTagging",
"s3:PutObjectTagging"
```

## Ejemplo de configuración de una acción

En el siguiente ejemplo se muestra la configuración de la acción.

### Ejemplo de configuración cuando **`Extract`** está establecido en falso

En el siguiente ejemplo se muestra la configuración de acción predeterminada cuando la acción se crea con el campo `Extract` establecido en `false`.

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: S3
      Version: '1'
    RunOrder: 1
    Configuration:
      BucketName: website-bucket
      Extract: 'false'
      ObjectKey: MyWebsite
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "S3",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "BucketName": "website-bucket",
                "Extract": "false",
                "ObjectKey": "MyWebsite"
                },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

## Ejemplo de configuración cuando **`Extract`** se establece en **`true`**

En el siguiente ejemplo se muestra la configuración de acción predeterminada cuando la acción se crea con el campo `Extract` establecido en `true`.

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
```

```
      Category: Deploy
      Owner: AWS
      Provider: S3
      Version: '1'
    RunOrder: 1
    Configuration:
      BucketName: website-bucket
      Extract: 'true'
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "S3",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "BucketName": "website-bucket",
                "Extract": "true"
                },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Tutorial: Crear una canalización que utilice Amazon S3 como proveedor de implementación: en este tutorial, se muestran dos ejemplos para crear una canalización con una acción de implementación de S3. Usted descarga los archivos de muestra, los carga en su CodeCommit repositorio, crea su bucket de S3 y lo configura para el alojamiento. A continuación, utiliza la CodePipeline consola para crear la canalización y especificar una configuración de implementación de Amazon S3.
- Referencia sobre la acción de origen de Amazon S3— Esta referencia de acciones proporciona información de referencia y ejemplos sobre las acciones de origen de Amazon S3 en CodePipeline.

# Referencia sobre la acción de origen de Amazon S3

Una acción de origen desencadena la canalización cuando se carga un nuevo objeto en el bucket y la clave de objeto configurados.

En este tema de referencia se describe la acción de origen de Amazon S3 CodePipeline cuando la ubicación de origen es un bucket de Amazon S3 configurado para el control de versiones. Para obtener información de referencia sobre la acción de implementación de Amazon S3 en CodePipeline, consulteReferencia de acción de implementación de Amazon S3.

Antes de conectar la canalización, cree un bucket fuente de Amazon S3, habilite el control de versiones y cargue los archivos fuente como un único archivo ZIP.

Puede crear un nuevo bucket en Amazon S3 o seleccionar un bucket de Amazon S3 existente para usarlo como ubicación de origen de los archivos de su aplicación. Para obtener más información sobre cómo habilitar el control de versiones, consulte el uso del control de versiones en la guía del usuario de Amazon S3.

Si Amazon S3 es el proveedor de código fuente de su canalización, puede comprimir el archivo o los archivos fuente en un único archivo .zip y subirlo a su bucket de código fuente. También puede cargar un único archivo descomprimido; sin embargo, las acciones posteriores que esperen un archivo .zip fallarán.

Si usa la consola para crear o editar la canalización, CodePipeline cree una EventBridge regla que inicie la canalización cuando se produzca un cambio en el bucket de origen de Amazon S3.

También puede crear una anulación de fuente utilizando la entrada de transformación de entrada EventBridge para usar la entrada `revisionValue` en el evento de canalización, donde `revisionValue` se deriva de la variable de evento de origen de su clave de objeto, confirmación o ID de imagen. Para obtener más información, consulta el paso opcional para la entrada de transformación de entrada que se incluye en los procedimientos que se indican a continuación. Conexión a acciones de origen de Amazon S3 con un origen habilitado para eventos

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Permisos del rol de servicio: acción de origen de S3
- Declaración de acciones
- Véase también


## Tipo de acción

- Categoría: `Source`
- Propietario: `AWS`
- Proveedor: `S3`
- Versión: `1`

## Parámetros de configuración

S3 Bucket

Obligatorio: sí

El nombre del bucket de Amazon S3 en el que se van a detectar los cambios de origen.

S3ObjectKey

Obligatorio: sí

El nombre de la clave de objeto de Amazon S3 en la que se van a detectar los cambios de origen.

AllowOverrideForS3ObjectKey

Obligatorio: no

`AllowOverrideForS3ObjectKey` controla si las anulaciones de origen de `StartPipelineExecution` pueden anular las `S3ObjectKey` ya configuradas en la acción de origen. Para obtener más información sobre las anulaciones de origen con la clave del objeto de S3, consulte Iniciar una canalización con una anulación de revisión de código fuente.

> **Important**
>
> Si lo omites`AllowOverrideForS3ObjectKey`, se establece de CodePipeline forma predeterminada la posibilidad de anular el S3 ObjectKey en la acción de origen configurando este parámetro en. `false`

Los valores válidos para este parámetro son:

- `true`: si se establece, la clave del objeto de S3 preconfigurada se puede anular mediante anulaciones de revisiones de origen durante la ejecución de una canalización.

  > **Note**
  >
  > Si quiere permitir a todos CodePipeline los usuarios la posibilidad de anular la clave de objeto de S3 preconfigurada al iniciar una nueva ejecución en canalización, debe configurarla en. `AllowOverrideForS3ObjectKey` `true`

- `false`:

  Si se establece, no CodePipeline permitirá anular la clave de objetos de S3 mediante anulaciones de revisión del código fuente. Este también es el valor predeterminado para este parámetro.

PollForSourceChanges

Obligatorio: no

`PollForSourceChanges`controla si CodePipeline sondea el bucket de origen de Amazon S3 para ver si hay cambios en el código fuente. En su lugar, le recomendamos que utilice CloudWatch Events and CloudTrail para detectar los cambios en el origen. Para obtener más

información sobre la configuración de CloudWatch eventos, consulte Migre las canalizaciones de sondeo con una fuente y un CloudTrail registro (CLI) de S3 oMigre las canalizaciones de sondeo con una fuente y un CloudTrail registro de S3 (CloudFormation plantilla).

> **Important**
>
> Si tiene intención de configurar CloudWatch eventos, debe configurarlos `PollForSourceChanges` para `false` evitar que se dupliquen las ejecuciones en canalización.

Los valores válidos para este parámetro son:

- `true`: Si se establece, CodePipeline sondea la ubicación de origen para ver si hay cambios en el origen.

  > **Note**
  >
  > Si lo omites`PollForSourceChanges`, de CodePipeline forma predeterminada sondea tu ubicación de origen para ver si hay cambios en el origen. Este comportamiento es el mismo que si se incluye `PollForSourceChanges` y se establece en `true`.

- `false`: Si se establece, CodePipeline no sondea la ubicación de origen para ver si hay cambios en el origen. Utilice esta configuración si desea configurar una regla de CloudWatch eventos para detectar los cambios en el origen.

## Artefactos de entrada

- Número de artefactos: `0`
- Descripción: los artefactos de entrada no se aplican a este tipo de acción.

## Artefactos de salida

- Número de artefactos: `1`
- Descripción: proporciona los artefactos que están disponibles en el bucket de origen configurado para conectarse a la canalización. Los artefactos generados a partir del bucket son los artefactos de salida de la acción de Amazon S3. Los metadatos del objeto de Amazon S3 (ETag e ID de

versión) se muestran CodePipeline como la revisión de la fuente para la ejecución del proceso desencadenado.

## Variables de salida

Cuando se configura, esta acción produce variables a las que se puede hacer referencia mediante la configuración de acción de una acción descendente en la canalización. Esta acción produce variables que se pueden ver como variables de salida, incluso si la acción no tiene un espacio de nombres. Configure una acción con un espacio de nombres para que esas variables estén disponibles para la configuración de las acciones posteriores.

Para obtener más información sobre las variables en CodePipeline, consulteReferencia de variables.

BucketName

El nombre del bucket de Amazon S3 relacionado con el cambio de origen que desencadenó la canalización.

ETag

La etiqueta de entidad del objeto relacionado con el cambio de código fuente que desencadenó la canalización. El ETag es un hash MD5 del objeto. ETag solo refleja los cambios en el contenido de un objeto, no los metadatos.

ObjectKey

El nombre de la clave del objeto de Amazon S3 relacionado con el cambio de origen que desencadenó la canalización.

VersionId

El ID de versión de la versión del objeto relacionada con el cambio de código fuente que desencadenó la canalización.

## Permisos del rol de servicio: acción de origen de S3

Para la compatibilidad con la acción de origen de S3, agregue lo siguiente a su declaración de política:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketVersioning",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::[[S3Bucket]]",
        "arn:aws:s3:::[[S3Bucket]]/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

## Declaración de acciones

YAML

```
Name: Source
Actions:
  - RunOrder: 1
    OutputArtifacts:
      - Name: SourceArtifact
    ActionTypeId:
      Provider: S3
      Owner: AWS
```

```
      Version: '1'
      Category: Source
    Region: us-west-2
    Name: Source
    Configuration:
      S3Bucket: amzn-s3-demo-source-bucket
      S3ObjectKey: my-application.zip
      PollForSourceChanges: 'false'
    InputArtifacts: []
```

JSON

```
{
    "Name": "Source",
    "Actions": [
        {
            "RunOrder": 1,
            "OutputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "ActionTypeId": {
                "Provider": "S3",
                "Owner": "AWS",
                "Version": "1",
                "Category": "Source"
            },
            "Region": "us-west-2",
            "Name": "Source",
            "Configuration": {
                "S3Bucket": "amzn-s3-demo-source-bucket",
                "S3ObjectKey": "my-application.zip",
                "PollForSourceChanges": "false"
            },
            "InputArtifacts": []
        }
    ]
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Tutorial: Crear una canalización simple (bucket de S3)— Este tutorial proporciona un AppSpec archivo de ejemplo y un grupo de CodeDeploy aplicaciones y despliegues de ejemplo. Utilice este tutorial para crear una canalización con un origen de Amazon S3 que se implementa en instancias de Amazon EC2.

# Referencia de la acción de implementación de AWS AppConfig

AWS AppConfig es una capacidad de AWS Systems Manager. AppConfig admite despliegues controlados en aplicaciones de cualquier tamaño e incluye controles de validación y monitoreo integrados. Puede usarlo AppConfig con aplicaciones alojadas en instancias, contenedores AWS Lambda, aplicaciones móviles o dispositivos IoT de Amazon EC2.

La acción de `AppConfig` despliegue es una AWS CodePipeline acción que despliega las configuraciones almacenadas en la ubicación de origen de la canalización en una AppConfig aplicación, un entorno y un perfil de configuración específicos. Utiliza las preferencias definidas en una estrategia de AppConfig implementación.

## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `AppConfig`
- Versión: `1`

## Parámetros de configuración

Application

Obligatorio: sí

El ID de la AWS AppConfig aplicación con los detalles de la configuración y el despliegue.

Environment

Obligatorio: sí

El ID del AWS AppConfig entorno en el que se implementa la configuración.

ConfigurationProfile

Obligatorio: sí

El ID del perfil de AWS AppConfig configuración que se va a implementar.

InputArtifactConfigurationPath

Obligatorio: sí

La ruta del archivo de los datos de configuración del artefacto de entrada que se va a implementar.

DeploymentStrategy

Obligatorio: no

La estrategia AWS AppConfig de despliegue que se utilizará para el despliegue.

## Artefactos de entrada

- Número de artefactos: 1
- Descripción: el artefacto de entrada para la acción de implementación.

## Artefactos de salida

No se usa.

## Permisos del rol de servicio: acción de **AppConfig**

Cuando CodePipeline se ejecuta la acción, la política de roles de CodePipeline servicio requiere los siguientes permisos, con el alcance adecuado hasta el nivel de recurso, a fin de mantener el acceso con el mínimo de privilegios.

JSON

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

```
    "Statement": [
        {
            "Action": [
                "appconfig:StartDeployment",
                "appconfig:StopDeployment",
                "appconfig:GetDeployment"
            ],
            "Resource": [
                "arn:aws:appconfig:*:111122223333:application/[[Application]]",
                "arn:aws:appconfig:*:111122223333:application/[[Application]]/*",
                "arn:aws:appconfig:*:111122223333:deploymentstrategy/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

## Ejemplo de configuración de una acción

### 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"
        }
    ]
}
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- AWS AppConfig— Para obtener información sobre AWS AppConfig las implementaciones, consulte la Guía del AWS Systems Manager usuario.
- Tutorial: Crear una canalización que se utilice AWS AppConfig como proveedor de despliegue— Este tutorial le permite empezar a configurar archivos y AppConfig recursos de configuración de

despliegues sencillos y le muestra cómo utilizar la consola para crear una canalización con una acción de AWS AppConfig despliegue.

# Referencia de la acción de implementación de CloudFormation

Ejecuta una operación en una CloudFormation pila. Una pila es un conjunto de AWS recursos que se pueden gestionar como una sola unidad. Los recursos de una pila se definen según la plantilla de CloudFormation de la pila. Un conjunto de cambios crea una comparación que se puede visualizar sin modificar la pila original. Para obtener información sobre los tipos de CloudFormation acciones que se pueden realizar en las pilas y los conjuntos de cambios, consulte el `ActionMode` parámetro.

Para crear un mensaje de error para una CloudFormation acción en la que se ha producido un error en una operación de apilado, CodePipeline llama a la CloudFormation `DescribeStackEvents` API. Si un rol de IAM de acción tiene permiso para acceder a esa API, los detalles sobre el primer recurso fallido se incluirán en el mensaje de CodePipeline error. De lo contrario, si la política de roles no tiene el permiso adecuado, CodePipeline ignorará el acceso a la API y, en su lugar, mostrará un mensaje de error genérico. Para ello, el permiso `cloudformation:DescribeStackEvents` debe agregarse al rol de servicio o a otros roles de IAM de la canalización.

Si no desea que los detalles del recurso aparezcan en los mensajes de error de la canalización, puede eliminar el permiso `cloudformation:DescribeStackEvents` para revocar este permiso para el rol de IAM de acción.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Permisos de rol de servicio: CloudFormation acción
- Declaración de acciones
- Véase también

## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `CloudFormation`
- Versión: `1`

## Parámetros de configuración

ActionMode

Obligatorio: sí

`ActionMode`es el nombre de la acción que se CloudFormation realiza en una pila o conjunto de cambios. Están disponibles los siguientes modos de acción:

- `CHANGE_SET_EXECUTE` ejecuta un conjunto de cambios para la pila de recursos que se basa en un conjunto de actualizaciones de recursos especificadas. Con esta acción, CloudFormation comienza a alterar la pila.
- `CHANGE_SET_REPLACE` crea el conjunto de cambios, si no existe, basándose en el nombre de la pila y la plantilla que envíe. Si el conjunto de cambios existe, lo CloudFormation elimina y, a continuación, crea uno nuevo.
- `CREATE_UPDATE` crea la pila si no existe. Si la pila existe, CloudFormation actualiza la pila. Utilice esta acción para actualizar pilas existentes. Por el contrario`REPLACE_ON_FAILURE`, si la pila existe y se encuentra en un estado fallido, CodePipeline no la eliminará ni la reemplazará.
- `DELETE_ONLY` elimina una pila. Si especifica una pila que no existe, la acción se realiza de forma satisfactoria sin eliminar una pila.
- `REPLACE_ON_FAILURE` crea una pila si no existe. Si la pila existe y se encuentra en un estado fallido, la CloudFormation elimina y, a continuación, crea una nueva pila. Si la pila no está en un estado fallido, la CloudFormation actualiza.

  La pila está en un estado de error cuando se muestra alguno de los siguientes tipos de estado en CloudFormation:

  - `ROLLBACK_FAILED`
  - `CREATE_FAILED`
  - `DELETE_FAILED`

- `UPDATE_ROLLBACK_FAILED`

Utilice esta acción para sustituir automáticamente pilas fallidas sin recuperarlos ni solucionar sus problemas.

> **Important**
>
> Le recomendamos que utilice `REPLACE_ON_FAILURE` solo con fines de prueba, ya que podría eliminar la pila.

StackName

Obligatorio: sí

`StackName` es el nombre de una pila existente o una pila que desea crear.

Capacidades

Obligatorio: condicional

El uso de `Capabilities` reconoce que la plantilla podría tener las capacidades de crear y actualizar algunos recursos por sí misma y que estas capacidades se determinan en función de los tipos de recursos de la plantilla.

Esta propiedad es necesaria si tiene recursos de IAM en la plantilla de pila o si crea una pila directamente desde una plantilla que contiene macros. Para que la CloudFormation acción funcione correctamente de esta manera, debe reconocer explícitamente que desea que lo haga con una de las siguientes capacidades:

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`
- `CAPABILITY_AUTO_EXPAND`

Puede especificar más de una capacidad mediante el uso de una coma (sin espacio) entre las capacidades. El ejemplo de Declaración de acciones muestra una entrada con las propiedades CAPABILITY_IAM y CAPABILITY_AUTO_EXPAND.

Para obtener más información al respecto`Capabilities`, consulte las propiedades que aparecen UpdateStacken la referencia de la AWS CloudFormation API.

ChangeSetName

Obligatorio: condicional

`ChangeSetName` es el nombre de un conjunto de cambios existentes o un nuevo conjunto de cambios que desee crear para la pila especificada.

Esta propiedad es necesaria para los siguientes modos de acción: CHANGE_SET_REPLACE y CHANGE_SET_EXECUTE. Para todos los demás modos de acción, se pasa por alto esta propiedad.

RoleArn

Obligatorio: condicional

El `RoleArn` es el ARN del rol de servicio de IAM que CloudFormation asume cuando opera en los recursos de la pila especificada. `RoleArn` no se aplica al ejecutar un conjunto de cambios. Si no lo utiliza CodePipeline para crear el conjunto de cambios, asegúrese de que el conjunto o la pila de cambios tengan una función asociada.

> **Note**
>
> Este rol debe estar en la misma cuenta que el role de la acción que se está ejecutando, tal como se configura en la declaración de acción `RoleArn`.

Esta propiedad es necesaria para los siguientes modos de acción:

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- DELETE_ONLY
- CHANGE_SET_REPLACE

> **Note**
>
> CloudFormation recibe una URL de la plantilla firmada por S3; por lo tanto, `RoleArn` no necesita permiso para acceder al depósito de artefactos. Sin embargo, la acción `RoleArn` necesita permiso para acceder al bucket de artefactos para generar la URL firmada.

TemplatePath

Obligatorio: condicional

`TemplatePath`representa el archivo de plantilla CloudFormation . Incluya el archivo en un artefacto de entrada en esta acción. El nombre del archivo sigue este formato:

*`Artifactname`*`::`*`TemplateFileName`*

`Artifactname` es el nombre del artefacto de entrada, tal y como aparece en CodePipeline. Por ejemplo, una etapa de origen con el nombre de artefacto de `SourceArtifact` y un nombre de archivo `template-export.json` crea un nombre `TemplatePath`, tal y como se muestra en este ejemplo:

```
"TemplatePath": "SourceArtifact::template-export.json"
```

Esta propiedad es necesaria para los siguientes modos de acción:

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- CHANGE_SET_REPLACE

Para todos los demás modos de acción, se pasa por alto esta propiedad.

> **Note**
>
> El archivo de CloudFormation plantilla que contiene el cuerpo de la plantilla tiene una longitud mínima de 1 byte y una longitud máxima de 1 MB. Para las acciones de CloudFormation despliegue en CodePipeline, el tamaño máximo del artefacto de entrada es siempre de 256 MB. Para obtener más información, consulte Cuotas en AWS CodePipeline y Límites de CloudFormation.

OutputFileName

Obligatorio: no

Se utiliza `OutputFileName` para especificar un nombre de archivo de salida, por ejemplo`CreateStackOutput.json`, que se CodePipeline añada al artefacto de salida de la canalización para esta acción. El archivo JSON contiene el contenido de la `Outputs` sección de la CloudFormation pila.

Si no especificas un nombre, CodePipeline no generará un archivo o artefacto de salida.

ParameterOverrides

Obligatorio: no

Los parámetros se definen en la plantilla de pila y le permiten proporcionarles valores en el momento de la creación o actualización de la pila. Puede utilizar un objeto JSON para establecer valores de parámetros en la plantilla (estos valores anulan los establecidos en el archivo de configuración de la plantilla). Para obtener más información sobre el uso de anulaciones de parámetros, consulte Propiedades de configuración (objeto JSON).

Le recomendamos que use el archivo de configuración de la plantilla para la mayoría de los valores de parámetros. Utilice anulaciones de parámetros solo para valores que no se conocen hasta que la canalización se está ejecutando. Para obtener más información, consulte Uso de funciones de anulación de parámetros con CodePipeline canalizaciones en la Guía del AWS CloudFormation usuario.

> **Note**
>
> Todos los nombres de parámetros deben estar presentes en la plantilla de pila.

TemplateConfiguration

Obligatorio: no

`TemplateConfiguration` es el archivo de configuración de la plantilla. Incluya el archivo en un artefacto de entrada en esta acción. Puede incluir valores de parámetros de plantilla y una política de pilas. Para obtener más información sobre el formato del archivo de configuración de plantilla, consulte Artefactos de AWS CloudFormation.

El nombre del archivo de configuración de la plantilla se ajusta a este formato:

*`Artifactname`*`::`*`TemplateConfigurationFileName`*

`Artifactname` es el nombre del artefacto de entrada, tal y como aparece en CodePipeline. Por ejemplo, una etapa de origen con el nombre de artefacto de `SourceArtifact` y un nombre de archivo `test-configuration.json` crea un nombre `TemplateConfiguration`, tal y como se muestra en este ejemplo:

```
"TemplateConfiguration": "SourceArtifact::test-configuration.json"
```

## Artefactos de entrada

- Número de artefactos: `0 to 10`
- Descripción: Como entrada, la CloudFormation acción acepta artefactos de forma opcional para los siguientes fines:
  - Para proporcionar el archivo de plantilla de pila que se va a ejecutar (consulte el parámetro `TemplatePath`).
  - Para proporcionar el archivo de configuración de la plantilla que se va a utilizar (consulte el parámetro `TemplateConfiguration`). Para obtener más información sobre el formato del archivo de configuración de plantilla, consulte Artefactos de AWS CloudFormation.
  - Proporcionar el artefacto para que una función Lambda se despliegue como parte de la CloudFormation pila.

## Artefactos de salida

- Número de artefactos: `0 to 1`
- Descripción: si se especifica el parámetro `OutputFileName`, esta acción produce un artefacto de salida que incluye un archivo JSON con el nombre especificado. El archivo JSON incluye el contenido de la sección Outputs de la pila de CloudFormation .

  Para obtener más información acerca de la sección Outputs que puede crear para la acción de CloudFormation , consulte Salidas.

## Variables de salida

Cuando se configura, esta acción produce variables a las que se puede hacer referencia mediante la configuración de acción de una acción descendente en la canalización. Configure una acción con un espacio de nombres para que esas variables estén disponibles para la configuración de las acciones posteriores.

En el caso de las CloudFormation acciones, las variables se generan a partir de cualquier valor designado en la `Outputs` sección de una plantilla de pila. Tenga en cuenta que los únicos modos de CloudFormation acción que generan resultados son aquellos que dan como resultado la creación o actualización de una pila, como la creación de la pila, las actualizaciones de la pila y la ejecución de conjuntos de cambios. Los modos de acción correspondientes que generan variables son:

- CHANGE_SET_EXECUTE
- CHANGE_SET_REPLACE
- CREATE_UPDATE
- REPLACE_ON_FAILURE

Para obtener más información, consulte Referencia de variables. Para ver un tutorial que muestra cómo crear una canalización con una acción de CloudFormation despliegue en una canalización que utilice variables CloudFormation de salida, consulteTutorial: Cree una canalización que utilice variables de AWS CloudFormation acciones de despliegue.

## Permisos de rol de servicio: CloudFormation acción

Cuando CodePipeline se ejecuta la acción, la política de rol de CodePipeline servicio requiere los siguientes permisos, que se limitan adecuadamente al ARN del recurso de canalización para mantener el acceso con los mínimos privilegios. Por ejemplo, agregue lo siguiente a su declaración de política:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCFNStackAccess",
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStacks",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStackEvents",
                "cloudformation:GetTemplate",
                "cloudformation:DescribeChangeSet",
                "cloudformation:CreateChangeSet",
                "cloudformation:DeleteChangeSet",
                "cloudformation:ExecuteChangeSet"
            ],
            "Resource": [
```

```
                "arn:aws:cloudformation:*:111122223333:stack/
[[cfnDeployStackNames]]/*"
            ]
        },
        {
            "Sid": "ValidateTemplate",
            "Effect": "Allow",
            "Action": [
                "cloudformation:ValidateTemplate"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowIAMPassRole",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/[[cfnExecutionRoles]]"
            ],
            "Condition": {
                "StringEqualsIfExists": {
                    "iam:PassedToService": [
                        "cloudformation.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Tenga en cuenta que el permiso `cloudformation:DescribeStackEvents` es opcional. Permite que la CloudFormation acción muestre un mensaje de error más detallado. Puede revocar este permiso desde el rol de IAM si no quiere que los detalles de los recursos aparezcan en los mensajes de error de la canalización.

## Declaración de acciones

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"
        }
    ]
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Referencia sobre las propiedades de configuración: este capítulo de referencia de la Guía del AWS CloudFormation usuario proporciona más descripciones y ejemplos de estos CodePipeline parámetros.
- AWS CloudFormation Referencia de la API: el CreateStackparámetro de la referencia de la AWS CloudFormation API describe los parámetros de pila de CloudFormation las plantillas.

# CloudFormation Referencia de la acción de implementación de StackSets

CodePipeline ofrece la posibilidad de realizar CloudFormation StackSets operaciones como parte de su CI/CD proceso. Utiliza un conjunto de pilas para crear pilas en AWS cuentas de distintas AWS regiones mediante una sola CloudFormation plantilla. Todos los recursos incluidos en cada pila se definen en la CloudFormation plantilla del conjunto de pilas. Al crear el conjunto de pilas, debe especificar la plantilla que desea utilizar, así como todos los parámetros y capacidades que requiere la plantilla.

Para obtener más información sobre los conceptos de CloudFormation StackSets, consulte StackSets los conceptos en la Guía del AWS CloudFormation usuario.

La canalización se integra CloudFormation StackSets mediante dos tipos de acción distintos que se utilizan de forma conjunta:

- La acción `CloudFormationStackSet` crea o actualiza un conjunto de pilas o instancias de pilas a partir de la plantilla almacenada en la ubicación de origen de la canalización. Cada vez que se crea o actualiza un conjunto de pilas, se inicia una implementación de esos cambios en instancias

específicas. En la consola, puedes elegir el proveedor de acciones de CloudFormation Stack Set al crear o editar tu canalización.

- La acción `CloudFormationStackInstances` implementa los cambios de la acción `CloudFormationStackSet` en instancias específicas, crea nuevas instancias de pila y define las anulaciones de parámetros en las instancias especificadas. En la consola, puedes elegir el proveedor de acciones de CloudFormation Stack Instances al editar una canalización existente.

Puedes usar estas acciones para implementarlas en AWS cuentas objetivo o en los ID de las unidades organizativas de la AWS organización objetivo.

> **Note**
>
> Para desplegarlas en cuentas de AWS organizaciones o identificadores de unidades organizativas de destino y usar el modelo de permisos administrados por el servicio, debes habilitar el acceso confiable entre organizaciones AWS CloudFormation StackSets y AWS organizaciones. Para obtener más información, consulta Cómo habilitar el acceso confiable con AWS CloudFormation Stacksets.

Temas


- Cómo CloudFormation StackSets las acciones funcionan
- Cómo estructurar StackSets las acciones en una canalización
- Acción CloudFormationStackSet
- La acción CloudFormationStackInstances
- Permisos del rol de servicio: acción de CloudFormationStackSet
- Permisos del rol de servicio: acción CloudFormationStackInstances
- Modelos de permisos para operaciones de conjunto de pilas
- Tipos de datos de parámetros de plantilla
- Véase también


## Cómo CloudFormation StackSets las acciones funcionan

Una acción `CloudFormationStackSet` crea o actualiza recursos en función de si la acción se ejecuta por primera vez.

La acción `CloudFormationStackSet` crea o actualiza el conjunto de pilas e implementa esos cambios en instancias específicas.

> **Note**
>
> Si utiliza esta acción para realizar una actualización que incluya la adición de instancias de pila, las nuevas instancias se implementan primero y la actualización se completa en último lugar. Las nuevas instancias reciben primero la versión anterior y, a continuación, la actualización se aplica a todas las instancias.

- Crear: cuando no se especifica ninguna instancia y el conjunto de pilas no existe, la CloudFormationStackSet acción crea el conjunto de pilas sin crear ninguna instancia.
- Actualización: cuando la CloudFormationStackSet acción se ejecuta para un conjunto de pilas que ya está creado, la acción actualiza el conjunto de pilas. Si no se especifica ninguna instancia y el conjunto de pilas ya existe, se actualizan todas las instancias. Si esta acción se usa para actualizar instancias específicas, todas las instancias restantes pasarán a un estado OBSOLETO.

  Puede usar la CloudFormationStackSet acción para actualizar el conjunto de pilas de las siguientes maneras.

  - Actualice la plantilla en algunas o todas las instancias.
  - Actualice los parámetros en algunas o todas las instancias.
  - Actualice el rol de ejecución del conjunto de pilas (debe coincidir con el rol de ejecución especificado en el rol de administrador).
  - Cambie el modelo de permisos (solo si no se ha creado ninguna instancia).
  - Enable/Disable `AutoDeployments`si el modelo de permisos del conjunto de pilas es`Service Managed`.
  - Actúa como administrador delegado en una cuenta de miembro si el modelo de permisos del conjunto de pilas es `Service Managed`.
  - Actualice el rol de administrador.
  - Actualice la descripción del conjunto de pila.
  - Añada los objetivos de implementación a la actualización del conjunto de pilas para crear nuevas instancias de pila.

La acción `CloudFormationStackInstances` crea nuevas instancias de pila o actualiza las instancias de pila obsoletas. Una instancia queda desactualizada cuando se actualiza un conjunto de pilas, pero no se actualizan todas las instancias que contiene.

- Crear: si la pila ya existe, la acción `CloudFormationStackInstances` solo actualiza las instancias y no crea instancias de pila.
- Actualizar: una vez realizada la acción `CloudFormationStackSet`, si la plantilla o los parámetros se han actualizado solo en algunos casos, el resto se marcará `OUTDATED`. En fases posteriores de la canalización, `CloudFormationStackInstances` actualiza el resto de las instancias del conjunto de pilas en oleadas para que todas las instancias queden marcadas `CURRENT`. Esta acción también se puede utilizar para añadir instancias adicionales o anular los parámetros de las instancias nuevas o existentes.

Como parte de una actualización, las acciones `CloudFormationStackSet` y `CloudFormationStackInstances` pueden especificar nuevos objetivos de implementación, lo que crea nuevas instancias apiladas.

Como parte de una actualización, las acciones `CloudFormationStackSet` y `CloudFormationStackInstances` no eliminan conjuntos de pilas, instancias o recursos. Cuando la acción actualiza una pila pero no especifica todas las instancias que se van a actualizar, las instancias que no se especificaron para la actualización se eliminan de la actualización y se establecen en un estado de `OUTDATED`.

Durante una implementación, las instancias de la pila también pueden mostrar un estado de `OUTDATED` en el que se indica si la implementación en las instancias ha fallado.

## Cómo estructurar StackSets las acciones en una canalización

Como práctica recomendada, debe construir su canalización de manera que el conjunto de pilas se cree y se implemente inicialmente en un subconjunto o en una sola instancia. Tras probar la implementación y ver el conjunto de pilas generado, añada la acción `CloudFormationStackInstances` para crear y actualizar las instancias restantes.

Utilice la consola o la CLI para crear la estructura de canalización recomendada de la siguiente manera:

1. Cree una canalización con una acción de origen (obligatoria) y la acción `CloudFormationStackSet` como acción de implementación. Ejecute su canalización.

2. Cuando su canalización se ejecuta por primera vez, la acción `CloudFormationStackSet` crea su conjunto de pilas y al menos una instancia inicial. Verifique la creación del conjunto de pilas y revise la implementación en su instancia inicial. Por ejemplo, para la creación inicial de un conjunto de pilas para una cuenta Account-A en la que `us-east-1` esté la región especificada, la instancia de pila se crea con el conjunto de pilas:

| Instancia de pila | Region | Status |
| --- | --- | --- |
| StackInstanceID-1 | us-east-1 | CURRENT |

3. Edita tu canalización para `CloudFormationStackInstances` añadirla como segunda acción de despliegue a create/update apilar instancias para los objetivos que designes. Por ejemplo, para crear una instancia de pila para la cuenta `Account-A` en la que se especifican las regiones `us-east-2` y `eu-central-1`, se crean las instancias de pila restantes y la instancia inicial permanece actualizada de la siguiente manera:

| Instancia de pila | Region | Status |
| --- | --- | --- |
| StackInstanceID-1 | us-east-1 | CURRENT |
| StackInstanceID-2 | us-east-2 | CURRENT |
| StackInstanceID-3 | eu-central-1 | CURRENT |

4. Ejecute su canalización según sea necesario para actualizar su conjunto de pilas y actualizar o crear instancias de pila.

Cuando inicie una actualización de pila en la que haya eliminado los objetivos de implementación de la configuración de la acción, las instancias de la pila que no estaban designadas para la actualización se eliminan de la implementación y pasan a un estado OUTDATED. Por ejemplo, para actualizar una instancia de pila para una cuenta `Account-A` en la que la `us-east-2` región se elimina de la configuración de la acción, se crean las instancias de pila restantes y la instancia eliminada se establece en OBSOLETA de la siguiente manera:

| Instancia de pila | Region | Status |
| --- | --- | --- |
| StackInstanceID-1 | us-east-1 | CURRENT |

| Instancia de pila | Region | Status |
| --- | --- | --- |
| StackInstanceID-2 | us-east-2 | OUTDATED |
| StackInstanceID-3 | eu-central-1 | CURRENT |

Para obtener más información sobre las prácticas recomendadas para implementar conjuntos de pilas, consulte las prácticas https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-bestpractices.html recomendadas StackSets en la Guía del AWS CloudFormation usuario.

## Acción **`CloudFormationStackSet`**

Esta acción crea o actualiza un conjunto de pilas a partir de la plantilla almacenada en la ubicación de origen de la canalización.

Después de definir un conjunto de pilas, puede crear, actualizar o eliminar pilas en las cuentas y regiones de destino especificadas en los parámetros de configuración. Al crear, actualizar y eliminar pilas, puede especificar otras preferencias, como el orden de las regiones para que se realicen las operaciones, el porcentaje de tolerancia a fallos a partir del cual se detienen las operaciones en las pilas y el número de cuentas en las que se realizan operaciones en las pilas de forma concurrente.

Un conjunto de pilas es un recurso regional. Si crea un conjunto de pilas en una AWS región, no podrá acceder a él desde otras regiones.

Cuando esta acción se utiliza como una acción de actualización del conjunto de pilas, no se permiten las actualizaciones de la pila sin una implementación en al menos una instancia de la pila.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Ejemplo CloudFormationStackSet configuración de la acción

## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `CloudFormationStackSet`
- Versión: `1`

## Parámetros de configuración

StackSetName

Obligatorio: sí

El nombre que se va a asociar con el conjunto de pila. Este nombre debe ser único en la región donde se crea.

Solo puede contener caracteres alfanuméricos y guiones. Debe comenzar por una letra y tener 128 caracteres o menos.

Description (Descripción)

Obligatorio: no

Una descripción del conjunto de pila. Puede usarlo para describir el propósito del conjunto de pilas u otra información relevante.

TemplatePath

Obligatorio: sí

La ubicación de la plantilla que define los recursos del conjunto de pila. Debe apuntar a una plantilla con un tamaño máximo de 460 800 bytes.

Introduzca la ruta del nombre del artefacto de origen y el archivo de plantilla en el formato `"InputArtifactName::TemplateFileName"`, como se muestra en el siguiente ejemplo.

```
SourceArtifact::template.txt
```

Parameters

Obligatorio: no

Una lista de parámetros de plantilla para el conjunto de pilas que se actualizan durante una implementación.

Puede proporcionar parámetros como una lista literal o una ruta de archivo:

- Puede introducir parámetros en el siguiente formato sintáctico abreviado: `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV` Para obtener más información acerca de estos tipos de datos, consulte Tipos de datos de parámetros de plantilla.

  El siguiente ejemplo muestra un parámetro denominado `BucketName` con el valor `amzn-s3-demo-source-bucket`.

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ```

  En el siguiente ejemplo, se muestra una entrada con varios parámetros:

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ParameterKey=Asset1,ParameterValue=true
  ParameterKey=Asset2,ParameterValue=true
  ```

- Puede introducir la ubicación del archivo que contiene una lista de modificaciones de parámetros de plantilla introducidas en el formato `"InputArtifactName::ParametersFileName"`, como se muestra en el siguiente ejemplo.

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

  En el siguiente ejemplo se muestra el contenido del archivo para `parameters.txt`.

  ```
  [
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      },
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      }
  ```

```
]
```

Capacidades

Obligatorio: no

Indica que la plantilla puede crear y actualizar recursos, en función de los tipos de recursos de la plantilla.

Debe utilizar esta propiedad si tiene recursos IAM en su plantilla de pila o si crea una pila directamente a partir de una plantilla que contenga macros. Para que la CloudFormation acción funcione correctamente de esta manera, debes usar una de las siguientes funciones:

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`

Puede especificar más de una capacidad mediante el uso de una coma sin espacios entre las capacidades. El ejemplo de Ejemplo CloudFormationStackSet configuración de la acción muestra una entrada con múltiples capacidades.

PermissionModel

Obligatorio: no

Determina cómo se crean y administran los roles de IAM. Si no se especifica el campo, se usa valor predeterminado. Para obtener información, consulte Modelos de permisos para operaciones de conjunto de pilas.

Los valores válidos son los siguientes:

- `SELF_MANAGED` (predeterminado): debe crear roles de administrador y ejecución para implementarlos en las cuentas de destino.
- `SERVICE_MANAGED`: crea CloudFormation StackSets automáticamente las funciones de IAM necesarias para implementarlas en las cuentas administradas por AWS las organizaciones. Esto requiere una cuenta para ser miembro de una organización.

> **Note**
>
> Este parámetro solo se puede cambiar cuando no hay instancias de pila en el conjunto de pilas.

AdministrationRoleArn

> **Note**
>
> Dado que CloudFormation StackSets realiza operaciones en varias cuentas, debes definir los permisos necesarios en esas cuentas antes de poder crear el conjunto de pilas.

Obligatorio: no

> **Note**
>
> Este parámetro es opcional para el modelo de permisos SELF_MANAGED y no se usa para el modelo de permisos SERVICE_MANAGED.

El ARN del rol de IAM en la cuenta de administrador utilizada para realizar las operaciones del conjunto de pilas.

El nombre puede contener caracteres alfanuméricos, cualquiera de los siguientes caracteres: _ +=, .@- y no incluir espacios. El nombre distingue entre mayúsculas y minúsculas. El nombre del rol debe tener una longitud mínima de 20 caracteres y una longitud máxima de 2048 caracteres. Los nombres del rol deben ser únicos dentro de la cuenta. El nombre del rol especificado aquí debe ser un nombre de rol existente. Si no especifica el nombre del rol, se establece en AWSCloudFormationStackSetAdministrationRole. Si lo especifica ServiceManaged, no debe definir un nombre de rol.

ExecutionRoleName

> **Note**
>
> Dado que CloudFormation StackSets realiza operaciones en varias cuentas, debe definir los permisos necesarios en esas cuentas antes de poder crear el conjunto de pilas.

Obligatorio: no

> **Note**
>
> Este parámetro es opcional para el modelo de permisos SELF_MANAGED y no se usa para el modelo de permisos SERVICE_MANAGED.

El nombre del rol de IAM en las cuentas de destino utilizadas para realizar las operaciones del conjunto de pilas. El nombre puede contener caracteres alfanuméricos, cualquiera de los siguientes caracteres: _+=, .@- y no incluir espacios. El nombre distingue entre mayúsculas y minúsculas. Este nombre de rol debe tener una longitud mínima de 1 carácter y máxima de 64 caracteres. Los nombres del rol deben ser únicos dentro de la cuenta. El nombre del rol especificado aquí debe ser un nombre de rol existente. No especifique este rol si utiliza roles de ejecución personalizados. Si no especifica el nombre del rol, se establece en `AWSCloudFormationStackSetExecutionRole`. Si lo establece Service_Managed en verdadero, no debe definir un nombre de rol.

OrganizationsAutoDeployment

Obligatorio: no

> **Note**
>
> Este parámetro es opcional para el modelo de permisos SERVICE_MANAGED y no se usa para el modelo de permisos SELF_MANAGED.

Describe si CloudFormation StackSets se implementa automáticamente en la AWS organización las cuentas que se agregan a una organización o unidad organizativa (OU) de destino. Si se especifica `OrganizationsAutoDeployment`, no especifique `DeploymentTargets` ni `Regions`.

> **Note**
>
> Si no se proporciona ninguna entrada para `OrganizationsAutoDeployment`, el valor predeterminado es `Disabled`.

Los valores válidos son los siguientes:

- `Enabled`. Obligatorio: no.

StackSets implementa automáticamente instancias de pila adicionales en AWS las cuentas de las organizaciones que se agregan a una organización o unidad organizativa (OU) de destino en las regiones especificadas. Si se elimina una cuenta de una organización u organización organizativa de destino, CloudFormation StackSets elimina las instancias de pila de la cuenta en las regiones especificadas.

- `Disabled`. Obligatorio: no.

  StackSets no implementa automáticamente instancias de pila adicionales en AWS las cuentas de las organizaciones que se agregan a una organización o unidad organizativa (OU) de destino en las regiones especificadas.

- `EnabledWithStackRetention`. Obligatorio: no.

  Los recursos de pila se conservan cuando se elimina una cuenta de una organización u OU de destino.

DeploymentTargets

Obligatorio: no

> **Note**
>
> Para el modelo de permisos SERVICE_MANAGED, puede proporcionar el ID raíz de la organización o los ID de la unidad organizativa para los objetivos de implementación. Para el modelo de permisos SELF_MANAGED, solo puede proporcionar cuentas.

> **Note**
>
> Cuando se selecciona este parámetro, también debe seleccionar Regiones.

Una lista de AWS las cuentas o identificadores de unidades organizativas en las que deberían estar las instancias del conjunto de pilas created/updated.

- Cuentas

  Puede proporcionar las cuentas como una lista literal o como una ruta de archivo:

  - Literal: introduzca los parámetros en el formato de sintaxis abreviada `account_ID,account_ID`, como se muestra en el siguiente ejemplo.

```
111111222222,333333444444
```

- Ruta del archivo: la ubicación del archivo que contiene la lista de AWS cuentas en las que deben estar las instancias del conjunto de pilas created/updated, introducida en el formato`InputArtifactName::AccountsFileName`. Si utilizas la ruta del archivo para especificar cuentas o OrganizationalUnitIds, el formato del archivo debe estar en JSON, como se muestra en el siguiente ejemplo.

  ```
  SourceArtifact::accounts.txt
  ```

  En el siguiente ejemplo se muestra el contenido del archivo para `accounts.txt`.

  ```
  [
      "111111222222"
  ]
  ```

  En el siguiente ejemplo, se muestra el contenido del archivo para incluir más de una cuenta de `accounts.txt`:

  ```
  [
      "111111222222","333333444444"
  ]
  ```

- OrganizationalUnitIds:

  > **Note**
  >
  > Este parámetro es opcional para el modelo de permisos SERVICE_MANAGED y no se usa para el modelo de permisos SELF_MANAGED. No lo utilices si seleccionas OrganizationsAutoDeployment.

  Las unidades AWS organizativas en las que se actualizarán las instancias de pila asociadas.

  Puede proporcionar los ID de las unidades organizativas como una lista literal o una ruta de archivo:

  - Literal: introduzca una matriz de cadenas separadas por comas, como se muestra en el siguiente ejemplo.

```
ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
```

- Ruta del archivo: la ubicación del archivo que contiene una lista de instancias OrganizationalUnitIds en las que se pueden crear o actualizar las instancias del conjunto de pilas. Si utilizas la ruta del archivo para especificar cuentas o OrganizationalUnitIds, el formato del archivo debe estar en JSON, como se muestra en el siguiente ejemplo.

  Introduzca una ruta al archivo con el formato `InputArtifactName::OrganizationalUnitIdsFileName`.

  ```
  SourceArtifact::OU-IDs.txt
  ```

  En el siguiente ejemplo se muestra el contenido del archivo para `OU-IDs.txt`.

  ```
  [
      "ou-examplerootid111-exampleouid111","ou-examplerootid222-exampleouid222"
  ]
  ```

Regiones

Obligatorio: no

> **Note**
>
> Cuando se selecciona este parámetro, también debe seleccionarlo DeploymentTargets.

Una lista de AWS regiones en las que se crean o actualizan las instancias de conjuntos de pilas. Las regiones se actualizan en el orden en el que se han introducido.

Introduzca una lista de AWS regiones válidas en el formato `Region1,Region2` que se muestra en el siguiente ejemplo.

```
us-west-2,us-east-1
```

FailureTolerancePercentage

Obligatorio: no

El porcentaje de cuentas por región en las que esta operación de apilamiento puede fallar antes CloudFormation de detener la operación en esa región. Si la operación se detiene en una región, CloudFormation no intente realizarla en las regiones siguientes. Al calcular el número de cuentas en función del porcentaje especificado, CloudFormation redondea a la baja al siguiente número entero.

MaxConcurrentPercentage

Obligatorio: no

El porcentaje máximo de cuentas en las que realizar esta operación de una vez. Al calcular el número de cuentas en función del porcentaje especificado, CloudFormation redondea a la baja al siguiente número entero. Si el redondeo a la baja da como resultado cero, CloudFormation establece el número como uno en su lugar. Si bien usa esta configuración para especificar el máximo, el número real de cuentas sobre las que se actúa simultáneamente puede ser menor en implementaciones grandes debido a la limitación controlada de los servicios.

RegionConcurrencyType

Obligatorio: no

Puede especificar si el conjunto de pilas debe implementarse en Regiones de AWS secuencialmente o en paralelo configurando el parámetro de implementación simultáneo de la región. Cuando se especifica la concurrencia regional para desplegar pilas Regiones de AWS en varias de ellas en paralelo, los tiempos generales de despliegue pueden ser más rápidos.

- Paralela: las implementaciones de conjuntos de pilas se llevarán a cabo al mismo tiempo, siempre y cuando los errores de implementación de una región no superen una tolerancia de fallos especificada.
- Secuencial: las implementaciones de conjuntos de pilas se realizarán una a la vez, siempre y cuando los errores de implementación de una región no superen una tolerancia de fallos especificada. La implementación secuencial es la selección predeterminada.

ConcurrencyMode

Obligatorio: no

El modo de simultaneidad permite elegir cómo se comporta el nivel de simultaneidad durante las operaciones del conjunto de pilas, ya sea con una tolerancia a los errores estricta o suave. Tolerancia estricta a errores reduce la velocidad de implementación a medida que se producen errores en las operaciones del conjunto de pilas, ya que la simultaneidad disminuye en cada

error. Soft Failure Tolerance  prioriza la velocidad de implementación y, al mismo tiempo, aprovecha las capacidades de seguridad. CloudFormation

- `STRICT_FAILURE_TOLERANCE`: esta opción reduce de forma dinámica el nivel de simultaneidad para garantizar que el número de cuentas con errores nunca supere un valor de tolerancia a errores en particular. Este es el comportamiento predeterminado.
- `SOFT_FAILURE_TOLERANCE`: esta opción desacopla la tolerancia a errores de la simultaneidad real. Esto permite que las operaciones del conjunto de pilas se ejecuten en el nivel de simultaneidad, independientemente del número de errores.

CallAs

Obligatorio: no

> **Note**
>
> Este parámetro es opcional para el modelo de permisos `SERVICE_MANAGED` y no se usa para el modelo de permisos `SELF_MANAGED`.

Especifica si el usuario actúa en la cuenta de administración de la organización o bien como administrador delegado en una cuenta de miembro.

> **Note**
>
> Si este parámetro se establece en `DELEGATED_ADMIN`, asegúrese de que el rol de IAM de la canalización tenga el permiso `organizations:ListDelegatedAdministrators`. De lo contrario, la acción fallará mientras se ejecuta y arrojará un error similar al siguiente: `Account used is not a delegated administrator`.

- `SELF`: la implementación del conjunto de pilas usará permisos administrados por servicios mientras se inicia sesión en la cuenta de administración.
- `DELEGATED_ADMIN`: la implementación del conjunto de pilas usará permisos administrados por servicios mientras se inicia sesión en una cuenta de administrador delegado.

## Artefactos de entrada

Debe incluir al menos un artefacto de entrada que contenga la plantilla del conjunto de pilas de una acción `CloudFormationStackSet`. Puede incluir más artefactos de entrada para las listas de objetivos, cuentas y parámetros de implementación.

- Número de artefactos: `1 to 3`
- Descripción: puede incluir artefactos para proporcionar:
  - El archivo de plantilla de pila. (consulte el parámetro `TemplatePath`).
  - El archivo de parámetros. (consulte el parámetro `Parameters`).
  - El archivo de cuentas. (consulte el parámetro `DeploymentTargets`).

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Variables de salida

Si configura esta acción, produce variables a las que se puede hacer referencia mediante la configuración de acción de una acción descendente en la canalización. Configure una acción con un espacio de nombres para que esas variables estén disponibles para la configuración de las acciones posteriores.

- StackSetId: El identificador del conjunto de pilas.
- OperationId: El identificador de la operación del conjunto de pilas.

Para obtener más información, consulte Referencia de variables.

## Ejemplo CloudFormationStackSet configuración de la acción

Los siguientes ejemplos muestran la configuración de la CloudFormationStackSet acción.

Ejemplo del modelo de permisos autoadministrados

El siguiente ejemplo muestra una CloudFormationStackSet acción en la que el objetivo de despliegue introducido es un identificador de AWS cuenta.

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"
}
```

Ejemplo del modelo de permisos administrados mediante servicios

El siguiente ejemplo muestra una  CloudFormationStackSet  acción para el modelo de permisos gestionados por el servicio, en el que la opción de implementación automática en AWS las organizaciones está habilitada con la retención de pilas.

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"
}
```

## La acción CloudFormationStackInstances

Esta acción crea nuevas instancias e implementa conjuntos de pilas en instancias específicas. Una instancia de pila es una referencia a una pila en una cuenta de destino dentro de una región. Una instancia de pila puede existir sin pila; por ejemplo, si la creación de la pila no se realiza correctamente, la instancia de pila muestra el motivo del fallo en la creación de la pila. Una instancia de pila está asociada con un solo conjunto de pilas.

Tras la creación inicial de un conjunto de pilas, puede añadir nuevas instancias de pilas mediante `CloudFormationStackInstances`. Los valores de los parámetros de la plantilla se pueden anular a nivel de instancia de pila durante las operaciones de creación o actualización de instancias de conjuntos de pilas.

Cada conjunto de pilas tiene una plantilla y un conjunto de parámetros de plantilla. Al actualizar la plantilla o los parámetros de la plantilla, se actualizan para todo el conjunto. A continuación, todos los estados de las instancias se establecen en `OUTDATED` hasta que los cambios se implementen en esa instancia.

Para anular los valores de los parámetros en instancias específicas, por ejemplo, si la plantilla contiene un parámetro para `stage` con un valor de `prod`, puede anular el valor de ese parámetro para que sea `beta` o `gamma`.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Ejemplo de configuración de una acción


## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `CloudFormationStackInstances`
- Versión: `1`

## Parámetros de configuración

StackSetName

Obligatorio: sí

El nombre que se va a asociar con el conjunto de pila. Este nombre debe ser único en la región donde se crea.

Solo puede contener caracteres alfanuméricos y guiones. Debe comenzar por una letra y tener 128 caracteres o menos.

DeploymentTargets

Obligatorio: no

> **Note**
>
> Para el modelo de permisos SERVICE_MANAGED, puede proporcionar el ID raíz de la organización o los ID de la unidad organizativa para los objetivos de implementación. Para el modelo de permisos SELF_MANAGED, solo puede proporcionar cuentas.

> **Note**
>
> Cuando se selecciona este parámetro, también debe seleccionar Regiones.

Una lista de AWS las cuentas o los ID de las unidades organizativas en las que deberían estar las instancias del conjunto de pilas created/updated.

- Cuentas

  Puede proporcionar las cuentas como una lista literal o como una ruta de archivo:

  - Literal: introduzca los parámetros en el formato de sintaxis abreviada `account_ID,account_ID`, como se muestra en el siguiente ejemplo.

    ```
    111111222222,333333444444
    ```

  - Ruta del archivo:  la ubicación del archivo que contiene la lista de AWS cuentas en las que deben estar las instancias del conjunto de pilas created/updated, introducida en el formato`InputArtifactName::AccountsFileName`. Si utilizas la ruta del archivo para especificar  cuentas  o  OrganizationalUnitIds, el formato del archivo debe estar en JSON, como se muestra en el siguiente ejemplo.

    ```
    SourceArtifact::accounts.txt
    ```

    En el siguiente ejemplo se muestra el contenido del archivo para `accounts.txt`.

    ```
    [
        "111111222222"
    ]
    ```

    En el siguiente ejemplo, se muestra el contenido del archivo para incluir más de una cuenta de `accounts.txt`:

```
[
    "111111222222","333333444444"
]
```

- OrganizationalUnitIds:

> **Note**
>
> Este parámetro es opcional para el modelo de permisos SERVICE_MANAGED y no se usa para el modelo de permisos SELF_MANAGED. No lo utilices si seleccionas OrganizationsAutoDeployment.

Las unidades AWS organizativas en las que se actualizarán las instancias de pila asociadas.

Puede proporcionar los ID de las unidades organizativas como una lista literal o una ruta de archivo.

- Literal: introduzca una matriz de cadenas separadas por comas, como se muestra en el siguiente ejemplo.

```
ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
```

- Ruta del archivo:  la ubicación del archivo que contiene una lista de instancias OrganizationalUnitIds en las que se pueden crear o actualizar las instancias del conjunto de pilas. Si utilizas la ruta del archivo para especificar  cuentas  o  OrganizationalUnitIds, el formato del archivo debe estar en JSON, como se muestra en el siguiente ejemplo.

  Introduzca una ruta al archivo con el formato `InputArtifactName::OrganizationalUnitIdsFileName`.

```
SourceArtifact::OU-IDs.txt
```

  En el siguiente ejemplo se muestra el contenido del archivo para `OU-IDs.txt`.

```
[
    "ou-examplerootid111-exampleouid111","ou-examplerootid222-exampleouid222"
]
```

Regiones

Obligatorio: sí

> **Note**
>
> Cuando se selecciona este parámetro, también debe seleccionarlo DeploymentTargets.

Una lista de AWS regiones en las que se crean o actualizan las instancias de conjuntos de pilas. Las regiones se actualizan en el orden en el que se han introducido.

Introduzca una lista de AWS regiones válidas con el formato:`Region1,Region2`, como se muestra en el siguiente ejemplo.

```
us-west-2,us-east-1
```

ParameterOverrides

Obligatorio: no

Una lista de parámetros de conjunto de pila que desea anular en las instancias de pila seleccionadas. Los valores de los parámetros anulados se aplican a todas las instancias de pila de las cuentas y regiones especificadas.

Puede proporcionar parámetros como una lista literal o una ruta de archivo:

- Puede introducir parámetros en el siguiente formato sintáctico abreviado:
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  Para obtener más información acerca de estos tipos de datos, consulte Tipos de datos de parámetros de plantilla.

  El siguiente ejemplo muestra un parámetro denominado `BucketName` con el valor `amzn-s3-demo-source-bucket`.

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ```

  En el siguiente ejemplo, se muestra una entrada con varios parámetros.

```
ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
ParameterKey=Asset1,ParameterValue=true
ParameterKey=Asset2,ParameterValue=true
```

- Puede introducir la ubicación del archivo que contiene una lista de modificaciones de parámetros de plantilla introducidas en el formato `InputArtifactName::ParameterOverridesFileName`, como se muestra en el siguiente ejemplo.

  ```
  SourceArtifact::parameter-overrides.txt
  ```

  En el siguiente ejemplo se muestra el contenido del archivo para `parameter-overrides.txt`.

  ```
  [
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      },
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      }
  ]
  ```

FailureTolerancePercentage

Obligatorio: no

El porcentaje de cuentas por región en las que esta operación de apilamiento puede fallar antes CloudFormation de detener la operación en esa región. Si la operación se detiene en una región, CloudFormation no intente realizarla en las regiones siguientes. Al calcular el número de cuentas en función del porcentaje especificado, CloudFormation redondea a la baja al siguiente número entero.

MaxConcurrentPercentage

Obligatorio: no

El porcentaje máximo de cuentas en las que realizar esta operación de una vez. Al calcular el número de cuentas en función del porcentaje especificado, CloudFormation redondea a la baja al siguiente número entero. Si el redondeo a la baja da como resultado cero, CloudFormation establece el número como uno en su lugar. Si bien especifica el máximo, el número real de cuentas sobre las que se actúa simultáneamente puede ser menor en implementaciones grandes debido a la limitación controlada de los servicios.

RegionConcurrencyType

Obligatorio: no

Puede especificar si el conjunto de pilas debe implementarse en Regiones de AWS secuencialmente o en paralelo configurando el parámetro de implementación simultáneo de la región. Cuando se especifica la concurrencia regional para desplegar pilas Regiones de AWS en varias de ellas en paralelo, los tiempos generales de despliegue pueden ser más rápidos.

- Paralela: las implementaciones de conjuntos de pilas se llevarán a cabo al mismo tiempo, siempre y cuando los errores de implementación de una región no superen una tolerancia de fallos especificada.
- Secuencial: las implementaciones de conjuntos de pilas se realizarán una a la vez, siempre y cuando los errores de implementación de una región no superen una tolerancia de fallos especificada. La implementación secuencial es la selección predeterminada.

ConcurrencyMode

Obligatorio: no

El modo de simultaneidad permite elegir cómo se comporta el nivel de simultaneidad durante las operaciones del conjunto de pilas, ya sea con una tolerancia a los errores estricta o suave. Tolerancia estricta a errores reduce la velocidad de implementación a medida que se producen errores en las operaciones del conjunto de pilas, ya que la simultaneidad disminuye en cada error. Soft Failure Tolerance prioriza la velocidad de implementación y, al mismo tiempo, aprovecha las capacidades de seguridad. CloudFormation

- `STRICT_FAILURE_TOLERANCE`: esta opción reduce de forma dinámica el nivel de simultaneidad para garantizar que el número de cuentas con errores nunca supere un valor de tolerancia a errores en particular. Este es el comportamiento predeterminado.
- `SOFT_FAILURE_TOLERANCE`: esta opción desacopla la tolerancia a errores de la simultaneidad real. Esto permite que las operaciones del conjunto de pilas se ejecuten en el nivel de simultaneidad, independientemente del número de errores.

CallAs

Obligatorio: no

> **Note**
>
> Este parámetro es opcional para el modelo de permisos `SERVICE_MANAGED` y no se usa para el modelo de permisos `SELF_MANAGED`.

Especifica si el usuario actúa en la cuenta de administración de la organización o bien como administrador delegado en una cuenta de miembro.

> **Note**
>
> Si este parámetro se establece en `DELEGATED_ADMIN`, asegúrese de que el rol de IAM de la canalización tenga el permiso `organizations:ListDelegatedAdministrators`. De lo contrario, la acción fallará mientras se ejecuta y arrojará un error similar al siguiente: `Account used is not a delegated administrator`.

- `SELF`: la implementación del conjunto de pilas usará permisos administrados por servicios mientras se inicia sesión en la cuenta de administración.
- `DELEGATED_ADMIN`: la implementación del conjunto de pilas usará permisos administrados por servicios mientras se inicia sesión en una cuenta de administrador delegado.

## Artefactos de entrada

`CloudFormationStackInstances` puede contener artefactos en los que se enumeran los objetivos y los parámetros del implementación.

- Número de artefactos: `0 to 2`
- Descripción: como entrada, la acción de conjunto de pila acepta de forma opcional artefactos para estos fines:
  - Para proporcionar el archivo de parámetros que se va a utilizar. (consulte el parámetro `ParameterOverrides`).

- Para proporcionar el archivo de cuentas de destino que se va a utilizar. (consulte el parámetro `DeploymentTargets`).

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Variables de salida

Cuando se configura, esta acción produce variables a las que se puede hacer referencia mediante la configuración de acción de una acción descendente en la canalización. Configure una acción con un espacio de nombres para que esas variables estén disponibles para la configuración de las acciones posteriores.

- StackSetId: El identificador del conjunto de pilas.
- OperationId: El identificador de la operación del conjunto de pilas.

Para obtener más información, consulte Referencia de variables.

## Ejemplo de configuración de una acción

Los siguientes ejemplos muestran la configuración de la CloudFormationStackInstances acción.

Ejemplo del modelo de permisos autoadministrados

El siguiente ejemplo muestra una CloudFormationStackInstances acción en la que el objetivo de despliegue introducido es un Cuenta de AWS identificador`111111222222`.

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"
}
```

Ejemplo del modelo de permisos administrados mediante servicios

El siguiente ejemplo muestra una  CloudFormationStackInstances  acción para el modelo de permisos administrados por el servicio en el que el objetivo de implementación es el identificador de una unidad organizativa de la AWS organización. `ou-1111-1example`

YAML

```
Name: Instances
```

```
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormationStackInstances
  Version: '1'
RunOrder: 2
Configuration:
  DeploymentTargets: ou-1111-1example
  Regions: us-east-1
  StackSetName: my-stackset
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
Region: eu-central-1
```

JSON

```
{
    "Name": "Instances",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackInstances",
        "Version": "1"
    },
    "RunOrder": 2,
    "Configuration": {
        "DeploymentTargets": "ou-1111-1example",
        "Regions": "us-east-1",
        "StackSetName": "my-stackset"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "eu-central-1"
}
```

## Permisos del rol de servicio: acción de `CloudFormationStackSet`

Para CloudFormation StackSets las acciones, se requieren los siguientes permisos mínimos.

Para la acción `CloudFormationStackSet`, añada lo siguiente a su instrucción de política:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackSet",
        "cloudformation:UpdateStackSet",
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:DescribeStackSet",
        "cloudformation:ListStackInstances"
    ],
    "Resource": "resource_ARN"
},
```

## Permisos del rol de servicio: acción `CloudFormationStackInstances`

Para la acción `CloudFormationStackInstances`, añada lo siguiente a su instrucción de política:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation"
    ],
    "Resource": "resource_ARN"
},
```

## Modelos de permisos para operaciones de conjunto de pilas

Dado que CloudFormation StackSets realiza operaciones en varias cuentas, debes definir los permisos necesarios en esas cuentas antes de poder crear el conjunto de pilas. Puede definir los permisos mediante permisos autoadministrados o permisos administrados por servicios.

Con los permisos autogestionados, se crean las dos funciones de IAM necesarias StackSets : una función de administrador, como la de la cuenta AWSCloudFormationStackSetAdministrationRole en

la que se define el conjunto de pilas, y una función de ejecución, como la de cada una de las cuentas AWSCloudFormationStackSetExecutionRole en las que se implementan las instancias del conjunto de pilas. Con este modelo de permisos, StackSets puedes realizar el despliegue en cualquier AWS cuenta en la que el usuario tenga permisos para crear un rol de IAM. Para obtener más información, consulte Concesión de permisos autoadministrados en la Guía del usuario de AWS CloudFormation .

> **Note**
>
> Dado que CloudFormation StackSets realiza operaciones en varias cuentas, debe definir los permisos necesarios en esas cuentas antes de poder crear el conjunto de pilas.

Con los permisos administrados por el servicio, puedes implementar instancias de pila en las cuentas administradas por AWS las organizaciones. Con este modelo de permisos, no tienes que crear las funciones de IAM necesarias, ya que las StackSets crea en tu nombre. Con este modelo, también puede habilitar implementaciones automáticas en cuentas que se añaden a la organización en el futuro. Consulte Habilitar el acceso confiable en las AWS organizaciones en la Guía del AWS CloudFormation usuario.

## Tipos de datos de parámetros de plantilla

Los parámetros de plantilla utilizados en las operaciones del conjunto de pilas incluyen los siguientes tipos de datos. Para obtener más información, consulte DescribeStackSet.

ParameterKey

- Descripción: la clave asociada con el parámetro. Si no especificas una clave y un valor para un parámetro en particular, AWS CloudFormation usa el valor predeterminado que se especifica en la plantilla.
- Ejemplo:

```
"ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket"
```

ParameterValue

- Descripción: el valor de entrada asociado con el parámetro.
- Ejemplo:

```
"ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket"
```

UsePreviousValue

- Durante una actualización de la pila, utilice el valor de parámetro existente que la pila está utilizando para una clave de parámetro determinada. Si especifica `true`, no especifique un valor de parámetro.
- Ejemplo:

```
"ParameterKey=Asset1,UsePreviousValue=true"
```

Cada conjunto de pilas tiene una plantilla y un conjunto de parámetros de plantilla. Al actualizar la plantilla o los parámetros de la plantilla, se actualizan para todo el conjunto. A continuación, todos los estados de las instancias se establecen en OUTDATED hasta que los cambios se implementen en esa instancia.

Para anular los valores de los parámetros en instancias específicas, por ejemplo, si la plantilla contiene un parámetro para `stage` con un valor de `prod`, puede anular el valor de ese parámetro para que sea `beta` o `gamma`.

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Tipos de parámetros: en este capítulo de referencia de la Guía del AWS CloudFormation usuario se proporcionan más descripciones y ejemplos de los parámetros CloudFormation de la plantilla.
- Prácticas recomendadas: para obtener más información sobre las prácticas recomendadas para implementar conjuntos de pilas, consulte https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-bestpractices.html en la Guía del usuario de AWS CloudFormation .
- AWS CloudFormation Referencia de la API: puedes consultar las siguientes CloudFormation acciones en la referencia de la AWS CloudFormation API para obtener más información sobre los parámetros que se utilizan en las operaciones de conjuntos de pilas:
  - La CreateStackSet acción crea un conjunto de pilas.
  - La UpdateStackSet acción actualiza el conjunto de pilas y las instancias de pila asociadas en las cuentas y regiones especificadas. Incluso si la operación del conjunto de pilas creada al actualizar el conjunto de pilas falla (total o parcialmente, por debajo o por encima de una tolerancia a errores especificada), el conjunto de pilas se actualiza con estos cambios. Las

CreateStackInstances llamadas subsiguientes al conjunto de pilas especificado utilizan el conjunto de pilas actualizado.

- La CreateStackInstances acción crea una instancia de pila para todas las regiones especificadas dentro de todas las cuentas especificadas en un modelo de permisos autogestionados, o dentro de todos los objetivos de despliegue especificados en un modelo de permisos gestionados por servicios. Puede anular los parámetros de las instancias creadas mediante esta acción. Si las instancias ya existen, CreateStackInstances llama UpdateStackInstances con los mismos parámetros de entrada. Cuando utiliza esta acción para crear instancias, no cambia el estado de las demás instancias de la pila.
- La UpdateStackInstances acción actualiza las instancias de pila con la pila configurada para todas las regiones especificadas dentro de todas las cuentas especificadas en un modelo de permisos autogestionados o dentro de todos los objetivos de implementación especificados en un modelo de permisos gestionados por servicios. Puede anular los parámetros de las instancias actualizados mediante esta acción. Cuando utiliza esta acción para actualizar un subconjunto de instancias, no cambia el estado de las demás instancias de la pila.
- La DescribeStackSetOperation acción devuelve la descripción de la operación del conjunto de pilas especificada.
- La DescribeStackSet acción devuelve la descripción del conjunto de pilas especificado.

# AWS CodeBuild referencia de acciones de creación y prueba

Le permite ejecutar compilaciones y pruebas como parte de la canalización. Cuando ejecutas una acción de CodeBuild compilación o prueba, los comandos especificados en la especificación de compilación se ejecutan dentro de un contenedor. CodeBuild Todos los artefactos que se especifican como artefactos de entrada para una CodeBuild acción están disponibles en el contenedor en el que se ejecutan los comandos. CodeBuild pueden proporcionar una acción de compilación o de prueba. Para obtener más información, consulte la Guía del usuario de AWS CodeBuild.

Cuando usas el CodePipeline asistente de la consola para crear un proyecto de compilación, el proyecto de CodeBuild compilación muestra que el proveedor de origen es CodePipeline. Cuando creas un proyecto de compilación en la CodeBuild consola, no puedes especificarlo CodePipeline como proveedor de origen, pero al añadir la acción de compilación a tu canalización, se ajusta el origen en la CodeBuild consola. Para obtener más información, consulta ProjectSource en la AWS CodeBuild Referencia de la API de .

Temas

- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Permisos de rol de servicio: CodeBuild acción
- Declaración de acción (CodeBuild ejemplo)
- Véase también


## Tipo de acción

- Categoría: `Build` o `Test`
- Propietario: `AWS`
- Proveedor: `CodeBuild`
- Versión: `1`

## Parámetros de configuración

ProjectName

Obligatorio: sí

`ProjectName`es el nombre del proyecto de compilación en CodeBuild.

PrimarySource

Obligatorio: condicional

El valor del `PrimarySource` parámetro debe ser el nombre de uno de los artefactos de entrada de la acción. CodeBuild busca el archivo buildspec y ejecuta los comandos buildspec en el directorio que contiene la versión descomprimida de este artefacto.

Este parámetro es necesario si se especifican varios artefactos de entrada para una acción de CodeBuild . Cuando solo hay un artefacto de origen para la acción, el artefacto `PrimarySource` se establece como valor predeterminado de dicho artefacto.

BatchEnabled

Obligatorio: no

El valor booleano del parámetro `BatchEnabled` permite que la acción ejecute varias compilaciones en la misma ejecución de compilación.

Cuando esta opción está habilitada, la opción `CombineArtifacts` está disponible.

Para ver ejemplos de canalizaciones con compilaciones por lotes habilitadas, consulta la integración con y las compilaciones por lotes. CodePipeline CodeBuild

BuildspecOverride

Obligatorio: no

Definición de buildspec insertada o declaración del archivo buildspec que invalida la última definida en el proyecto de compilación, solo para esta compilación. Las especificaciones de buildspec que se han definido en el proyecto no se modifican.

Si se establece este valor, existen las opciones siguientes:

- Una definición de buildspec insertada. Para obtener más información, consulte la referencia de la sintaxis en Sintaxis de buildspec.
- La ruta de un archivo buildspec alternativo relativa al valor de la variable de entorno `CODEBUILD_SRC_DIR` integrada o la ruta de un bucket de S3. El bucket debe encontrarse en la misma Región de AWS que el proyecto de compilación. Especifique el archivo buildspec utilizando su ARN (por ejemplo, `arn:aws:s3:::my-codebuild-sample2/buildspec.yml`). Si este valor no se proporciona o se establece en una cadena vacía, el código fuente debe contener un archivo buildspec en su directorio raíz. Para obtener más información sobre cómo agregar una ruta, consulte Nombre de archivo y ubicación de almacenamiento de buildspec.

> **Note**
>
> Como esta propiedad permite cambiar los comandos de compilación que se ejecutarán en el contenedor, debe tener en cuenta que una entidad principal de IAM con capacidad para llamar a esta API y establecer este parámetro puede anular la configuración predeterminada. Además, le recomendamos que utilice una ubicación para buildspec que sea de confianza, como un archivo en su repositorio de origen o un bucket de Amazon S3.

CombineArtifacts

Obligatorio: no

El valor booleano del parámetro `CombineArtifacts` combina todos los artefactos de compilación de una compilación por lotes en un único archivo de artefactos para la acción de compilación.

Para utilizar esta opción, el parámetro `BatchEnabled` debe estar activado.

EnvironmentVariables

Obligatorio: no

El valor de este parámetro se utiliza para establecer variables de entorno para la acción de CodeBuild de la canalización. El valor del parámetro `EnvironmentVariables` toma la forma de una matriz JSON de objetos de variables de entorno. Consulte el parámetro de ejemplo en Declaración de acción (CodeBuild ejemplo).

Cada objeto tiene tres partes, todas las cuales son cadenas:

- `name`: el nombre o la clave de la variable de entorno.
- `value`: el valor de la variable de entorno. Al usar el `SECRETS_MANAGER` tipo `PARAMETER_STORE` o, este valor debe ser el nombre de un parámetro que ya haya almacenado en el almacén de parámetros de AWS Systems Manager o un secreto que ya haya almacenado en AWS Secrets Manager, respectivamente.

  > **Note**
  >
  > No recomendamos encarecidamente el uso de variables de entorno para almacenar valores confidenciales, especialmente AWS credenciales. Cuando usas la CodeBuild consola o la AWS CLI, las variables de entorno se muestran en texto sin formato. Para valores confidenciales, se recomienda utilizar el tipo `SECRETS_MANAGER`en su lugar.

- `type`: (opcional) el tipo de la variable de entorno. Los valores válidos son `PARAMETER_STORE`, `SECRETS_MANAGER` o `PLAINTEXT`. Si no se especifica, toma el valor predeterminado `PLAINTEXT`.

> **Note**
>
> Al introducir y `type` para la `name` configuración de las variables de entorno, especialmente si la variable de entorno contiene la sintaxis de las variables de

CodePipeline salida, no supere el límite de 1000 caracteres para el campo de valores de la configuración. `value` Cuando se supera este límite, se devuelve un error de validación.

Para obtener más información, consulte la referencia EnvironmentVariable de la AWS CodeBuild API. Para ver un ejemplo de CodeBuild acción con una variable de entorno que se resuelve en el nombre de la GitHub sucursal, consultaEjemplo: usa una BranchName variable con variables de CodeBuild entorno.

ServiceRoleArnOverride

Obligatorio: no

Un ARN de rol de IAM que se CodeBuild utiliza como rol de servicio para las compilaciones activadas por esta acción y anula el rol de servicio predeterminado configurado en el proyecto. CodeBuild Cuando está ausente, el CodeBuild proyecto usa su rol de servicio predeterminado.

El rol especificado debe tener una política de confianza que permita al `codebuild.amazonaws.com` servicio asumirlo. Tanto la identidad utilizada para crear o actualizar la canalización como la función de servicio de canalización deben tener `iam:PassRole` permiso para la función especificada. Para ver una política de ejemplo, consulte Permisos de rol de servicio: CodeBuild acción.

Como práctica recomendada de seguridad, cuando un CodeBuild proyecto se comparte entre varias canalizaciones, recomendamos especificar un rol específico para cada canalización a fin de seguir el principio de mínimo privilegio. En el caso de las canalizaciones que utilizan acciones de CodeConnections origen, consulta un Añade CodeBuild GitClone permisos para las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com ejemplo de cómo determinar el alcance de los permisos de conexión del rol a un repositorio específico.

> **Note**
>
> El rol debe estar en la misma AWS cuenta que el CodeBuild proyecto.

## Artefactos de entrada

- Número de artefactos: `1 to 5`

- Descripción: CodeBuild busca el archivo buildspec y ejecuta los comandos buildspec desde el directorio del artefacto fuente principal. Cuando se especifica una sola fuente de entrada o cuando se especifica más de una fuente de entrada para la CodeBuild acción, el artefacto único, o el artefacto principal en el caso de varias fuentes de entrada, se debe configurar mediante el parámetro de configuración de acción en. `PrimarySource` CodePipeline

  Cada artefacto de entrada se extrae en su propio directorio, cuyas ubicaciones se almacenan en variables de entorno. El directorio del artefacto de origen principal está disponible con `$CODEBUILD_SRC_DIR`. Los directorios del resto de artefactos de entrada están disponibles con `$CODEBUILD_SRC_DIR_yourInputArtifactName`.

  > **Note**
  >
  > El artefacto configurado en tu CodeBuild proyecto se convierte en el artefacto de entrada utilizado por la CodeBuild acción de tu canalización.

## Artefactos de salida

- Número de artefactos: `0 to 5`
- Descripción: Se pueden usar para que los artefactos definidos en el archivo CodeBuild buildspec estén disponibles para las acciones posteriores del proceso. Cuando solo se define un artefacto de salida, este artefacto se puede definir directamente en la sección `artifacts` del archivo buildspec. Cuando se especifica más de un artefacto de salida, todos los artefactos a los que se hace referencia deben definirse como artefactos secundarios en el archivo buildspec. Los nombres de los artefactos de salida CodePipeline deben coincidir con los identificadores de artefactos del archivo buildspec.

  > **Note**
  >
  > El artefacto configurado en tu CodeBuild proyecto se convierte en el artefacto de CodePipeline entrada de tu acción en canalización.

  Si se selecciona el parámetro `CombineArtifacts` para compilaciones por lotes, la ubicación del artefacto de salida contiene los artefactos combinados de varias compilaciones que se ejecutaron en la misma ejecución.

## Variables de salida

Esta acción producirá como variables todas las variables de entorno que se exportaron como parte de la compilación. Para obtener más información sobre cómo exportar variables de entorno, consulta la EnvironmentVariable Guía de AWS CodeBuild API.

Para obtener más información sobre el uso de variables de CodeBuild entorno en CodePipeline, consulte los ejemplos deCodeBuild variables de salida de la acción. Para obtener una lista de las variables de entorno que puede usar CodeBuild, consulte Variables de entorno en entornos de construcción en la Guía del AWS CodeBuild usuario.

## Permisos de rol de servicio: CodeBuild acción

Para obtener CodeBuild asistencia, añada lo siguiente a su declaración de política:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "codebuild:BatchGetBuilds",
                "codebuild:StartBuild",
                "codebuild:BatchGetBuildBatches",
                "codebuild:StartBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:*:111122223333:project/[[ProjectName]]"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Si usas el `ServiceRoleArnOverride` parámetro, agrega el siguiente `iam:PassRole` permiso CodePipeline para poder transferir el rol a CodeBuild:

```
{
```

```
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::111122223333:role/your-codebuild-override-role",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": "codebuild.amazonaws.com"
        }
    }
}
```

## Declaración de acción (CodeBuild ejemplo)

### YAML

```
Name: Build
Actions:
  - Name: PackageExport
    ActionTypeId:
      Category: Build
      Owner: AWS
      Provider: CodeBuild
      Version: '1'
    RunOrder: 1
    Configuration:
      BatchEnabled: 'true'
      CombineArtifacts: 'true'
      ProjectName: my-build-project
      PrimarySource: MyApplicationSource1
      EnvironmentVariables:
 '[{"name":"TEST_VARIABLE","value":"TEST_VALUE","type":"PLAINTEXT"},
{"name":"ParamStoreTest","value":"PARAMETER_NAME","type":"PARAMETER_STORE"}]'
      ServiceRoleArnOverride: 'arn:aws:iam::111122223333:role/my-scoped-codebuild-
role'
    OutputArtifacts:
      - Name: MyPipeline-BuildArtifact
    InputArtifacts:
      - Name: MyApplicationSource1
      - Name: MyApplicationSource2
```

### JSON

```
{
```

```
    "Name": "Build",
    "Actions": [
        {
            "Name": "PackageExport",
            "ActionTypeId": {
                "Category": "Build",
                "Owner": "AWS",
                "Provider": "CodeBuild",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "BatchEnabled": "true",
                "CombineArtifacts": "true",
                "ProjectName": "my-build-project",
                "PrimarySource": "MyApplicationSource1",
                "EnvironmentVariables": "[{\"name\":\"TEST_VARIABLE\",\"value\":
\"TEST_VALUE\",\"type\":\"PLAINTEXT\"},{\"name\":\"ParamStoreTest\",\"value\":
\"PARAMETER_NAME\",\"type\":\"PARAMETER_STORE\"}]",
                "ServiceRoleArnOverride": "arn:aws:iam::111122223333:role/my-scoped-
codebuild-role"
            },
            "OutputArtifacts": [
                {
                    "Name": "MyPipeline-BuildArtifact"
                }
            ],
            "InputArtifacts": [
                {
                    "Name": "MyApplicationSource1"
                },
                {
                    "Name": "MyApplicationSource2"
                }
            ]
        }
    ]
}
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- AWS CodeBuild Guía del usuario: para ver un ejemplo de canalización con una CodeBuild acción, consulta Usar CodePipeline with CodeBuild para probar código y ejecutar compilaciones. Para ver ejemplos de proyectos con varios CodeBuild artefactos de entrada y salida, consulta el Ejemplo de CodePipeline integración con CodeBuild múltiples fuentes de entrada y artefactos de salida y el Ejemplo de múltiples fuentes de entrada y artefactos de salida.
- Tutorial: Crea una canalización que compile y pruebe tu aplicación para Android con AWS Device Farm— Este tutorial proporciona un archivo de especificaciones de compilación y una aplicación de ejemplo para crear una canalización con una GitHub fuente que compila y prueba una aplicación de Android con y. CodeBuild AWS Device Farm
- Referencia sobre las especificaciones de compilación para CodeBuild : en este tema de referencia se proporcionan definiciones y ejemplos para entender CodeBuild los archivos buildspec. Para obtener una lista de las variables de entorno que puedes usar CodeBuild, consulta la sección Variables de entorno en entornos de compilación en la Guía del AWS CodeBuild usuario.

# AWS CodePipeline invocar referencia a la acción

Las acciones de CodePipeline invocación se utilizan para simplificar la activación de las ejecuciones de canalizaciones posteriores y la transferencia de variables de canalización y revisiones de código fuente entre canalizaciones.

> **Note**
>
> Esta acción solo se admite para canalizaciones de tipo V2.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Permisos de política de roles de servicio para la acción de CodePipeline invocación
- Declaración de acciones
- Véase también

## Tipo de acción

- Categoría: `Invoke`
- Propietario: `AWS`
- Proveedor: `CodePipeline`
- Versión: `1`

## Parámetros de configuración

PipelineName

Obligatorio: sí

El nombre de la canalización que, al ejecutarse, iniciará la canalización de destino actual. Debe haber creado previamente la canalización de invocación. La acción iniciará la canalización `s3-pipeline-test` (de destino) cuando la canalización (de invocación) denominada `my-s3-pipeline` inicie una ejecución.

SourceRevisions

Obligatorio: no

Las revisiones de origen que desea que utilice la canalización de destino cuando la inicie la canalización que la invoca. Por ejemplo, una acción de origen de S3 proporciona variables de salida como el ID de versión de S3 y la clave de objeto. Puede especificar un valor de revisión que se utilizará cuando se invoque la canalización.

Para la CLI, especifique las revisiones de origen como una cadena JSON serializada. Para obtener más información sobre el uso de las anulaciones de las revisiones de código fuente, consulta SourceRevisionOverride la guía de API. CodePipeline

La asignación utiliza el formato de cadena que se muestra en el siguiente ejemplo:

```
[{"actionName":"Source","revisionType":"S3_OBJECT_VERSION_ID","revision
Value":"zq8mjNEXAMPLE"}]
```

Variables

Obligatorio: no

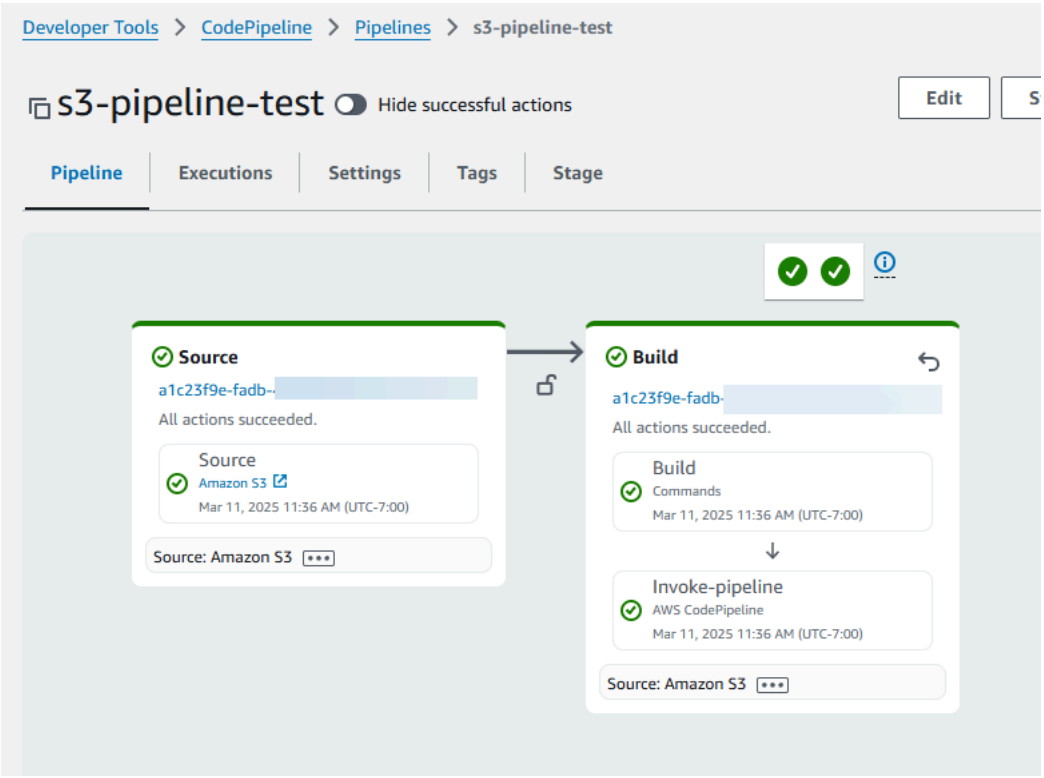
Los nombres y valores de las variables que desea que admita la acción.

Para la CLI, las variables se especifican como una cadena JSON serializada. Para obtener más información sobre el uso de variables de canalización, consulta PipelineVariable la Guía de CodePipeline API.

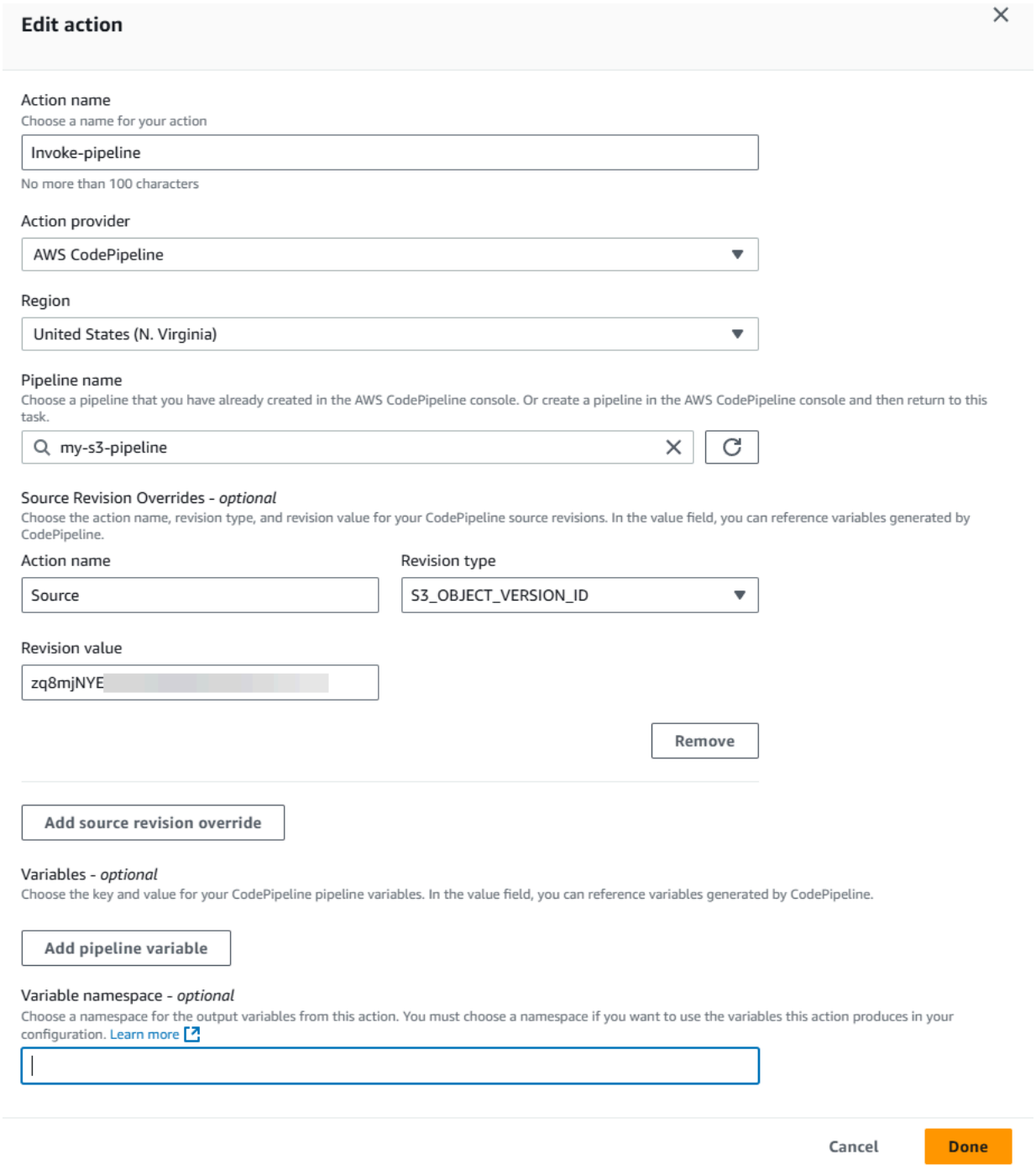
La asignación utiliza el formato de cadena que se muestra en el siguiente ejemplo:

```
[{"name":"VAR1","value":"VALUE1"}]
```

En la siguiente imagen, se muestra un ejemplo de la acción que se agregó a una canalización en la consola.




En la siguiente imagen, se muestra un ejemplo de la página Editar de la acción. En el siguiente ejemplo, la canalización con el nombre `s3-pipeline-test` tiene una acción de invocación de canalización configurada como se muestra para la consola. La acción iniciará la canalización `s3-pipeline-test` cuando la canalización denominada `my-s3-pipeline` inicie una ejecución. En el ejemplo, se muestra la anulación de revisión de origen para la anulación de origen S3_OBJECT_VERSION_ID con un valor de revisión especificado de `zq8mjNYEexample`.




# Artefactos de entrada

- Número de artefactos: `0`
- Descripción: los artefactos de entrada no se aplican a este tipo de acción.

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Permisos de política de roles de servicio para la acción de CodePipeline invocación

Cuando CodePipeline se ejecuta la acción, la política de roles de CodePipeline servicio requiere el `codepipeline:StartPipelineExecution` permiso, con un alcance adecuado hasta el ARN del recurso de canalización para mantener el acceso con los mínimos privilegios.

```
{
           "Sid": "StatementForPipelineInvokeAction",
           "Effect": "Allow",
           "Action": "codepipeline:StartPipelineExecution",
           "Resource": [
               "arn:aws:codepipeline:{{region}}:{{AccountId}}:{{pipelineName}}"
           ]
       }
```

## Declaración de acciones

YAML

```
name: Invoke-pipeline
actionTypeId:
  category: Invoke
  owner: AWS
  provider: CodePipeline
  version: '1'
runOrder: 2
configuration:
  PipelineName: my-s3-pipeline
  SourceRevisions:
 '[{"actionName":"Source","revisionType":"S3_OBJECT_VERSION_ID","revision
Value":"zq8mjNEXAMPLE"}]'
  Variables: '[{"name":"VAR1","value":"VALUE1"}]'
```

JSON

```
{
    "name": "Invoke-pipeline",
    "actionTypeId": {
        "category": "Invoke",
        "owner": "AWS",
        "provider": "CodePipeline",
        "version": "1"
    },
    "runOrder": 2,
    "configuration": {
        "PipelineName": "my-s3-pipeline",
        "SourceRevisions": "[{\"actionName\":\"Source\",\"revisionType\":
\"S3_OBJECT_VERSION_ID\",\"revisionValue\":\"zq8mjNEXAMPLE\"}]",
        "Variables": "[{\"name\":\"VAR1\",\"value\":\"VALUE1\"}]"
    }
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Iniciar una canalización con una anulación de revisión de código fuente— En esta sección se describe cómo iniciar una canalización con revisiones del código fuente de forma manual o mediante el transformador de entrada de EventBridge eventos.

# CodeCommit referencia de acción de origen

Inicia la canalización cuando se realiza una nueva confirmación en el CodeCommit repositorio y la rama configurados.

Si usas la consola para crear o editar la canalización, CodePipeline crea una EventBridge regla que inicie la canalización cuando se produzca un cambio en el repositorio.

> **Note**
>
> En el caso de Amazon ECR, Amazon S3 o CodeCommit las fuentes, también puedes crear una anulación de fuente utilizando la entrada de transformación de entrada EventBridge para usar la entrada `revisionValue` en el evento de canalización, donde `revisionValue` se

deriva de la variable de evento de origen de tu clave de objeto, confirmación o ID de imagen. Para obtener más información, consulte el paso opcional para la entrada de transformación de entrada, que se incluye en los procedimientos de Acciones y recursos de origen de Amazon ECR EventBridge, Conexión a acciones de origen de Amazon S3 con un origen habilitado para eventos o CodeCommit acciones de origen y EventBridge.

Debe haber creado ya un CodeCommit repositorio antes de conectar la canalización mediante una CodeCommit acción.

Después de detectar un cambio de código, tiene las siguientes opciones para pasar el código a acciones posteriores:

- Predeterminado: configura la acción de CodeCommit origen para generar un archivo ZIP con una copia superficial de la confirmación.
- Clonación completa: configura la acción de origen para generar una referencia de URL de Git al repositorio para acciones posteriores.

  Actualmente, la referencia URL de Git solo la pueden usar CodeBuild las acciones posteriores para clonar el repositorio y los metadatos de Git asociados. Si se intenta pasar una referencia de URL de Git a una URL que no se ha realizado ninguna CodeBuild acción, se produce un error.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Permisos de rol de servicio: CodeCommit acción
- Ejemplo de configuración de una acción
- Véase también


## Tipo de acción

- Categoría: `Source`

- Propietario: `AWS`
- Proveedor: `CodeCommit`
- Versión: `1`

## Parámetros de configuración

RepositoryName

Obligatorio: sí

El nombre del repositorio en el que se van a detectar los cambios de origen.

BranchName

Obligatorio: sí

El nombre de la ramificación donde se van a detectar los cambios de origen.

PollForSourceChanges

Obligatorio: no

`PollForSourceChanges`controla si CodePipeline sondea el CodeCommit repositorio en busca de cambios en el código fuente. En su lugar, le recomendamos que utilice CloudWatch los eventos para detectar los cambios en la fuente. Para obtener más información sobre la configuración de CloudWatch eventos, consulte Migrar los canales de sondeo (CodeCommit fuente) (CLI) oMigrar los canales de sondeo (CodeCommit fuente) (CloudFormation plantilla).

> **Important**
>
> Si quieres configurar una regla de CloudWatch eventos, debes configurarla `PollForSourceChanges` para `false` evitar que se dupliquen las ejecuciones de canalización.

Los valores válidos para este parámetro son:

- `true`: Si se establece, CodePipeline sondea tu repositorio para ver si hay cambios en el código fuente.

> **Note**
>
> Si lo omites`PollForSourceChanges`, de CodePipeline forma predeterminada sondea tu repositorio para ver si hay cambios en el código fuente. Este comportamiento es el mismo que si se incluye `PollForSourceChanges` y se establece en `true`.

- `false`: Si se establece, CodePipeline no sondea tu repositorio para ver si hay cambios en el código fuente. Usa esta configuración si quieres configurar una regla de CloudWatch eventos para detectar los cambios en el origen.

OutputArtifactFormat

Obligatorio: no

El formato del artefacto de salida. Los valores pueden ser `CODEBUILD_CLONE_REF` o `CODE_ZIP`. Si no se especifica, el valor predeterminado es `CODE_ZIP`.

> **Important**
>
> La opción `CODEBUILD_CLONE_REF` solo se puede utilizar en acciones de CodeBuild posteriores.
> Si elige esta opción, debe agregar el `codecommit:GitPull` permiso a su rol de CodeBuild servicio como se muestra enAgregue CodeBuild GitClone permisos para las acciones CodeCommit de origen. También debe añadir el `codecommit:GetRepository` permiso a su rol de CodePipeline servicio, tal y como se muestra enAgrega permisos al rol de CodePipeline servicio. Para ver un tutorial que muestra cómo utilizar la opción Clonación completa, consulte Tutorial: Usa la clonación completa con una fuente de CodeCommit canalización.

## Artefactos de entrada

- Número de artefactos: `0`
- Descripción: los artefactos de entrada no se aplican a este tipo de acción.

## Artefactos de salida

- Número de artefactos: 1

- Descripción: el artefacto de salida de esta acción es un archivo ZIP que contiene el contenido del repositorio configurado y la ramificación en la confirmación especificada como la revisión de origen para la ejecución de la canalización. Los artefactos generados desde el repositorio son los artefactos de salida de la CodeCommit acción. El identificador de confirmación del código fuente se muestra CodePipeline como la revisión de origen para la ejecución de la canalización activada.

## Variables de salida

Cuando se configura, esta acción produce variables a las que se puede hacer referencia mediante la configuración de acción de una acción descendente en la canalización. Esta acción produce variables que se pueden ver como variables de salida, incluso si la acción no tiene un espacio de nombres. Configure una acción con un espacio de nombres para que esas variables estén disponibles para la configuración de las acciones posteriores.

Para obtener más información, consulte Referencia de variables.

CommitId

El ID de CodeCommit confirmación que activó la ejecución de la canalización. Los ID de confirmación son el volumen de dispositivo raíz completo de la confirmación.

CommitMessage

El mensaje de descripción, si lo hay, asociado a la confirmación que desencadenó la ejecución de la canalización.

RepositoryName

El nombre del CodeCommit repositorio en el que se realizó la confirmación que activó la canalización.

BranchName

El nombre de la rama del CodeCommit repositorio en el que se realizó el cambio de fuente.

AuthorDate

Fecha en la que se creó la confirmación, en formato de marca temporal.

CommitterDate

Fecha en la que se ha confirmado la confirmación, en formato de marca temporal.

## Permisos de rol de servicio: CodeCommit acción

Cuando CodePipeline se ejecuta la acción, la política de funciones de CodePipeline servicio requiere los siguientes permisos, con un alcance adecuado según el ARN del recurso canalizado, a fin de mantener el acceso con los mínimos privilegios. Por ejemplo, agregue lo siguiente a su declaración de política:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecommit:CancelUploadArchive",
                "codecommit:GetBranch",
                "codecommit:GetCommit",
                "codecommit:GetRepository",
                "codecommit:GetUploadArchiveStatus",
                "codecommit:UploadArchive"
            ],
            "Resource": [
                "arn:aws:codecommit:*:111122223333:[[codecommitRepostories]]"
            ]
        }
    ]
}
```

## Ejemplo de configuración de una acción

### Ejemplo de formato de artefacto de salida predeterminado

YAML

```
name: Source
actionTypeId:
  category: Source
  owner: AWS
```

```
  provider: CodeCommit
  version: '1'
runOrder: 1
configuration:
  BranchName: main
  PollForSourceChanges: 'false'
  RepositoryName: MyWebsite
outputArtifacts:
  - name: Artifact_MyWebsiteStack
inputArtifacts: []
region: us-west-2
namespace: SourceVariables
```

JSON

```
{
    "name": "Source",
    "actionTypeId": {
        "category": "Source",
        "owner": "AWS",
        "provider": "CodeCommit",
        "version": "1"
    },
    "runOrder": 1,
    "configuration": {
        "BranchName": "main",
        "PollForSourceChanges": "false",
        "RepositoryName": "MyWebsite"
    },
    "outputArtifacts": [
        {
            "name": "Artifact_MyWebsiteStack"
        }
    ],
    "inputArtifacts": [],
    "region": "us-west-2",
    "namespace": "SourceVariables"
}
```

## Ejemplo de formato de artefacto de salida de clonación

### 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"
}
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Tutorial: Crear una canalización simple (CodeCommit repositorio)— Este tutorial proporciona un AppSpec archivo de ejemplo y un grupo de CodeDeploy aplicaciones y despliegues de ejemplo. Utilice este tutorial para crear una canalización con una CodeCommit fuente que se implemente en las instancias de Amazon EC2.

# Referencia de la acción de implementación de AWS CodeDeploy

Utiliza una AWS CodeDeploy acción para implementar el código de la aplicación en su flota de implementación. Su flota de implementación puede estar compuesta por instancias de Amazon EC2, instancias en las instalaciones o ambas.

> **Note**
>
> En este tema de referencia se describe la acción de CodeDeploy despliegue en los CodePipeline casos en los que la plataforma de despliegue es Amazon EC2. Para obtener información de referencia sobre Amazon Elastic Container Service y las acciones de CodeDeploy blue/green despliegue en CodePipeline ellas, consulteReferencia de acciones de implementación de Amazon Elastic Container Service y CodeDeploy azul-verde.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Permisos de rol de servicio: AWS CodeDeploy acción

- Declaración de acciones
- Véase también

## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `CodeDeploy`
- Versión: `1`

## Parámetros de configuración

ApplicationName

Obligatorio: sí

El nombre de la aplicación en la que creó CodeDeploy.

DeploymentGroupName

Obligatorio: sí

El grupo de despliegue en el que creó CodeDeploy.

## Artefactos de entrada

- Número de artefactos: `1`
- Descripción: el AppSpec archivo que se CodeDeploy utiliza para determinar:
  - Qué instalar en sus instancias desde la revisión de la aplicación en Amazon S3 o GitHub.
  - Los enlaces de eventos del ciclo de vida que se deben ejecutar en respuesta a los eventos del ciclo de vida de la implementación.

  Para obtener más información sobre el AppSpec archivo, consulte la referencia del CodeDeploy AppSpec archivo.

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Permisos de rol de servicio: AWS CodeDeploy acción

Para obtener AWS CodeDeploy asistencia, añada lo siguiente a su declaración de política:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:CreateDeployment",
                "codedeploy:GetApplication",
                "codedeploy:GetDeployment",
                "codedeploy:RegisterApplicationRevision",
                "codedeploy:ListDeployments",
                "codedeploy:ListDeploymentGroups",
                "codedeploy:GetDeploymentGroup"
            ],
            "Resource": [
                "arn:aws:codedeploy:*:111122223333:application:
[[codedeployApplications]]",
                "arn:aws:codedeploy:*:111122223333:deploymentgroup:
[[codedeployApplications]]/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:GetDeploymentConfig"
            ],
            "Resource": [
                "arn:aws:codedeploy:*:111122223333:deploymentconfig:
[[deploymentConfigs]]"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "codedeploy:ListDeploymentConfigs"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

## Declaración de acciones

### 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"
        }
    ]
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Tutorial: Crear una canalización simple (bucket de S3)— Este tutorial explica cómo crear un bucket de código fuente, instancias EC2 y CodeDeploy recursos para implementar una aplicación de muestra. Luego, crea su canalización con una acción de CodeDeploy implementación que implementa el código mantenido en su bucket de S3 en su instancia de Amazon EC2.
- Tutorial: Crear una canalización simple (CodeCommit repositorio)— En este tutorial, se explica cómo crear el repositorio de CodeCommit origen, las instancias de EC2 y CodeDeploy los recursos para implementar una aplicación de muestra. A continuación, crea su canalización con una acción de CodeDeploy despliegue que despliega el código de su CodeCommit repositorio en su instancia de Amazon EC2.
- CodeDeploy AppSpec Referencia de archivos: este capítulo de referencia de la Guía del AWS CodeDeploy usuario proporciona información de referencia y ejemplos de CodeDeploy AppSpec archivos.

# CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas

Las acciones de origen para las conexiones son compatibles con. AWS CodeConnections AWS CodeConnections permite crear y administrar conexiones entre AWS recursos y repositorios de terceros, como GitHub. Inicia una canalización cuando se realiza una nueva confirmación en un repositorio de código fuente de terceros. La acción de origen recupera los cambios de código cuando se ejecuta manualmente una canalización o cuando se envía un evento webhook desde el proveedor de origen.

Puede configurar acciones en la canalización para usar una configuración de Git que permita iniciar la canalización con desencadenadores. Para configurar los desencadenadores de canalización para filtrar con ellos, consulte más información en Agregación de desencadenadores con tipos de eventos de solicitud de inserción o extracción de código.

> **Note**
>
> Esta función no está disponible en las regiones Asia Pacífico (Hong Kong), Asia Pacífico (Hyderabad), Asia Pacífico (Yakarta), Asia Pacífico (Melbourne), Asia Pacífico (Osaka), África (Ciudad del Cabo), Oriente Medio (Baréin), Oriente Medio (Emiratos Árabes Unidos), Europa (España), Europa (Zúrich), Israel (Tel Aviv) o AWS GovCloud (). US-West Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline. Para ver consideraciones sobre esta acción en la región de Europa (Milán), consulte la nota que aparece en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

Connections puede asociar tus AWS recursos a los siguientes repositorios de terceros:

- Bitbucket Cloud (mediante la opción de proveedor de Bitbucket en la CodePipeline consola o el `Bitbucket` proveedor en la CLI)

  > **Note**
  >
  > Puede crear conexiones a un repositorio de Bitbucket Cloud. Los tipos de proveedores de Bitbucket instalados, como Bitbucket Server, no son compatibles.

- > **Note**
  >
  > Si utiliza un espacio de trabajo de Bitbucket, debe tener acceso de administrador para crear la conexión.

- GitHub y GitHub Enterprise Cloud (a través de la opción de proveedor GitHub (mediante GitHub aplicación) en la CodePipeline consola o el `GitHub` proveedor en la CLI)

  > **Note**
  >
  > Si tu repositorio está en una GitHub organización, debes ser el propietario de la organización para crear la conexión. Si está utilizando un repositorio que no está en una organización, debe ser el propietario del repositorio.

- GitHub Enterprise Server (mediante la opción de proveedor de GitHub Enterprise Server en la CodePipeline consola o el `GitHub Enterprise Server` proveedor en la CLI)
- GitLab.com (mediante la opción GitLabde proveedor en la CodePipeline consola o el `GitLab` proveedor en la CLI)

  > **Note**
  >
  > Puede crear conexiones a un repositorio en el que tenga el rol de propietario y GitLab, a continuación, la conexión se puede utilizar con el repositorio con recursos como CodePipeline: En el caso de los repositorios en grupos, no es necesario que sea el propietario del grupo.

- Self-managed instalación para GitLab (Enterprise Edition o Community Edition) (mediante la opción de proveedor GitLab autogestionado en la CodePipeline consola o el `GitLabSelfManaged` proveedor en la CLI)

> **Note**
>
> Cada conexión es compatible con todos los repositorios que tenga con ese proveedor. Solo tiene que crear una conexión nueva para cada tipo de proveedor.

Las conexiones permiten que su canalización detecte los cambios de origen a través de la aplicación de instalación del proveedor externo. Por ejemplo, los webhooks se utilizan para suscribirse a tipos de GitHub eventos y se pueden instalar en una organización, un repositorio o una GitHub aplicación. Tu conexión instala un webhook de repositorio en tu GitHub aplicación que se suscribe a GitHub eventos de tipo push.

Después de detectar un cambio de código, tiene las siguientes opciones para pasar el código a acciones posteriores:

- Predeterminado: al igual que otras acciones de CodePipeline código fuente existentes, `CodeStarSourceConnection` puedes generar un archivo ZIP con una copia superficial de tu confirmación.
- Full cone (Clonación completa): `CodeStarSourceConnection` también se puede configurar para generar una referencia URL al repositorio para acciones posteriores.

  Actualmente, la referencia a la URL de Git solo la pueden usar las CodeBuild acciones posteriores para clonar el repositorio y los metadatos de Git asociados. Si se intenta pasar una referencia a una URL de Git a una falta de CodeBuild acción, se produce un error.

CodePipeline te pide que añadas la aplicación de instalación de AWS Connector a tu cuenta de terceros al crear una conexión. Debe haber creado ya su cuenta y repositorio de proveedor externo para poder conectarse a través de la acción `CodeStarSourceConnection`.

> **Note**
>
> Para crear o asociar una política a su rol con los permisos necesarios para usar las conexiones de AWS CodeStar , consulte Referencia de permisos de conexiones. En función de cuándo se creó su función de CodePipeline servicio, es posible que necesite actualizar sus permisos para admitir AWS CodeStar las conexiones. Para obtener instrucciones, consulte Agrega permisos al rol de CodePipeline servicio.

> **Note**
>
> Para utilizar las conexiones en Europa (Milán) Región de AWS, debe:
>
> 1. Instalar una Region-specific aplicación
> 2. Habilitar la región

Esta Region-specific aplicación admite conexiones en la región de Europa (Milán). Se publica en el sitio del proveedor externo y es independiente de la aplicación existente que admite conexiones para otras regiones. Al instalar esta aplicación, autoriza a los proveedores externos a compartir sus datos con el servicio únicamente para esta región, y puede revocar los permisos en cualquier momento desinstalando la aplicación.
El servicio no procesará ni almacenará sus datos a menos que habilite la región. Al habilitar esta región, otorga a nuestro servicio permisos para procesar y almacenar sus datos.
Aunque la región no esté habilitada, los proveedores externos pueden compartir tus datos con nuestro servicio si la Region-specific aplicación sigue instalada, así que asegúrate de desinstalarla una vez que desactives la región. Para obtener más información, consulte Habilitar una región.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Permisos del rol de servicio: AWS CodeConnections acción
- Declaración de acciones
- Instalación de la aplicación de instalación y creación de una conexión
- Véase también


## Tipo de acción

- Categoría: `Source`
- Propietario: `AWS`
- Proveedor: `CodeStarSourceConnection`
- Versión: `1`

# Parámetros de configuración

ConnectionArn

Obligatorio: sí

El ARN de conexión que se ha configurado y autenticado para el proveedor de código fuente.

FullRepositoryId

Obligatorio: sí

El propietario y el nombre del repositorio en el que se van a detectar los cambios de código fuente.

Ejemplo: `some-user/my-repo`

> **Important**
>
> Debes mantener las mayúsculas y minúsculas correctas para el FullRepositoryIdvalor. Por ejemplo, si tu nombre de usuario es `some-user` y el nombre del repositorio es`My-Repo`, el valor recomendado FullRepositoryIdes`some-user/My-Repo`.

BranchName

Obligatorio: sí

El nombre de la ramificación donde se van a detectar los cambios de origen.

OutputArtifactFormat

Obligatorio: no

Especifica el formato del artefacto de salida. Puede ser `CODEBUILD_CLONE_REF` o `CODE_ZIP`. Si no se especifica, el valor predeterminado es `CODE_ZIP`.

> **Important**
>
> La opción `CODEBUILD_CLONE_REF` solo se puede utilizar en acciones de CodeBuild posteriores.
> Si eliges esta opción, tendrás que actualizar los permisos de tu función de servicio de CodeBuild proyectos, tal y como se muestra en Añade CodeBuild GitClone permisos para

las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com la siguiente. Para ver un tutorial que muestra cómo utilizar la opción Clonación completa, consulte Tutorial: Utilice un clon completo con una fuente de GitHub canalización.

DetectChanges

Obligatorio: no

Controla automáticamente el inicio de su canalización cuando se realiza una nueva confirmación en el repositorio y la rama configurados. Si no se especifica, el valor predeterminado es `true` y el campo no se muestra de forma predeterminada. Los valores válidos para este parámetro son:

- `true`: inicia CodePipeline automáticamente tu canalización con las nuevas confirmaciones.
- `false`: CodePipeline no inicia tu canalización con nuevas confirmaciones.

## Artefactos de entrada

- Número de artefactos: `0`
- Descripción: los artefactos de entrada no se aplican a este tipo de acción.

## Artefactos de salida

- Número de artefactos: `1`
- Descripción: los artefactos generados desde el repositorio son los artefactos de salida de la acción `CodeStarSourceConnection`. El ID de confirmación del código fuente se muestra CodePipeline como la revisión de origen de la ejecución de la canalización activada. Puede configurar el artefacto de salida de esta acción en:
  - Un archivo ZIP que incluye el contenido del repositorio configurado y la ramificación en la confirmación especificada como la revisión de código fuente para la ejecución de la canalización.
  - Un archivo JSON que contiene una referencia URL al repositorio para que las acciones posteriores puedan ejecutar comandos de Git directamente.

> **Important**
>
> Esta opción solo la pueden utilizar las acciones CodeBuild posteriores.

Si elige esta opción, tendrá que actualizar los permisos de su función de servicio de CodeBuild proyectos, tal y como se muestra en Solución de problemas CodePipeline la siguiente. Para ver un tutorial que muestra cómo utilizar la opción Clonación completa, consulte Tutorial: Utilice un clon completo con una fuente de GitHub canalización.

## Variables de salida

Cuando se configura, esta acción produce variables a las que se puede hacer referencia mediante la configuración de acción de una acción descendente en la canalización. Esta acción produce variables que se pueden ver como variables de salida, incluso si la acción no tiene un espacio de nombres. Configure una acción con un espacio de nombres para que esas variables estén disponibles para la configuración de las acciones posteriores.

Para obtener más información, consulte Referencia de variables.

AuthorDate

Fecha en la que se creó la confirmación, en formato de marca temporal.

BranchName

El nombre de la ramificación del repositorio donde se realizó el cambio de origen.

CommitId

El ID de confirmación de que desencadenó la ejecución de la canalización.

CommitMessage

El mensaje de descripción, si lo hay, asociado a la confirmación que desencadenó la ejecución de la canalización.

ConnectionArn

El ARN de conexión que se ha configurado y autenticado para el proveedor de código fuente.

FullRepositoryName

El nombre del repositorio de donde se realizó la confirmación que activó la canalización.

# Permisos del rol de servicio: AWS CodeConnections acción

Para AWS CodeConnections ello, se requiere el siguiente permiso para crear canalizaciones con una fuente que utilice una conexión, como Bitbucket Cloud.

```
{
    "Effect": "Allow",
    "Action": [
        "codeconnections:UseConnection",
        "codestar-connections:UseConnection"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Ambas acciones son obligatorias. Las operaciones se invocan mediante una llamada`codestar-connections:UseConnection`, mientras que las operaciones se invocan fuera de la llamada`codeconnections:UseConnection`. La concesión de ambas acciones garantiza que la función funcione para ambas rutas de código, independientemente de si el ARN del recurso de la conexión utiliza el prefijo `codeconnections` o el prefijo de `codestar-connections` servicio.

Para obtener más información sobre los permisos de IAM para las conexiones, consulte Referencia de permisos de conexiones.

## Declaración de acciones

En el siguiente ejemplo, el artefacto de salida se establece en el formato ZIP predeterminado de `CODE_ZIP` para la conexión con ARN `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"
        }
    ]
```

```
},
```

## Instalación de la aplicación de instalación y creación de una conexión

La primera vez que utilices la consola para añadir una nueva conexión a un repositorio de terceros, debes autorizar el CodePipeline acceso a tus repositorios. Elegirá y creará una aplicación de instalación que le ayude a conectarse a la cuenta en la que ha creado su repositorio de código de terceros.

Cuando utilice la plantilla AWS CLI o una CloudFormation plantilla, debe proporcionar el ARN de conexión de una conexión que ya haya pasado por el protocolo de enlace de instalación. De lo contrario, la canalización no se activará.

> **Note**
>
> Para una acción de origen `CodeStarSourceConnection`, no es necesario configurar un webhook ni utilizar el sondeo de forma predeterminada. La acción de conexiones gestiona automáticamente la detección de cambio de origen.

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- AWS::CodeStarConnections: :Conexión: la referencia CloudFormation de plantilla del recurso AWS CodeStar Connections proporciona parámetros y ejemplos de conexiones en plantillas. CloudFormation
- AWS CodeStarReferencia de la API de AWS CodeStar conexiones: la referencia de la API de conexiones proporciona información de referencia para las acciones de conexión disponibles.
- Para ver los pasos para crear una canalización con acciones de origen compatibles con las conexiones, consulte lo siguiente:
  - Para Bitbucket Cloud, use la opción Bitbucket en la consola o la acción `CodestarSourceConnection` en la CLI. Consulte Conexiones de Bitbucket Cloud.
  - Para GitHub GitHub Enterprise Cloud, utilice la opción de GitHubproveedor en la consola o la `CodestarSourceConnection` acción en la CLI. Consulte GitHub conexiones.

  - Para GitHub Enterprise Server, utilice la opción de proveedor de GitHub Enterprise Server en la consola o la `CodestarSourceConnection` acción en la CLI. Consulte GitHub Conexiones de Enterprise Server.
  - Para GitLab.com ello, utilice la opción de GitLabproveedor en la consola o la `CodestarSourceConnection` acción con el `GitLab` proveedor en la CLI. Consulte GitLab.com conexiones.
- Para ver un tutorial de introducción que crea una canalización con una fuente y una CodeBuild acción de Bitbucket, consulta Cómo empezar con las conexiones.
- Para ver un tutorial que te muestra cómo conectarte a un GitHub repositorio y cómo usar la opción de clonación completa con una CodeBuild acción posterior, consulta. Tutorial: Utilice un clon completo con una fuente de GitHub canalización

# Referencia de la acción de Comandos

La acción de Comandos permite ejecutar comandos del intérprete de comandos en una instancia de computación virtual. Al ejecutar la acción, los comandos especificados en la configuración de la acción se ejecutan en un contenedor independiente. Todos los artefactos que se especifican como artefactos de entrada para una CodeBuild acción están disponibles en el contenedor en el que se ejecutan los comandos. Esta acción permite especificar comandos sin tener que crear primero un CodeBuild proyecto. Para obtener más información consulte ActionDeclaration y OutputArtifact en la Referencia de la API de AWS CodePipeline .

> **Important**
>
> Esta acción usa la CodeBuild computación CodePipeline administrada para ejecutar comandos en un entorno de compilación. Si ejecuta la acción de Comandos, se le cobrarán cargos por separado en AWS CodeBuild.

> **Note**
>
> La acción de implementación de comandos solo está disponible para canalizaciones de tipo V2.

Temas

- Consideraciones sobre la acción de Comandos
- Permisos para las políticas de roles de servicio
- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de entorno
- Permisos del rol de servicio: acción de Comandos
- Declaración de acciones (ejemplo)
- Véase también


## Consideraciones sobre la acción de Comandos

Las siguientes consideraciones se aplican a la acción de Comandos.

- La acción de comando usa CodeBuild recursos similares a los de la CodeBuild acción, al tiempo que permite los comandos del entorno shell en una instancia de procesamiento virtual sin necesidad de asociar o crear un proyecto de compilación.

  > **Note**
  >
  > Si ejecuta la acción de Comandos, se le cobrarán cargos por separado en AWS CodeBuild.

- Como la acción Comandos CodePipeline usa CodeBuild recursos, las compilaciones que ejecute la acción se atribuirán a los límites de compilación de tu cuenta en CodeBuild. Las compilaciones ejecutadas mediante la acción de Comandos se tendrán en cuenta para los límites de compilación simultánea configurados para esa cuenta.
- El tiempo de espera para las compilaciones con la acción Comandos es de 55 minutos, según las CodeBuild compilaciones.
- La instancia de procesamiento usa un entorno de compilación aislado en CodeBuild.

> **Note**
>
> Debido a que el entorno de compilación aislado se usa a nivel de cuenta, es posible que una instancia se reutilice para otra ejecución de canalización.

- Se admiten todos los formatos, excepto los formatos multilínea. Debe utilizar el formato de una sola línea al introducir comandos.
- La acción de comandos es compatible con las acciones en varias cuentas. Para agregar una acción de comandos en varias cuentas, agregue `actionRoleArn` en su cuenta de destino en la declaración de la acción.
- Para esta acción, CodePipeline asumirá la función de servicio de canalización y la usará para permitir el acceso a los recursos en tiempo de ejecución. Se recomienda configurar el rol de servicio para que los permisos se limiten al nivel de acción.
- Los permisos agregados a la función CodePipeline de servicio se detallan enAgrega permisos al rol de CodePipeline servicio.
- El permiso necesario para ver los registros en la consola se detalla en Permisos necesarios para ver los registros de procesamiento en la consola.
- A diferencia de otras acciones CodePipeline, en la configuración de la acción no se configuran los campos; los campos de configuración de la acción se establecen fuera de la configuración de la acción.

## Permisos para las políticas de roles de servicio

Cuando CodePipeline ejecuta la acción, CodePipeline crea un grupo de registros con el nombre de la canalización de la siguiente manera. Esto permite reducir los permisos para registrar los recursos mediante el nombre de la canalización.

```
/aws/codepipeline/MyPipelineName
```

Si utiliza un rol de servicio existente, para utilizar la acción de Comandos tendrá que agregar los siguientes permisos para el rol de servicio.

- registros: CreateLogGroup
- registros: CreateLogStream
- registros: PutLogEvents

En la declaración de las políticas de roles de servicio, limite los permisos al nivel de la canalización como se muestra en el siguiente ejemplo.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": [
        "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME",
        "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
   ]
}
```

Para ver los registros en la consola mediante la página del cuadro de diálogo de detalles de la acción, se debe agregar el permiso para ver los registros al rol de la consola. Para obtener más información, consulte el ejemplo de política de permisos para consolas en Permisos necesarios para ver los registros de procesamiento en la consola.

## Tipo de acción

- Categoría: `Compute`
- Propietario: `AWS`
- Proveedor: `Commands`
- Versión: `1`

## Parámetros de configuración

Comandos

Obligatorio: sí

Puede proporcionar comandos del intérprete de comandos para que se ejecute la acción `Commands`. En la consola, los comandos se introducen en líneas separadas. En la CLI, los comandos se introducen como cadenas independientes.

> **Note**
>
> Multi-line los formatos no son compatibles y generarán un mensaje de error. Single-line se debe usar el formato para introducir comandos en el campo Comandos.

> **Important**
>
> Los ComputeType valores EnvironmentType y coinciden con los de CodeBuild. Admitimos un subconjunto de los tipos disponibles. Para obtener más información, consulte Tipos de computación de entorno de compilación.

EnvironmentType

Obligatorio: no

La imagen del sistema operativo del entorno de compilación que admite la acción de comandos. Estos son los valores válidos para los entornos de compilación:

- LINUX_CONTAINER
- WINDOWS_SERVER_2022_CONTAINER

La selección de EnvironmentType permitirá entonces incluir en el ComputeType campo el tipo de cálculo para ese sistema operativo. Para obtener más información sobre los tipos de CodeBuild cómputos disponibles para esta acción, consulta la referencia sobre los modos y tipos de cómputos del entorno de compilación en la Guía del CodeBuild usuario.

> **Note**
>
> Si no se especifica, la computación utilizará de forma predeterminada lo siguiente para el entorno de compilación:
>
> - Tipo de computación: BUILD_GENERAL1_SMALL
> - Tipo de entorno: LINUX_CONTAINER

ComputeType

Obligatorio: no

En función de la selección EnvironmentType, se puede proporcionar el tipo de procesamiento. A continuación, se muestran los valores disponibles para la computación, pero tenga en cuenta que pueden variar según el sistema operativo.

- BUILD_GENERAL1_SMALL
- BUILD_GENERAL1_MEDIUM
- BUILD_GENERAL1_LARGE

> **Important**
>
> Algunos tipos de computación no son compatibles con ciertos tipos de entornos. Por ejemplo, WINDOWS_SERVER_2022_CONTAINER no es compatible con BUILD_GENERAL1_SMALL. El uso de combinaciones incompatibles hará que la acción falle y generará un error de tiempo de ejecución.

outputVariables

Obligatorio: no

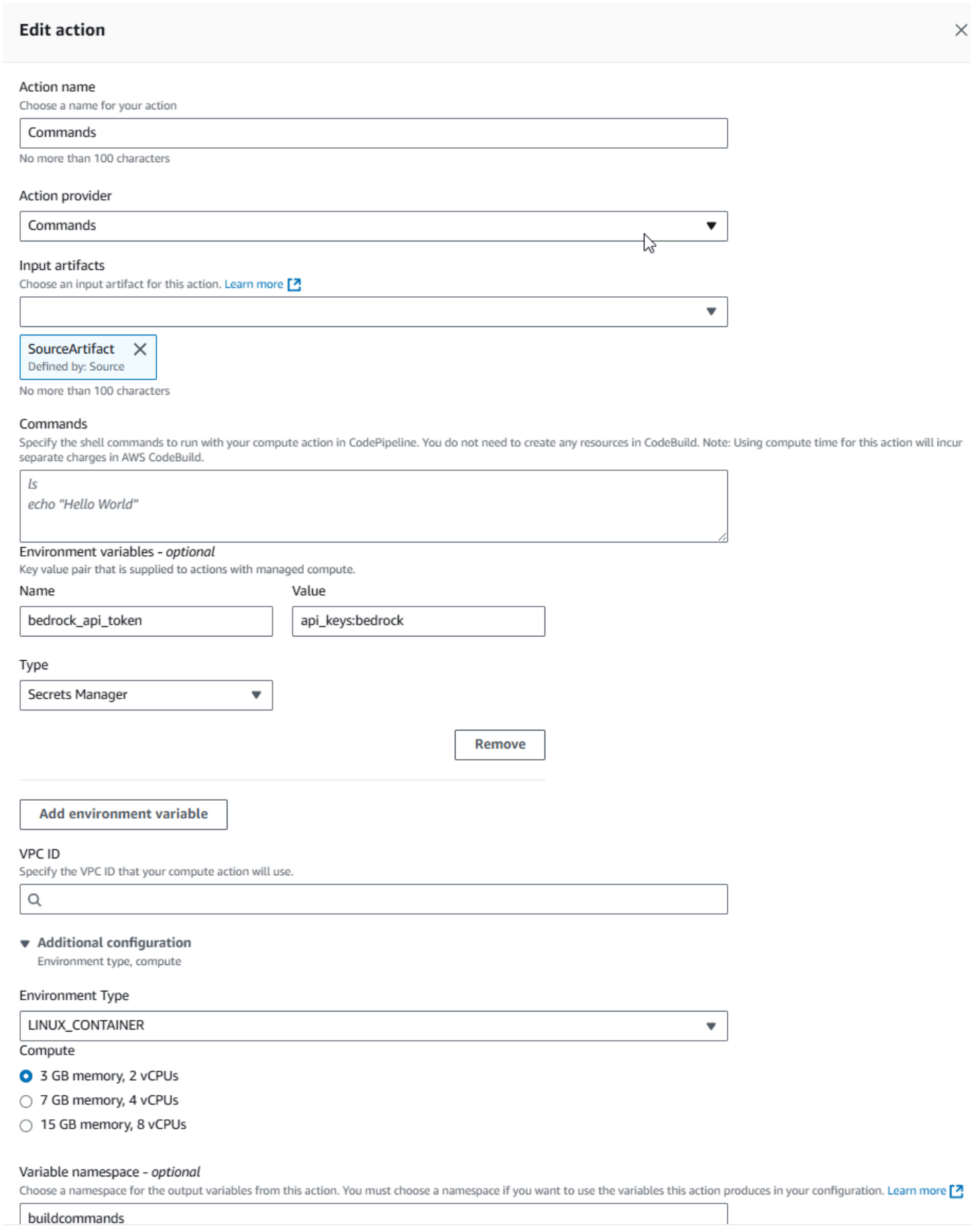
Especifique los nombres de las variables del entorno que desee exportar. Para obtener una referencia de las variables de CodeBuild entorno, consulte las variables de entorno en entornos de construcción en la Guía del CodeBuild usuario.

Archivos

Obligatorio: no

Puede proporcionar los archivos que desee exportar como artefactos de salida para la acción.

El formato admitido para los archivos es el mismo que para los patrones de CodeBuild archivos. Por ejemplo, introduzca `**/` para todos los archivos. Para obtener más información, consulte la referencia sobre las especificaciones de compilación CodeBuild en la Guía del CodeBuild usuario.


Edit action
Action name
Choose a name for your action
Commands
No more than 100 characters
Action provider
Commands
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
Defined by: Source
No more than 100 characters
Commands
Specify the shell commands to run with your compute action in CodePipeline. You do not need to create any resources in CodeBuild. Note: Using compute time for this action will incur separate charges in AWS CodeBuild.
ls
echo "Hello World"
Environment variables - optional
Key value pair that is supplied to actions with managed compute.
Name
Value
bedrock_api_token
api_keys:bedrock
Type
Secrets Manager
Remove
Add environment variable
VPC ID
Specify the VPC ID that your compute action will use.
Additional configuration
Environment type, compute
Environment Type
LINUX_CONTAINER
Compute
3 GB memory, 2 vCPUs
7 GB memory, 4 vCPUs
15 GB memory, 8 vCPUs
Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. Learn more
buildcommands
Cancel
Done

VpcId

Obligatorio: no

El ID de VPC de sus recursos.

Subredes

Obligatorio: no

Las subredes de VPC. Este campo es necesario cuando los comandos deben conectarse a los recursos de una VPC.

SecurityGroupIds

Obligatorio: no

Los grupos de seguridad de la VPC. Este campo es necesario cuando los comandos deben conectarse a los recursos de una VPC.

A continuación, se muestra un ejemplo de JSON de la acción en el que se muestran los campos de configuración para el entorno y el tipo de procesamiento, junto con un ejemplo de variable de entorno.

```
{
           "name": "Commands1",
           "actionTypeId": {
             "category": "Compute",
             "owner": "AWS",
             "provider": "Commands",
             "version": "1"
           },
           "inputArtifacts": [
             {
               "name": "SourceArtifact"
             }
           ],
           "commands": [
             "ls",
             "echo hello",
             "echo $BEDROCK_TOKEN",
           ],
           "configuration": {
```

```
            "EnvironmentType": "LINUX_CONTAINER",
            "ComputeType": "BUILD_GENERAL1_MEDIUM"
          },
          "environmentVariables": [
            {
              "name": "BEDROCK_TOKEN",
              "value": "apiTokens:bedrockToken",
              "type": "SECRETS_MANAGER"
            }
          ],
          "runOrder": 1
        }
```

## Artefactos de entrada

- Número de artefactos: `1 to 10`

## Artefactos de salida

- Número de artefactos: `0 to 1`

## Variables de entorno

Key

La clave de un par de variables de entorno de clave-valor, como `BEDROCK_TOKEN`.

Valor

El valor del par clave-valor, como `apiTokens:bedrockToken`. El valor se puede parametrizar con variables de salida de acciones o variables de canalización.

Al usar el `SECRETS_MANAGER` tipo, este valor debe ser el nombre de un secreto que ya hayas almacenado en AWS Secrets Manager.

Tipo

Especifica el tipo de uso del valor de la variable de entorno. El valor puede ser `PLAINTEXT` o `SECRETS_MANAGER`. Si el valor es `SECRETS_MANAGER`, indique la referencia de Secrets en el valor `EnvironmentVariable`. Si no se especifica, toma el valor predeterminado `PLAINTEXT`.

> **Note**
>
> No recomendamos encarecidamente el uso de variables de entorno de texto simple para almacenar valores confidenciales, especialmente AWS credenciales. Al usar la CodeBuild consola o AWS CLI, las variables de entorno en texto plano se muestran en texto sin formato. Para valores confidenciales, se recomienda utilizar el tipo `SECRETS_MANAGER`en su lugar.

> **Note**
>
> Al introducir y `type` para la `name` configuración de las variables de entorno, especialmente si la variable de entorno contiene la sintaxis de las variables de CodePipeline salida, no supere el límite de 1000 caracteres para el campo de valores de la configuración. `value` Cuando se supera este límite, se devuelve un error de validación.

Para ver un ejemplo de declaración de acción donde se muestre una variable de entorno, consulte Parámetros de configuración.

> **Note**
>
> - Este tipo `SECRETS_MANAGER` solo se admite para la acción de comandos.
> - Los secretos a los que se hace referencia en la acción Comandos se redactarán en los registros de compilación de forma similar a. CodeBuild Sin embargo, los usuarios de la canalización que tengan acceso de edición en la canalización podrán seguir accediendo a estos valores secretos modificando los comandos.
> - Para utilizarla SecretsManager, debes añadir los siguientes permisos a tu función de servicio de canalización:
>
>   ```
>   {
>               "Effect": "Allow",
>               "Action": [
>                   "secretsmanager:GetSecretValue"
>               ],
>               "Resource": [
>                   "SECRET_ARN"
>               ]
>   ```

```
        }
```

## Permisos del rol de servicio: acción de Comandos

Para admitir la acción de Comandos, agregue lo siguiente a la declaración de la política:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:iam::*:role/Service*",
                "arn:aws:iam::*:role/Service*"
            ]
        }
    ]
}
```

## Declaración de acciones (ejemplo)

YAML

```
name: Commands_action
actionTypeId:
  category: Compute
  owner: AWS
  provider: Commands
  version: '1'
runOrder: 1
configuration: {}
commands:
```

```
- ls
- echo hello
- 'echo pipeline Execution Id is #{codepipeline.PipelineExecutionId}'
outputArtifacts:
- name: BuildArtifact
  files:
  - **/
inputArtifacts:
- name: SourceArtifact
outputVariables:
- AWS_DEFAULT_REGION
region: us-east-1
namespace: compute
```

JSON

```
{
    "name": "Commands_action",
    "actionTypeId": {
        "category": "Compute",
        "owner": "AWS",
        "provider": "Commands",
        "version": "1"
    },
    "runOrder": 1,
    "configuration": {},
    "commands": [
        "ls",
        "echo hello",
        "echo pipeline Execution Id is #{codepipeline.PipelineExecutionId}"
    ],
    "outputArtifacts": [
        {
            "name": "BuildArtifact",
            "files": [
                "**/"
            ]
        }
    ],
    "inputArtifacts": [
        {
            "name": "SourceArtifact"
        }
```

```
    ],
    "outputVariables": [
        "AWS_DEFAULT_REGION"
    ],
    "region": "us-east-1",
    "namespace": "compute"
}
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Tutorial: creación de una canalización que ejecute comandos mediante computación (tipo V2): este tutorial proporciona un ejemplo de canalización con la acción de Comandos.

# AWS Device Farm referencia de acción de prueba

En proceso, puedes configurar una acción de prueba que se utilice AWS Device Farm para ejecutar y probar tu aplicación en los dispositivos. Device Farm utiliza grupos de pruebas de dispositivos y marcos de pruebas para probar aplicaciones en dispositivos específicos. Para obtener información sobre los tipos de marcos de prueba que admite la acción Device Farm, consulte Cómo trabajar con tipos de prueba en AWS Device Farm.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Permisos de rol de servicio: AWS Device Farm acción
- Declaración de acciones
- Véase también


## Tipo de acción

- Categoría: `Test`

- Propietario: `AWS`
- Proveedor: `DeviceFarm`
- Versión: `1`

## Parámetros de configuración

AppType

Obligatorio: sí

El sistema operativo y el tipo de aplicación que está probando. A continuación, se muestra una lista de valores válidos para TestType:

- `iOS`
- `Android`
- `Web`

ProjectId

Obligatorio: sí

El ID de proyecto de Device Farm.

Para encontrar el ID del proyecto, en la consola de Device Farm, elija su proyecto. En el navegador, copie la URL de su nuevo proyecto. La dirección URL contiene el ID del proyecto. El ID del proyecto es el valor que aparece en la URL después de `projects/`. En el siguiente ejemplo, el ID de proceso es `eec4905f-98f8-40aa-9afc-4c1cfexample`.

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

App

Obligatorio: sí

El nombre y la ubicación del archivo de la aplicación en el artefacto de entrada. Por ejemplo: `s3-ios-test-1.ipa`

TestSpec

Condicional: sí

La ubicación del archivo de definición de especificaciones de prueba en el artefacto de entrada. Es necesario para la prueba en modo personalizado.

DevicePoolArn

Obligatorio: sí

El ARN del grupo de dispositivos de Device Farm.

Para obtener los ARN de los grupos de dispositivos disponibles para el proyecto, incluido el ARN de los principales dispositivos, utilice la AWS CLI para introducir el siguiente comando:

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

TestType

Obligatorio: sí

Especifica el marco de pruebas compatible para la prueba. A continuación, se muestra una lista de valores válidos para `TestType`:

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG
- APPIUM_NODE
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- BUILTIN_FUZZ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

> **Note**
>
> La acción de CodePipeline:`WEB_PERFORMANCE_PROFILE`, y no admite los siguientes tipos de prueba. `REMOTE_ACCESS_RECORD` `REMOTE_ACCESS_REPLAY`

Para obtener información sobre los tipos de pruebas de Device Farm, consulte Trabajar con tipos de pruebas en AWS Device Farm.

RadioBluetoothEnabled

Obligatorio: no

Un valor booleano que indica si se habilita Bluetooth al comienzo de la prueba.

RecordAppPerformanceData

Obligatorio: no

Un valor booleano que indica si se deben registrar los datos de rendimiento del dispositivo, como el rendimiento de la CPU, FPS y la memoria, durante la prueba.

RecordVideo

Obligatorio: no

Valor booleano que indica si se debe grabar vídeo durante la prueba.

RadioWifiEnabled

Obligatorio: no

Un valor booleano que indica si se debe habilitar Wi-Fi al principio de la prueba.

RadioNfcEnabled

Obligatorio: no

Un valor booleano que indica si se habilita NFC al comienzo de la prueba.

RadioGpsEnabled

Obligatorio: no

Un valor booleano que indica si se habilita GPS al comienzo de la prueba.

Test

Obligatorio: no

El nombre y la ruta del archivo de definición de la prueba en la ubicación de origen. La ruta es relativa a la raíz del artefacto de entrada de la prueba.

FuzzEventCount

Obligatorio: no

El número de eventos de interfaz de usuario para la prueba de difusión que se va a realizar, entre 1 y 10 000.

FuzzEventThrottle

Obligatorio: no

El número de milisegundos que debe esperar la prueba de difusión antes de realizar el siguiente evento de interfaz de usuario, entre 1 y 1000.

FuzzRandomizerSeed

Obligatorio: no

Semilla para la prueba de difusión para aleatorizar los eventos de interfaz de usuario. Utilizar el mismo número en las subsiguientes pruebas de difusión genera secuencias de eventos idénticas.

CustomHostMachineArtifacts

Obligatorio: no

La ubicación de la máquina host en la que se almacenarán los artefactos personalizados.

CustomDeviceArtifacts

Obligatorio: no

La ubicación en el dispositivo donde se almacenarán los artefactos personalizados.

UnmeteredDevicesOnly

Obligatorio: no

Un valor booleano que indica si solo se utilizan los dispositivos no medidos al ejecutar las pruebas en este paso.

JobTimeoutMinutes

Obligatorio: no

Número de minutos que se ejecutará una prueba por dispositivo antes de que se agote el tiempo de espera.

Latitud

Obligatorio: no

La latitud del dispositivo expresada en grados del sistema de coordenadas geográficas.

Longitud

Obligatorio: no

La longitud del dispositivo expresada en grados del sistema de coordenadas geográficas.

## Artefactos de entrada

- Número de artefactos: `1`
- Descripción: el conjunto de artefactos que se pondrán a disposición de la acción de prueba. Device Farm busca la aplicación integrada y las definiciones de prueba que se van a utilizar.

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Permisos de rol de servicio: AWS Device Farm acción

Cuando CodePipeline se ejecuta la acción, la política de funciones de CodePipeline servicio requiere los siguientes permisos, con un alcance adecuado según el ARN del recurso canalizado, a fin de mantener el acceso con los mínimos privilegios. Por ejemplo, agregue lo siguiente a su declaración de política:

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

## Declaración de acciones

### 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"
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Trabajar con tipos de pruebas en Device Farm: en este capítulo de referencia de la Guía para desarrolladores de Device Farm se proporciona más información sobre los marcos de pruebas de aplicaciones web, iOS y Android compatibles con Device Farm.
- Acciones en Device Farm: las llamadas a la API y los parámetros de la guía Device Farm API Reference pueden ayudarle a trabajar con proyectos de Device Farm.
- Tutorial: Crea una canalización que compile y pruebe tu aplicación para Android con AWS Device Farm— Este tutorial proporciona un archivo de especificaciones de compilación y una

aplicación de ejemplo para crear una canalización con una GitHub fuente que compile y pruebe una aplicación de Android con Device Farm. CodeBuild

- Tutorial: Crea una canalización que pruebe tu aplicación para iOS con AWS Device Farm: en este tutorial se proporciona un ejemplo de aplicación para crear una canalización con una fuente de Amazon S3 que pruebe una aplicación iOS creada con Device Farm.

# Referencia de la acción de implementación de Elastic Beanstalk

Elastic Beanstalk es una plataforma interna AWS que se usa para implementar y escalar aplicaciones web. Se utiliza una acción de Elastic Beanstalk para implementar el código de aplicación en el entorno de implementación.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Permisos del rol de servicio: acción de implementación ElasticBeanstalk
- Declaración de acciones
- Véase también


## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `ElasticBeanstalk`
- Versión: `1`

## Parámetros de configuración

ApplicationName

Obligatorio: sí

Es el nombre de la aplicación que creó en Elastic Beanstalk.

EnvironmentName

Obligatorio: sí

Es el nombre del entorno que creó en Elastic Beanstalk. Un entorno es un conjunto de AWS recursos que ejecutan una versión de la aplicación. Cada entorno ejecuta solo una versión de la aplicación a la vez; sin embargo, puede ejecutar la misma versión de la aplicación o diferentes versiones de la aplicación en muchos entornos simultáneamente.

## Artefactos de entrada

- Número de artefactos: `1`
- Descripción: son los artefactos de entrada para la acción.

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Permisos del rol de servicio: acción de implementación **`ElasticBeanstalk`**

A continuación se indican los permisos mínimos necesarios en para crear canalizaciones con una acción de implementación de `ElasticBeanstalk`.

```
{
    "Effect": "Allow",
    "Action": [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
        "cloudformation:*",
```

```
        "rds:*",
        "sqs:*",
        "ecs:*"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Debe sustituir los caracteres comodín de la política de recursos por los recursos de la cuenta a la que desee limitar el acceso. Para obtener más información acerca de la creación de un política que concede acceso con privilegios mínimos, consulte https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege.

## Declaración de acciones

YAML

```
Name: Deploy
Actions:
  - Name: Deploy
    ActionTypeId:
      Category: Deploy
      Owner: AWS
      Provider: ElasticBeanstalk
      Version: '1'
    RunOrder: 1
    Configuration:
      ApplicationName: my-application
      EnvironmentName: my-environment
    OutputArtifacts: []
    InputArtifacts:
      - Name: SourceArtifact
    Region: us-west-2
    Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
```

```
    "Actions": [
        {
            "Name": "Deploy",
            "ActionTypeId": {
                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "ElasticBeanstalk",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "ApplicationName": "my-application",
                "EnvironmentName": "my-environment"
            },
            "OutputArtifacts": [],
            "InputArtifacts": [
                {
                    "Name": "SourceArtifact"
                }
            ],
            "Region": "us-west-2",
            "Namespace": "DeployVariables"
        }
    ]
},
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Implementación de una aplicación Flask en Elastic Beanstalk: en este tutorial, se explica la creación de los recursos de aplicación y entorno en Elastic Beanstalk mediante una aplicación Flask de ejemplo. A continuación, puede crear su canalización con una acción de implementación de Elastic Beanstalk que implemente la aplicación desde el repositorio de origen en el entorno de Elastic Beanstalk.

# Referencia de la acción de invocación de `InspectorScan` de Amazon Inspector

Amazon Inspector es un servicio de administración de vulnerabilidades que detecta automáticamente las cargas de trabajo y las analiza de forma continua en busca de vulnerabilidades de software y exposiciones de red no deseadas. La `InspectorScan` acción CodePipeline automatiza la detección y la corrección de las vulnerabilidades de seguridad en su código fuente abierto. Se trata de una acción de computación administrada con capacidades de análisis de seguridad. Puedes usarlo InspectorScan con el código fuente de la aplicación en un repositorio de terceros, como Bitbucket Cloud, GitHub o con imágenes para aplicaciones contenedoras. Su acción analizará los niveles de vulnerabilidad y las alertas que configure e informará al respecto.

> **Important**
>
> Esta acción utiliza la CodeBuild computación CodePipeline gestionada para ejecutar comandos en un entorno de compilación. Si ejecuta la acción, se le cobrarán cargos por separado en AWS CodeBuild.

Temas


- ID de tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Permisos del rol de servicio: acción InspectorScan
- Declaración de acciones
- Véase también


## ID de tipo de acción

- Categoría: `Invoke`
- Propietario: `AWS`
- Proveedor: `InspectorScan`

- Versión: 1

Ejemplo:

```
        {
            "Category": "Invoke",
            "Owner": "AWS",
            "Provider": "InspectorScan",
            "Version": "1"
        },
```

## Parámetros de configuración

InspectorRunMode

(Obligatorio) La cadena que indica el modo de análisis. Los valores válidos son `SourceCodeScan | ECRImageScan`.

ECRRepositoryNombre

El nombre del repositorio de Amazon ECR al que se envió la imagen.

ImageTag

La etiqueta utilizada para la imagen.

Los parámetros de esta acción buscan los niveles de vulnerabilidad que especifique. Los siguientes niveles de umbrales de vulnerabilidad están disponibles:

CriticalThreshold

El número de vulnerabilidades de gravedad crítica encontradas en tu fuente más allá de las cuales no se CodePipeline debería ejecutar la acción.

HighThreshold

El número de vulnerabilidades de alta gravedad encontradas en su fuente más allá de las cuales no CodePipeline debería realizarse la acción.

MediumThreshold

El número de vulnerabilidades de gravedad media que se encuentran en su fuente CodePipeline si no se supera el límite de la acción.

LowThreshold

El número de vulnerabilidades de baja gravedad encontradas en la fuente, más allá de las cuales no CodePipeline debería realizarse la acción.

Action name
Choose a name for your action
Scan
No more than 100 characters

Action provider
AWS InspectorScan

Region
US East (N. Virginia)

Input artifacts
Choose an input artifact for this action. Learn more

SourceArtifact
Defined by: Source
No more than 100 characters

Inspector Run mode
InspectorRunMode: SourceCodeScan or ECRImageScan.

Source Code Scan
SourceCodeScan: Scan the source code. Choose an input artifact for this run mode.

ECR Image Scan
ECRImageScan: Scan the ECR image.

Critical Threshold - *optional*
Threshold value for critical severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

High Threshold - *optional*
Threshold value for high severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

Medium Threshold - *optional*
Threshold value for medium severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

Low Threshold - *optional*
Threshold value for low severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

## Artefactos de entrada

- Número de artefactos: 1
- Descripción: el código fuente en el que se buscarán vulnerabilidades. Si el análisis es para un repositorio de ECR, este artefacto de entrada no es necesario.

## Artefactos de salida

- Número de artefactos: 1
- Descripción: detalles de la vulnerabilidad del origen en forma de archivo de lista de materiales de software (SBOM).

## Variables de salida

Cuando se configura, esta acción produce variables a las que se puede hacer referencia mediante la configuración de acción de una acción descendente en la canalización. Esta acción produce variables que se pueden ver como variables de salida, incluso si la acción no tiene un espacio de nombres. Configure una acción con un espacio de nombres para que esas variables estén disponibles para la configuración de las acciones posteriores.

Para obtener más información, consulte Referencia de variables.

HighestScannedSeverity

El resultado de mayor gravedad del análisis. Los valores válidos son `medium | high | critical`.

## Permisos del rol de servicio: acción **`InspectorScan`**

Para la compatibilidad con la acción `InspectorScan`, agregue lo siguiente a su declaración de política:

```
{
        "Effect": "Allow",
        "Action": "inspector-scan:ScanSbom",
        "Resource": "*"
    },
    {
```

```
        "Effect": "Allow",
        "Action": [
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
        ],
        "Resource": "resource_ARN"
    },
```

Además, si aún no los has añadido para la acción Comandos, añade los siguientes permisos a tu rol de servicio para poder ver CloudWatch los registros.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Reduzca los permisos al nivel de recursos de la canalización a través de los permisos basados en recursos de la declaración de la política del rol de servicio.

## Declaración de acciones

YAML

```
name: Scan
actionTypeId:
  category: Invoke
  owner: AWS
  provider: InspectorScan
  version: '1'
runOrder: 1
configuration:
  InspectorRunMode: SourceCodeScan
```

```
outputArtifacts:
- name: output
inputArtifacts:
- name: SourceArtifact
region: us-east-1
```

JSON

```
{
                    "name": "Scan",
                    "actionTypeId": {
                        "category": "Invoke",
                        "owner": "AWS",
                        "provider": "InspectorScan",
                        "version": "1"
                    },
                    "runOrder": 1,
                    "configuration": {
                        "InspectorRunMode": "SourceCodeScan"
                    },
                    "outputArtifacts": [
                        {
                            "name": "output"
                        }
                    ],
                    "inputArtifacts": [
                        {
                            "name": "SourceArtifact"
                        }
                    ],
                    "region": "us-east-1"
                },
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Para obtener más información acerca de Amazon Inspector, consulte la Guía del usuario de Amazon Inspector.

# AWS Lambda invocar referencia a la acción

Le permite ejecutar una función de Lambda como una acción en la canalización. Al utilizar el objeto de evento que es una entrada para esta función, la función tiene acceso a la configuración de la acción, las ubicaciones de los artefactos de entrada, las ubicaciones de los artefactos de salida y otra información necesaria para obtener acceso a los artefactos. Para ver un evento de ejemplo que se ha trasladado a una función de invocación de Lambda, consulte Ejemplo de evento JSON. Como parte de la implementación de la función Lambda, debe haber una llamada a `PutJobSuccessResult API` o `PutJobFailureResult API`. De lo contrario, la ejecución de esta acción se bloquea hasta que se agote el tiempo de espera de dicha acción. Si especifica artefactos de salida para la acción, estos deben cargarse en el bucket de S3 como parte de la implementación de la función.

> **Important**
>
> No registre el evento JSON que se CodePipeline envía a Lambda, ya que esto puede provocar que las credenciales de usuario se registren en CloudWatch los registros. El CodePipeline rol usa un evento JSON para pasar credenciales temporales a Lambda en el `artifactCredentials` campo. Para ver un ejemplo de evento, consulte Ejemplo de evento JSON.

## Tipo de acción

- Categoría: `Invoke`
- Propietario: `AWS`
- Proveedor: `Lambda`
- Versión: `1`

## Parámetros de configuración

FunctionName

Obligatorio: sí

`FunctionName` es el nombre de la función creada en Lambda.

UserParameters

Obligatorio: no

Una cadena que la función de Lambda puede procesar como entrada.

## Artefactos de entrada

- Número de artefactos: `0 to 5`
- Descripción: el conjunto de artefactos que se pondrán a disposición de la función de Lambda.

## Artefactos de salida

- Número de artefactos: `0 to 5`
- Descripción: el conjunto de artefactos que la función de Lambda produce como salida.

## Variables de salida

Esta acción generará como variables todos los pares clave-valor que se incluyen en la `outputVariables` sección de la PutJobSuccessResult solicitud de API.

Para obtener más información sobre las variables en CodePipeline, consulte. Referencia de variables

## Ejemplo de configuración de una acción

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"
        }
    ]
},
```

## Ejemplo de evento JSON

La acción de Lambda envía un evento JSON que contiene el ID de trabajo, la configuración de la acción de canalización, las ubicaciones de artefactos de entrada y salida y cualquier información de cifrado de los artefactos. El proceso de trabajo obtiene acceso a estos detalles para completar la acción de Lambda. Para obtener más información, consulte Detalles del trabajo. El siguiente es un evento de ejemplo.

```
{
    "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"
            }
        }
    }
}
```

El evento JSON proporciona los siguientes detalles del trabajo para la acción de Lambda en CodePipeline:

- `id`: el ID exclusivo generado por el sistema del trabajo.
- `accountId`: el ID de AWS cuenta asociado al trabajo.
- `data`: otra información necesaria para que un proceso de trabajo complete el trabajo.

- `actionConfiguration`: los parámetros de acción de la acción de Lambda. Para ver definiciones, consulte Parámetros de configuración.
- `inputArtifacts`: el artefacto suministrado a la acción.
  - `location`: la ubicación de almacenamiento de artefactos.
    - `s3Location`: la información de ubicación del artefacto de entrada para la acción.
      - `bucketName`: el nombre del almacenamiento de artefactos de canalización para el bucket de Amazon S3 denominado codepipeline-us-east-2-1234567890).
      - `objectKey`: el nombre de la aplicación (por ejemplo, `CodePipelineDemoApplication.zip`).
    - `type`: el tipo de artefacto en la ubicación. Actualmente, `S3` es el único tipo de artefacto válido.
  - `revision`: el ID de revisión del artefacto. Según el tipo de objeto, puede ser un ID de confirmación (GitHub) o un ID de revisión (Amazon Simple Storage Service). Para obtener más información, consulte ArtifactRevision.
  - `name`: el nombre del artefacto en el que se va a trabajar, como, por ejemplo, `MyApp`.
- `outputArtifacts`: la salida de la acción.
  - `location`: la ubicación de almacenamiento de artefactos.
    - `s3Location`: la información de ubicación del artefacto de salida para la acción.
      - `bucketName`: el nombre del almacenamiento de artefactos de canalización para el bucket de Amazon S3 denominado codepipeline-us-east-2-1234567890).
      - `objectKey`: el nombre de la aplicación (por ejemplo, `CodePipelineDemoApplication.zip`).
    - `type`: el tipo de artefacto en la ubicación. Actualmente, `S3` es el único tipo de artefacto válido.
  - `revision`: el ID de revisión del artefacto. Según el tipo de objeto, puede ser un ID de confirmación (GitHub) o un ID de revisión (Amazon Simple Storage Service). Para obtener más información, consulte ArtifactRevision.
  - `name`: el nombre de la salida de un artefacto, como, por ejemplo, `MyApp`.
- `artifactCredentials`: Las credenciales de AWS sesión utilizadas para acceder a los artefactos de entrada y salida del bucket de Amazon S3. Estas credenciales son credenciales temporales emitidas por AWS Security Token Service (AWS STS).
  - `secretAccessKey`: la clave de acceso secreta de la sesión.

    - `sessionToken`: el token de la sesión.
    - `accessKeyId`: El ID de la clave de acceso de la sesión.
- `continuationToken`: un token generado por la acción. Las acciones futuras utilizan este token para identificar la instancia en ejecución de la acción. Cuando la acción se haya completado, no se debe proporcionar ningún token de continuación.
- `encryptionKey`: La clave de cifrado utilizada para cifrar los datos del almacén de artefactos, como una AWS KMS clave. Si no se ha definido, se utiliza la clave predeterminada de Amazon Simple Storage Service.
    - `id`: el ID utilizado para identificar la clave. Si se trata de una clave de AWS KMS , puede utilizar el ID de la clave, el ARN de la clave o el ARN del alias.
    - `type`: el tipo de clave de cifrado, como, por ejemplo, la clave de AWS KMS .

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- AWS CloudFormation Guía del usuario: para obtener más información sobre las acciones y los CloudFormation artefactos de Lambda para las canalizaciones, consulte Uso de funciones de anulación de parámetros en CodePipeline canalizaciones, Automatización de la implementación de aplicaciones y artefactos. Lambda-based AWS CloudFormation
- Invoca un AWS Lambda función en una canalización en CodePipeline: este procedimiento proporciona un ejemplo de función de Lambda y muestra cómo utilizar la consola para crear una canalización con una acción de invocación de Lambda.

# AWS OpsWorks Referencia de la acción de implementación de

Usas una AWS OpsWorks acción para implementarla con el OpsWorks uso de tu canalización.

## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `OpsWorks`
- Versión: `1`

## Parámetros de configuración

App

Obligatorio: sí

La OpsWorks aplicación. La aplicación representa el código que desea implementar y ejecutar.

Pila

Obligatorio: sí

La OpsWorks pila. Una pila es un contenedor para la infraestructura de aplicaciones.

Capa

Obligatorio: no

La OpsWorks capa. Una capa especifica la configuración y los recursos de un conjunto de instancias.

## Artefactos de entrada

- Número de artefactos: `1`
- Descripción: es el artefacto de entrada para la acción.

## Artefactos de salida

- Número de artefactos: `0 to 1`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Permisos de rol de servicio: AWS OpsWorks acción

Para obtener AWS OpsWorks asistencia, añada lo siguiente a su declaración de política:

```
{
    "Effect": "Allow",
    "Action": [
        "opsworks:CreateDeployment",
        "opsworks:DescribeApps",
```

```
        "opsworks:DescribeCommands",
        "opsworks:DescribeDeployments",
        "opsworks:DescribeInstances",
        "opsworks:DescribeStacks",
        "opsworks:UpdateApp",
        "opsworks:UpdateStack"
    ],
    "Resource": "resource_ARN"
},
```

## Ejemplo de configuración de una acción

### YAML

```
Name: ActionName
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Version: 1
  Provider: OpsWorks
InputArtifacts:
  - Name: myInputArtifact
Configuration:
  Stack: my-stack
  App: my-app
```

### JSON

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "OpsWorks"
    },
    "InputArtifacts": [
        {
            "Name": "myInputArtifact"
        }
    ],
    "Configuration": {
        "Stack": "my-stack",
```

```
        "App": "my-app"
    }
}
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- AWS OpsWorks Guía del usuario: para obtener información sobre la implementación con AWS OpsWorks, consulte la Guía del AWS OpsWorks usuario.

# AWS Service Catalog Referencia de la acción de implementación de

Usas una AWS Service Catalog acción para implementar plantillas mediante tu canalización. Se trata de plantillas de recursos que ha creado en Service Catalog.

## Tipo de acción

- Categoría: `Deploy`
- Propietario: `AWS`
- Proveedor: `ServiceCatalog`
- Versión: `1`

## Parámetros de configuración

TemplateFilePath

Obligatorio: sí

La ruta del archivo de la plantilla de recursos en la ubicación de origen.

ProductVersionName

Obligatorio: sí

La versión del producto en Service Catalog.

ProductType

Obligatorio: sí

El tipo de producto en Service Catalog.

ProductId

Obligatorio: sí

El ID de producto en Service Catalog.

ProductVersionDescription

Obligatorio: no

La descripción de la versión del producto en Service Catalog.

## Artefactos de entrada

- Número de artefactos: `1`
- Descripción: es el artefacto de entrada para la acción.

## Artefactos de salida

- Número de artefactos: `0`
- Descripción: los artefactos de salida no se aplican a este tipo de acción.

## Permisos del rol de servicio: acción de Service Catalog

Para admitir Service Catalog, añada lo siguiente a su instrucción de política:

```
{
    "Effect": "Allow",
    "Action": [
        "servicecatalog:ListProvisioningArtifacts",
        "servicecatalog:CreateProvisioningArtifact",
        "servicecatalog:DescribeProvisioningArtifact",
        "servicecatalog:DeleteProvisioningArtifact",
        "servicecatalog:UpdateProduct"
    ],
```

```
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:ValidateTemplate"
    ],
    "Resource": "resource_ARN"
}
```

## Ejemplos de configuraciones de la acción por tipo de archivo de configuración

En el ejemplo siguiente se muestra una configuración válida para una acción de implementación que utiliza Service Catalog, para una canalización creada en la consola sin un archivo de configuración distinto:

```
"configuration": {
  "TemplateFilePath": "S3_template.json",
  "ProductVersionName": "devops S3 v2",
  "ProductType": "CLOUD_FORMATION_TEMPLATE",
  "ProductVersionDescription": "Product version description",
  "ProductId": "prod-example123456"
}
```

En el ejemplo siguiente se muestra una configuración válida para una acción de implementación que utiliza Service Catalog, para una canalización creada en la consola con un archivo de configuración de `sample_config.json`distinto:

```
"configuration": {
  "ConfigurationFilePath": "sample_config.json",
  "ProductId": "prod-example123456"
}
```

### Ejemplo de configuración de una acción

YAML

```
Name: ActionName
ActionTypeId:
  Category: Deploy
```

```
  Owner: AWS
  Version: 1
  Provider: ServiceCatalog
OutputArtifacts:
- Name: myOutputArtifact
Configuration:
  TemplateFilePath: S3_template.json
  ProductVersionName: devops S3 v2
  ProductType: CLOUD_FORMATION_TEMPLATE
  ProductVersionDescription: Product version description
  ProductId: prod-example123456
```

JSON

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "ServiceCatalog"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
    ],
    "Configuration": {
        "TemplateFilePath": "S3_template.json",
        "ProductVersionName": "devops S3 v2",
        "ProductType": "CLOUD_FORMATION_TEMPLATE",
        "ProductVersionDescription": "Product version description",
        "ProductId": "prod-example123456"
    }
}
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Guía del usuario de Service Catalog: para obtener información sobre los recursos y las plantillas de Service Catalog, consulte la Guía del usuario de Service Catalog.

- Tutorial: Crear una canalización que se implemente en Service Catalog— En este tutorial se muestra cómo crear y configurar una canalización para implementar la plantilla de producto en Service Catalog y entregar los cambios que haya realizado en su repositorio de origen.

# AWS Step Functions invocar referencia de acción

Una AWS CodePipeline acción que hace lo siguiente:

- Inicia la ejecución de una máquina de AWS Step Functions estados desde tu canalización.
- Proporciona a la máquina de estados un estado inicial a través de una propiedad de la configuración de la acción o de un archivo ubicado en un artefacto de la canalización que se va a pasar como entrada.
- Si lo desea, puede especificar un prefijo del ID de ejecución para identificar las ejecuciones que se originan en la acción.
- Admite máquinas de estados estándar y rápidas .

> **Note**
>
> La acción Step Functions se ejecuta en Lambda y, por lo tanto, tiene cuotas de tamaño de artefacto que son las mismas que las cuotas de tamaño de artefacto de las funciones de Lambda. Para obtener más información, consulte Cuotas de Lambda en la Guía para desarrolladores de Lambda.

## Tipo de acción

- Categoría: `Invoke`
- Propietario: `AWS`
- Proveedor: `StepFunctions`
- Versión: `1`

# Parámetros de configuración

StateMachineArn

Obligatorio: sí

Nombre del recurso de Amazon (ARN) de la máquina de estados que se va a invocar.

ExecutionNamePrefix

Obligatorio: no

De forma predeterminada, el ID de ejecución de la acción se utiliza como nombre de ejecución de la máquina de estados. Si se proporciona un prefijo, se antepone al ID de ejecución de la acción con un guión, y todo ello se utiliza en conjunto como el nombre de ejecución de la máquina de estados.

```
myPrefix-1624a1d1-3699-43f0-8e1e-6bafd7fde791
```

En las máquinas de estados rápidas, el nombre solo debe contener los caracteres 0-9, A-Z, a-z, - y _.

InputType

Obligatorio: no

- Literal (valor predeterminado): cuando se especifica, el valor del campo Input se pasa directamente a la entrada de la máquina de estados.

  Ejemplo de entrada del campo Input cuando se selecciona Literal:

  ```
  {"action": "test"}
  ```

- FilePath: El contenido de un archivo en el artefacto de entrada especificado en el campo de entrada se utiliza como entrada para la ejecución de la máquina de estados. Se requiere un artefacto de entrada cuando InputTypese establece en. FilePath

  Ejemplo de entrada para el campo de entrada cuando FilePathestá seleccionado:

  ```
  assets/input.json
  ```

Input

Obligatorio: condicional

- Literal: si InputTypese establece en Literal (predeterminado), este campo es opcional.

  Si se proporciona, el campo Input se utiliza directamente como entrada en la ejecución de la máquina de estados. De lo contrario, la máquina de estados se invoca con un objeto JSON vacío, `{}`.

- FilePath: Si InputTypeestá establecido en FilePath, este campo es obligatorio.

  Si se establece en, también InputTypese requiere un artefacto de entrada. FilePath

  El contenido del archivo del artefacto de entrada especificado se utiliza como entrada para la ejecución de la máquina de estados.

## Artefactos de entrada

- Número de artefactos: `0 to 1`
- Descripción: Si InputTypese establece en FilePath, este artefacto es obligatorio y se utiliza como fuente de entrada para la ejecución de la máquina de estados.

## Artefactos de salida

- Número de artefactos: `0 to 1`
- Descripción:
  - Máquinas de estados estándar: si se proporciona el artefacto de salida, se rellena con la salida de la máquina de estados. Esto se obtiene de la `output` propiedad de la respuesta de la DescribeExecution API Step Functions después de que la ejecución de la máquina de estados se complete correctamente.
  - Máquinas de estados rápidas: no son compatibles.

## Variables de salida

Esta acción produce variables de salida a las que se puede hacer referencia en la configuración de una acción descendente de la canalización.

Para obtener más información, consulte Referencia de variables.

StateMachineArn

ARN de la máquina de estados.

ExecutionArn

ARN de la ejecución de la máquina de estados. Solo máquinas de estados estándar.

## Permisos del rol de servicio: acción `StepFunctions`

A continuación se indican los permisos mínimos que se necesitan en `StepFunctions` para crear canalizaciones con una acción de invocación de Step Functions.

```
{
    "Effect": "Allow",
    "Action": [
        "states:DescribeStateMachine",
        "states:DescribeExecution",
        "states:StartExecution"
    ],
    "Resource": "resource_ARN"
},
```

## Ejemplo de configuración de una acción

### Ejemplo de una entrada predeterminada

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"
    }
}
```

## Ejemplo de una entrada literal

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\"}"
    }
}
```

## Ejemplo de un archivo de entrada

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"
    }
}
```

## Comportamiento

Durante una versión, CodePipeline ejecuta la máquina de estados configurada utilizando la entrada especificada en la configuración de la acción.

Cuando InputTypese establece en Literal, el contenido del campo de configuración de la acción de entrada se utiliza como entrada para la máquina de estados. Cuando no se proporciona una entrada literal, la ejecución de la máquina de estados utiliza un objeto JSON vacío, `{}`. Para obtener más información sobre cómo ejecutar una máquina de estados sin entrada, consulta la StartExecutionAPI Step Functions.

Cuando InputTypese establece en FilePath, la acción descomprime el artefacto de entrada y utiliza el contenido del archivo especificado en el campo de configuración de la acción de entrada como entrada para la máquina de estados. Si FilePathse especifica, el campo de entrada es obligatorio y debe existir un artefacto de entrada; de lo contrario, la acción fallará.

Después de una ejecución de inicio correcta, el comportamiento será distinto en los dos tipos de máquina de estados, estándar y rápida.

### Máquinas de estados estándar

Si la ejecución de la máquina de estado estándar se inició correctamente, CodePipeline sondea la `DescribeExecution` API hasta que la ejecución alcance un estado terminal. Si la ejecución se completa correctamente, la acción será correcta; de lo contrario, se producirá un error.

Si hay un artefacto de salida configurado, contendrá el valor de retorno de la máquina de estados. Esto se obtiene de la `output` propiedad de la respuesta de la DescribeExecution API Step Functions después de que la ejecución de la máquina de estados se complete correctamente. Tenga en cuenta que existen restricciones en la longitud de salida de esta API.

Gestión de errores

- Si la acción no puede iniciar la ejecución de una máquina de estados, se producirá un error.
- Si la ejecución de la máquina de estados no alcanza un estado terminal antes de que la acción de CodePipeline Step Functions alcance su tiempo de espera (7 días por defecto), se produce un error en la ejecución de la acción. A pesar de este error, la máquina de estados podría seguir funcionando. Para obtener más información sobre los tiempos de espera de ejecución de las máquinas de estado en Step Funcions, consulte Flujos de trabajo estándar en comparación con flujos de trabajo rápidos.

  > **Note**
  >
  > Puede solicitar un aumento de la cuota del tiempo de espera de la acción de invocación en la cuenta que realiza la acción. Sin embargo, el aumento de cuota se aplicará a todas las acciones de ese tipo en todas las regiones de dicha cuenta.

- Si la ejecución de la máquina de estados alcanza el estado terminal FAILED, TIMED_OUT o ABORTED, se producirá un error.

### Máquinas de estados rápidas

Si la ejecución de la máquina de estados rápida se inició correctamente, la ejecución de la acción de invocación también se realizará correctamente.

Consideraciones sobre las acciones configuradas para máquinas de estados rápidas:

- No puede designar un artefacto de salida.
- La acción no espera a que se complete la ejecución de la máquina de estados.
- Una vez iniciada la ejecución de la acción CodePipeline, la ejecución de la acción se realiza correctamente aunque la ejecución de la máquina de estados falle.

Gestión de errores

- Si CodePipeline no se puede iniciar la ejecución de una máquina de estados, se produce un error en la ejecución de la acción. De lo contrario, la acción terminará correctamente de inmediato. La acción se lleva a cabo correctamente CodePipeline independientemente del tiempo que tarde en completarse la ejecución de la máquina de estados o del resultado.

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- AWS Step Functions Guía para desarrolladores: para obtener información sobre las máquinas de estado, las ejecuciones y las entradas de las máquinas de estado, consulte la Guía para AWS Step Functions desarrolladores.
- Tutorial: Usa una acción de AWS Step Functions invocación en una canalización: este tutorial le permite empezar con un ejemplo de máquina de estados estándar y le muestra cómo usar la consola para actualizar una canalización añadiendo una acción de invocación de Step Functions.

# Referencia de estructura de las reglas

En esta sección se hace referencia únicamente a la configuración de reglas. Para obtener información general conceptual sobre la estructura de la canalización, consulte CodePipeline referencia de estructura de tuberías.

Cada proveedor de reglas de CodePipeline utiliza un conjunto de campos de configuración obligatorios y opcionales en la estructura de la canalización. En esta sección se proporciona la siguiente información de referencia según el proveedor de reglas:

- Valores válidos para los campos `RuleType` incluidos en el bloque de regla de estructura de canalización, como, por ejemplo, `Owner` y `Provider`.
- Descripciones y otra información de referencia sobre los parámetros `Configuration` (obligatorios y opcionales) incluidos en la sección de regla de estructura de canalización.
- Ejemplos de campos de configuración de reglas JSON y YAML válidos.

La información de referencia está disponible para los siguientes proveedores de reglas:

Temas


- CloudWatchAlarm
- Regla de CodeBuild
- Comandos
- DeploymentWindow
- LambdaInvoke
- VariableCheck


# CloudWatchAlarm

Al crear una condición, puede agregar la regla `CloudWatchAlarm`. En esta sección, se proporciona una referencia para los parámetros de reglas. Para obtener más información acerca de las reglas y condiciones, consulte Funcionamiento de las condiciones de las etapas.

Debes haber creado ya una alarma en Amazon CloudWatch como recurso independiente.

Temas

- Tipo de regla
- Parámetros de configuración
- Ejemplo de configuraciones de regla
- Véase también


## Tipo de regla

- Categoría: `Rule`
- Propietario: `AWS`
- Proveedor: `CloudWatchAlarm`
- Versión: `1`

## Parámetros de configuración

AlarmName

Obligatorio: sí

El nombre de la CloudWatch alarma. Se trata de un recurso independiente creado en CloudWatch.

AlarmStates

Obligatorio: no

Los estados de CloudWatch alarma deseados para que la regla los supervise. Los valores válidos son ALARM, OK y INSUFFICIENT_DATA.

WaitTime

Obligatorio: no

El tiempo de espera en minutos para permitir un cambio de estado antes de ejecutar el resultado de la regla. Por ejemplo, configure 20 minutos para esperar a que un estado de alarma cambie a OK antes de aplicar el resultado de la regla.

# Ejemplo de configuraciones de regla

## YAML

```
rules:
    - name: MyMonitorRule
      ruleTypeId:
        category: Rule
        owner: AWS
        provider: CloudWatchAlarm
        version: '1'
      configuration:
        AlarmName: CWAlarm
        WaitTime: '1'
      inputArtifacts: []
      region: us-east-1
```

## JSON

```
                        "rules": [
                            {
                                "name": "MyMonitorRule",
                                "ruleTypeId": {
                                    "category": "Rule",
                                    "owner": "AWS",
                                    "provider": "CloudWatchAlarm",
                                    "version": "1"
                                },
                                "configuration": {
                                    "AlarmName": "CWAlarm",
                                    "WaitTime": "1"
                                },
                                "inputArtifacts": [],
                                "region": "us-east-1"
                            }
                        ]
                    }
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta regla.

- Creación de condiciones de fallo: en esta sección se proporcionan los pasos necesarios para crear una condición de fallo con una regla de alarma.

# Regla de CodeBuild

Al crear una condición, puede agregar la regla de CodeBuild. En esta sección, se proporciona una referencia para los parámetros de reglas. Para obtener más información acerca de las reglas y condiciones, consulte Funcionamiento de las condiciones de las etapas.

Puede usar la regla de CodeBuild para crear una condición para que la ejecución del proyecto de compilación sea correcta si cumple los criterios de la regla, por ejemplo, que la ejecución de la compilación sea correcta para una condición beforeEntry.

> **Note**
>
> Para las condiciones beforeEntry que se configuran con el resultado Skip, solo están disponibles las reglas `LambdaInvoke` y `VariableCheck`.

Temas


- Permisos para las políticas de roles de servicio
- Tipo de regla
- Parámetros de configuración
- Ejemplo de configuraciones de regla
- Véase también


## Permisos para las políticas de roles de servicio

Para conceder permisos a esta regla, agregue lo siguiente a la declaración de la política de rol de servicio de CodePipeline. Reduzca los permisos al nivel de recurso.

```
{
```

```
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
    ],
    "Resource": "resource_ARN"
},
```

## Tipo de regla

- Categoría: `Rule`
- Propietario: : `AWS`
- Proveedor: `CodeBuild`
- Versión: `1`

## Parámetros de configuración

ProjectName

Obligatorio: sí

`ProjectName` es el nombre del proyecto de compilación de CodeBuild.

PrimarySource

Obligatorio: condicional

El valor del parámetro `PrimarySource` debe ser el nombre de uno de los artefactos de entrada de la acción. CodeBuild busca el archivo buildspec y ejecuta los comandos buildspec en el directorio que contiene la versión descomprimida de este artefacto.

Este parámetro es necesario si se especifican varios artefactos de entrada para una acción de CodeBuild. Cuando solo hay un artefacto de origen para la acción, el artefacto `PrimarySource` se establece como valor predeterminado de dicho artefacto.

BatchEnabled

Obligatorio: no

El valor booleano del parámetro `BatchEnabled` permite que la acción ejecute varias compilaciones en la misma ejecución de compilación.

Cuando esta opción está habilitada, la opción `CombineArtifacts` está disponible.

Para ver ejemplos de canalizaciones con compilaciones por lotes habilitadas, consulte Integración de CodePipeline con CodeBuild y compilaciones por lotes.

CombineArtifacts

Obligatorio: no

El valor booleano del parámetro `CombineArtifacts` combina todos los artefactos de compilación de una compilación por lotes en un único archivo de artefactos para la acción de compilación.

Para utilizar esta opción, el parámetro `BatchEnabled` debe estar activado.

EnvironmentVariables

Obligatorio: no

El valor de este parámetro se utiliza para establecer variables de entorno para la acción de CodeBuild de la canalización. El valor del parámetro `EnvironmentVariables` toma la forma de una matriz JSON de objetos de variables de entorno. Consulte el parámetro de ejemplo en Declaración de acción (CodeBuild ejemplo).

Cada objeto tiene tres partes, todas las cuales son cadenas:

- `name`: el nombre o la clave de la variable de entorno.
- `value`: el valor de la variable de entorno. Al utilizar el tipo `PARAMETER_STORE` o `SECRETS_MANAGER`, este valor debe ser el nombre de un parámetro que ya haya almacenado en el almacén de parámetros de AWS Systems Manager o un secreto en el que ya haya almacenado en AWS Secrets Manager, respectivamente.

  > **Note**
  >
  > Se desaconseja usar variables de entorno para almacenar valores confidenciales, especialmente las credenciales de AWS. Cuando utiliza la consola de CodeBuild o la CLI de AWS, las variables de entorno se muestran en texto sin formato. Para valores confidenciales, se recomienda utilizar el tipo `SECRETS_MANAGER`en su lugar.

- `type`: (opcional) el tipo de la variable de entorno. Los valores válidos son `PARAMETER_STORE`, `SECRETS_MANAGER` o `PLAINTEXT`. Si no se especifica, toma el valor predeterminado `PLAINTEXT`.

> **Note**
>
> Cuando introduzca los valores `name`, `value`, y `type` de configuración de las variables de entorno, especialmente si la variable de entorno contiene sintaxis de variable de salida de CodePipeline, no supere el límite de 1000 caracteres del campo de valor de la configuración. Cuando se supera este límite, se devuelve un error de validación.

Para obtener más información, consulte EnvironmentVariable en la AWS CodeBuildAPI Reference. Para ver un ejemplo de acción de CodeBuild con una variable de entorno que se resuelva en el nombre de la ramificación de GitHub, consulte Ejemplo: usa una BranchName variable con variables de CodeBuild entorno.

## Ejemplo de configuraciones de regla

YAML

```
name: codebuild-rule
ruleTypeId:
  category: Rule
  owner: AWS
  provider: CodeBuild
  version: '1'
configuration:
  ProjectName: my-buildproject
  EnvironmentVariables: '[{"name":"VAR1","value":"variable","type":"PLAINTEXT"}]'
inputArtifacts:
- name: SourceArtifact
region: us-east-1
```

JSON

```
{
    "name": "codebuild-rule",
    "ruleTypeId": {
        "category": "Rule",
        "owner": "AWS",
        "provider": "CodeBuild",
        "version": "1"
    },
```

```
    "configuration": {
        "ProjectName": "my-buildproject"
    },
    "inputArtifacts": [
        {
            "name": "SourceArtifact",
            "EnvironmentVariables": "[{\"name\":\"VAR1\",\"value\":\"variable\",
\"type\":\"PLAINTEXT\"}]"
        }
    ],
    "region": "us-east-1"
}
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta regla.

- Para obtener más información sobre las reglas y condiciones, consulte Condition, RuleTypeId y RuleExecution en la Guía de la API de CodePipeline.

# Comandos

Al crear una condición, puede agregar la regla `Commands`. En esta sección, se proporciona una referencia para los parámetros de reglas. Para obtener más información acerca de las reglas y condiciones, consulte Funcionamiento de las condiciones de las etapas.

Puede usar la regla `Commands` para crear una condición en virtud de la cual, para la correcta ejecución de los comandos, se deben cumplir los criterios de la regla, como que la salida y la ruta del archivo de los comandos sean correctos para una condición beforeEntry.

> **Note**
>
> Para las condiciones beforeEntry que se configuran con el resultado Skip, solo están disponibles las reglas `LambdaInvoke` y `VariableCheck`.

Temas

- Consideraciones sobre la regla de comandos

- Permisos para las políticas de roles de servicio
- Tipo de regla
- Parámetros de configuración
- Ejemplo de configuraciones de regla
- Véase también


## Consideraciones sobre la regla de comandos

Las siguientes consideraciones se aplican a la regla de comandos.

- La regla de comandos usa CodeBuild recursos similares a los de la CodeBuild acción y, al mismo tiempo, permite los comandos del entorno de shell en una instancia de cómputo virtual sin necesidad de asociar o crear un proyecto de compilación.

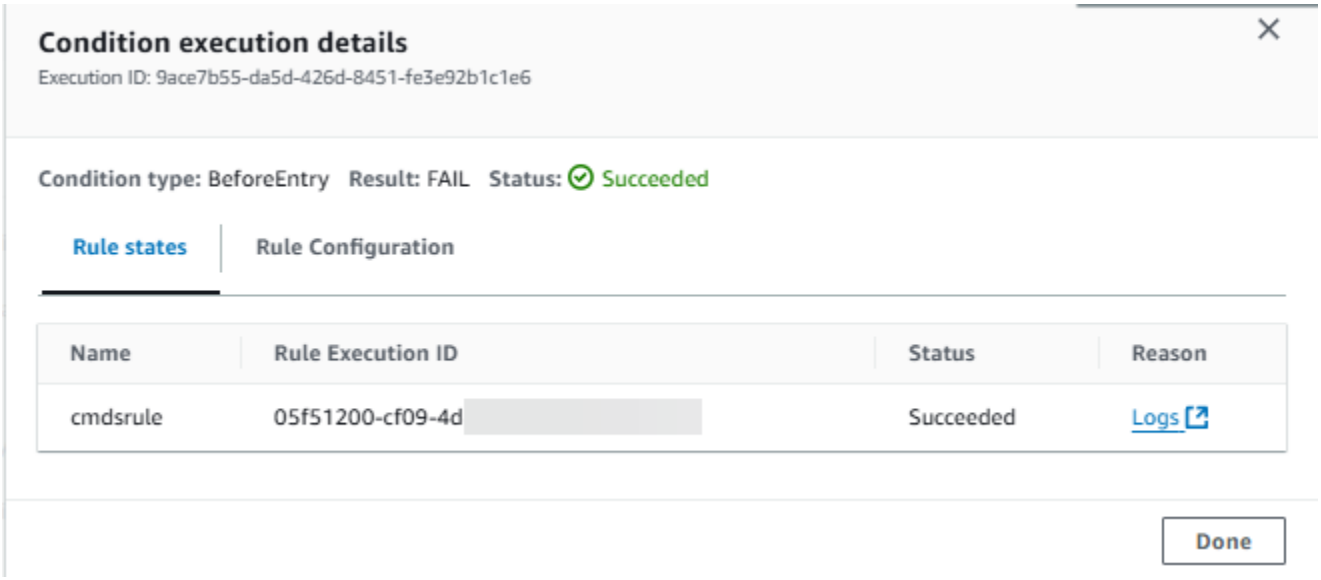
  > **Note**
  >
  > La ejecución de la regla de comandos generará cargos separados. AWS CodeBuild

- Como la regla de comandos CodePipeline utiliza CodeBuild recursos, las compilaciones ejecutadas por la acción se atribuirán a los límites de creación de tu cuenta en CodeBuild. Las compilaciones ejecutadas por la regla de comandos se tendrán en cuenta para los límites de compilación simultánea configurados para esa cuenta.
- El tiempo de espera para las compilaciones con la regla de comandos es de 55 minutos, según las CodeBuild compilaciones.
- La instancia de procesamiento utiliza un entorno de compilación aislado en CodeBuild.

  > **Note**
  >
  > Debido a que el entorno de compilación aislado se usa a nivel de cuenta, es posible que una instancia se reutilice para otra ejecución de canalización.

- Se admiten todos los formatos, excepto los formatos multilínea. Debe utilizar el formato de una sola línea al introducir comandos.
- Para esta regla, CodePipeline asumirá la función de servicio de canalización y la usará para permitir el acceso a los recursos en tiempo de ejecución. Se recomienda configurar el rol de servicio para que los permisos se limiten al nivel de acción.

- Los permisos agregados a la función CodePipeline de servicio se detallan enAgrega permisos al rol de CodePipeline servicio.
- El permiso necesario para ver los registros en la consola se detalla en Permisos necesarios para ver los registros de procesamiento en la consola. En las siguientes pantallas de ejemplo, utilice el enlace Registros para ver los registros y ver si una regla de comandos en los CloudWatch registros es correcta.







- A diferencia de otras acciones CodePipeline, no se establecen campos en la configuración de acciones; se establecen los campos de configuración de acciones fuera de la configuración de acciones.

## Permisos para las políticas de roles de servicio

Cuando CodePipeline se ejecuta la regla, CodePipeline crea un grupo de registros con el nombre de la canalización, tal como se indica a continuación. Esto permite reducir los permisos para registrar los recursos mediante el nombre de la canalización.

```
/aws/codepipeline/MyPipelineName
```

Si utiliza un rol de servicio existente, para utilizar la acción de Comandos tendrá que agregar los siguientes permisos para el rol de servicio.

- registros: CreateLogGroup
- registros: CreateLogStream
- registros: PutLogEvents

En la declaración de las políticas de roles de servicio, limite los permisos al nivel de la canalización como se muestra en el siguiente ejemplo.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
}
```

Para ver los registros en la consola mediante la página del cuadro de diálogo de detalles de la acción, se debe agregar el permiso para ver los registros al rol de la consola. Para obtener más información, consulte el ejemplo de política de permisos para consolas en Permisos necesarios para ver los registros de procesamiento en la consola.

## Tipo de regla

- Categoría: `Rule`
- Propietario: `AWS`

- Proveedor: `Commands`
- Versión: 1

## Parámetros de configuración

Comandos

Obligatorio: sí

Puede proporcionar comandos del intérprete de comandos para que se ejecute la regla `Commands`. En la consola, los comandos se introducen en líneas separadas. En la CLI, los comandos se introducen como cadenas independientes.

> **Note**
>
> Los formatos multilínea no son compatibles y generarán un mensaje de error. Se debe utilizar el formato de una sola línea para introducir comandos en el campo Comandos.

Los siguientes detalles proporcionan la computación predeterminada que se utiliza para la regla de comandos. Para obtener más información, consulte la referencia de tipos y modos de procesamiento del entorno de compilación en la Guía del CodeBuild usuario.

- CodeBuild imagen: aws/codebuild/amazonlinux 2-x86_64-standard:5.0
- Tipo de computación: Linux Small
- Valor de ComputEtype del entorno: BUILD_ _SMALL GENERAL1
- Valor del tipo de entorno: LINUX_CONTAINER

## Ejemplo de configuraciones de regla

YAML

```
result: FAIL
rules:
- name: CommandsRule
  ruleTypeId:
    category: Rule
    owner: AWS
```

```
    provider: Commands
    version: '1'
  configuration: {}
  commands:
  - ls
  - printenv
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
```

JSON

```
{
    "result": "FAIL",
    "rules": [
        {
            "name": "CommandsRule",
            "ruleTypeId": {
                "category": "Rule",
                "owner": "AWS",
                "provider": "Commands",
                "version": "1"
            },
            "configuration": {},
            "commands": [
                "ls",
                "printenv"
            ],
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-east-1"
        }
    ]
}
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta regla.

- Para obtener más información sobre las reglas y condiciones, consulte Condición y RuleExecutionen la Guía de RuleTypeIdAPI. CodePipeline

# DeploymentWindow

Al crear una condición, puede agregar la regla `DeploymentWindow`. En esta sección, se proporciona una referencia para los parámetros de reglas. Para obtener más información acerca de las reglas y condiciones, consulte Funcionamiento de las condiciones de las etapas.

Temas

- Tipo de regla
- Parámetros de configuración
- Ejemplo de configuraciones de regla
- Véase también

## Tipo de regla

- Categoría: `Rule`
- Propietario: `AWS`
- Proveedor: `DeploymentWindow`
- Versión: `1`

## Parámetros de configuración

Cron

Obligatorio: sí

La expresión que define los días y las horas en que se permitirá la implementación. Las expresiones cron se componen de 6 campos obligatorios y un campo opcional separados por un espacio en blanco. Los campos de expresión cron permiten especificar un patrón de programación con una expresión cron de la siguiente manera.

| Nombre del campo | Valores permitidos | Caracteres especiales permitidos |
|---|---|---|
| Segundos | N/A | * |
| Minutos | 0-59 | , - * / |
| Horas | 0-23 | , - * / |
| Day-of-month | 1-31 | , - * ? / L W |
| Mes | 1-12 o JAN-DEC | , - * / |
| Day-of-Week | 1-7 o SUN-SAT | , - * ? / L # |
| Año (opcional) | vacío, 1970-2199 | , - * / |

- El carácter “*” se utiliza para especificar todos los valores. Por ejemplo, “*” en el campo Minutos significa “cada minuto”.
- El carácter “?” se puede utilizar en los campos Día del mes y Día de la semana. Se usa para especificar que “no hay ningún valor específico”. Esto resulta útil cuando necesite especificar algo en uno de los dos campos, pero no en el otro.
- El carácter '-' se usa para especificar los rangos. Por ejemplo, «10-12» en el campo horario significa «las horas 10, 11 y 12».
- El carácter “,” se utiliza para especificar valores adicionales. Por ejemplo, “MON,WED,FRI” en el campo Día de la semana significa “los días lunes (MON), miércoles (WED) y viernes (FRI)”.
- El carácter “/” se usa para especificar los incrementos. Por ejemplo, «0/15» en el campo de segundos significa «los segundos 0, 15, 30 y 45». Y «5/15» en el campo de segundos significa «los segundos 5, 20, 35 y 50». Si se especifica “*” antes del carácter “/”, equivale a especificar que 0 es el valor con el que empezar.
- El carácter “L” se puede utilizar en los campos Día del mes y Día de la semana. Este carácter es la abreviatura de la palabra inglesa “last” (último), pero tiene un significado diferente en cada uno de esos dos campos. Por ejemplo, el valor “L” en el campo Día del mes significa “el último día del mes”: por ejemplo, el día 31 de enero, y el día 28 de febrero en los años no bisiestos. Si se utiliza en el campo Día de la semana por sí solo, significa “7” o “SAT” (sábado, ya que este se considera el último día de la semana en Estados Unidos). Sin embargo, si se utiliza

en el campo Día de la semana después de otro valor, significa “el último <specified_day> día del mes”; por ejemplo, “6L” significa “el último viernes del mes”. También puede especificar un desfase con respecto al último día del mes, por ejemplo «L-3», que significaría el penúltimo día del mes natural.

- El carácter “W” se puede utilizar en el campo Día del mes. Este carácter se usa para especificar el día de la semana (Monday-Friday) más cercano a un día determinado. Por ejemplo, si especificara “15W” como valor para el campo Día del mes, el significado sería: “el día entre semana más cercano al día 15 del mes”. Por lo tanto, si el día 15 fuera sábado, el desencadenador se activaría el viernes 14. Si el día 15 fuera domingo, el desencadenador se activaría el lunes 16. Si el día 15 fuera martes, el desencadenador se activaría el mismo martes 15.
- Los caracteres “L” y “W” también se pueden combinar para que la expresión del día del mes obtenga “LW”, que se traduce como el “último día entre semana del mes”.
- El carácter “#” se puede utilizar en el campo Día de la semana. Este carácter se utiliza para especificar el “.er” o “.º” <specified_day> día del mes. Por ejemplo, el valor de “6#3” en el campo Día de la semana significa el tercer viernes del mes (día 6 = viernes, y “#3” = el 3.er viernes del mes).
- Los caracteres legales y los nombres de los meses y días de la semana no distinguen mayúsculas de minúsculas.

TimeZone

Obligatorio: no

La zona horaria de la ventana de implementación. La expresión regular coincide con los patrones de los siguientes formatos:

- Region/City formato.  El valor coincide con una zona horaria con el formato Region/City Region/City _City. Por ejemplo, `America/New_York` o `Europe/Berlin`.
- Formato UTC. El valor coincide con la cadena UTC seguida, opcionalmente, de un desfase en el formato + HH:MM o -HH:MM. Por ejemplo, `UTC`, `UTC+05:30` o `UTC-03:00`. Este es el formato predeterminado si el parámetro no está establecido de otro modo.
- Formato de abreviatura. El valor coincide con una abreviatura de entre 3 y 5 caracteres para una zona horaria. Por ejemplo, `EST` o `IST`.

Para ver una tabla de valores de TimeZone ID válidos, consulte https://docs.oracle.com/middleware/1221/wcs/tag-ref/MISC/TimeZones.html. Tenga en cuenta que algunas

abreviaturas son abreviaturas duplicadas, como por ejemplo CST, que puede referirse a la hora estándar central, a la hora estándar de China y a la hora estándar de Cuba.

## Ejemplo de configuraciones de regla

YAML

```
- name: MyDeploymentRule
  ruleTypeId:
    category: Rule
    owner: AWS
    provider: DeploymentWindow
    version: '1'
  configuration:
    Cron: 0 0 9-17 ? * MON-FRI *
    TimeZone: PST
  inputArtifacts: []
  region: us-east-1
```

JSON

```
[
                    {
                        "name": "MyDeploymentRule",
                        "ruleTypeId": {
                            "category": "Rule",
                            "owner": "AWS",
                            "provider": "DeploymentWindow",
                            "version": "1"
                        },
                        "configuration": {
                            "Cron": "0 0 9-17 ? * MON-FRI *",
                            "TimeZone": "PST"
                        },
                        "inputArtifacts": [],
                        "region": "us-east-1"
                    }
                ]
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta regla.

- Creación de condiciones de éxito: en esta sección se proporcionan los pasos para crear una condición de éxito con una regla de ventana de implementación.
- Para obtener más información sobre las reglas y condiciones, consulte Condición y RuleExecution en la Guía de CodePipeline API. RuleTypeId

# LambdaInvoke

Al crear una condición, puede agregar la regla `LambdaInvoke`. En esta sección, se proporciona una referencia para los parámetros de reglas. Para obtener más información acerca de las reglas y condiciones, consulte Funcionamiento de las condiciones de las etapas.

Debe haber creado previamente una función en Lambda como recurso independiente.

Temas

- Tipo de regla
- Parámetros de configuración
- Ejemplo de configuraciones de regla
- Véase también

## Tipo de regla

- Categoría: `Rule`
- Propietario: : `AWS`
- Proveedor: `LambdaInvoke`
- Versión: `1`

## Parámetros de configuración

FunctionName

Obligatorio: sí

El nombre de la función de Lambda.

UserParameters

Obligatorio: no

Estos son parámetros que se proporcionan como entrada para la función en formato de par clave-valor.

## Ejemplo de configuraciones de regla

YAML

```
- name: MyLambdaRule
  ruleTypeId:
    category: Rule
    owner: AWS
    provider: LambdaInvoke
    version: '1'
  configuration:
    FunctionName: my-function
  inputArtifacts:
  - name: SourceArtifact
  region: us-east-1
```

JSON

```
[
    {
        "name": "MyLambdaRule",
        "ruleTypeId": {
            "category": "Rule",
            "owner": "AWS",
            "provider": "LambdaInvoke",
            "version": "1"
        },
        "configuration": {
            "FunctionName": "my-function"
        },
        "inputArtifacts": [
            {
                "name": "SourceArtifact"
```

```
            }
        ],
        "region": "us-east-1"
    }
]
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta regla.

- Creación de condiciones de fallo: en esta sección se proporcionan los pasos necesarios para crear una condición de fallo con una regla de alarma.

# VariableCheck

Al crear una condición, puede agregar la regla `VariableCheck`. En esta sección, se proporciona una referencia para los parámetros de reglas. Para obtener más información acerca de las reglas y condiciones, consulte Funcionamiento de las condiciones de las etapas.

Es posible usar la regla `VariableCheck` para crear una condición en la que la variable de salida se compare con una expresión proporcionada. La regla supera la comprobación cuando el valor de la variable cumple los criterios de la regla, por ejemplo, si el valor es igual o mayor que una variable de salida especificada.

Temas


- Tipo de regla
- Parámetros de configuración
- Ejemplo de configuraciones de regla
- Véase también


## Tipo de regla

- Categoría: `Rule`
- Propietario: `AWS`
- Proveedor: `VariableCheck`

- Versión: 1

## Parámetros de configuración

Operador

Obligatorio: sí

El operador que indica qué operación se debe realizar para la verificación de la variable.

En el siguiente ejemplo, se comprobará si la variable de salida del nombre del repositorio es igual a `MyDemoRepo`.

```
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
```

Los siguientes operadores están disponibles para crear una expresión de la siguiente manera.

- Es igual a: elija este operador para comprobar si la variable es igual al valor de la cadena.

  Parámetro de la CLI: `EQ`
- Contiene: elija este operador para comprobar si la variable contiene el valor de la cadena como una subcadena.

  Parámetro de la CLI: `CONTAINS`
- Coincide: elija este operador para comprobar si la variable coincide con una expresión regex determinada como valor de cadena.

  Todas las expresiones regulares se CodePipeline ajustan a la sintaxis de expresiones regulares de Java. Para obtener una descripción completa de la sintaxis de expresiones regulares de Java y sus constructos, consulte java.util.regex.Pattern.

  Parámetro de la CLI: `MATCHES`
- No es igual a: elija este operador para comprobar si la variable no es igual al valor de la cadena.

  Parámetro de la CLI: `NE`

Variable

Obligatorio: sí

Las variables de canalización que se van a comprobar.

Valor

Obligatorio: sí

El valor de la expresión que se va a comparar.

En el siguiente ejemplo, se comprobará si la variable de salida del nombre del repositorio es igual a `MyDemoRepo`.

```
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
```

En el siguiente ejemplo de JSON, se definen dos reglas distintas, una para una sentencia `EQ` (equals) que comprueba el nombre del repositorio y de la rama con el formato # {. SourceVariables RepositoryName} y otra `CONTAINS` comprueba la variable de salida del mensaje de confirmación con el formato # {. SourceVariables CommitMessage} contra el valor proporcionado «update».

```
"beforeEntry": {
                  "conditions": [
                      {
                          "result": "FAIL",
                          "rules": [
                              {
                                  "name": "MyVarCheckRule",
                                  "ruleTypeId": {
                                      "category": "Rule",
                                      "owner": "AWS",
                                      "provider": "VariableCheck",
                                      "version": "1"
                                  },
                                  "configuration": {
                                      "Operator": "EQ",
                                      "Value": "MyDemoRepo",
```

```
                                    "Variable": "#{SourceVariables.RepositoryName}"
                                },
                                "inputArtifacts": [],
                                "region": "us-east-1"
                            },
                            {
                                "name": "MyVarCheckRuleContains",
                                "ruleTypeId": {
                                    "category": "Rule",
                                    "owner": "AWS",
                                    "provider": "VariableCheck",
                                    "version": "1"
                                },
                                "configuration": {
                                    "Operator": "CONTAINS",
                                    "Value": "update",
                                    "Variable": "#{SourceVariables.CommitMessage}"
                                },
                                "inputArtifacts": [],
                                "region": "us-east-1"
                            }
                        ]
                    }
                ]
            }
        }
    ],
```

## Ejemplo de configuraciones de regla

### YAML

```
- name: MyVariableCheck
  ruleTypeId:
    category: Rule
    owner: AWS
    provider: VariableCheck
    version: '1'
  configuration:
    Variable: "#{SourceVariables.RepositoryName}"
    Value: MyDemoRepo
    Operator: EQ
  inputArtifacts: []
```

```
  region: us-west-2
```

JSON

```
"rules": [
    {
        "name": "MyVariableCheck",
        "ruleTypeId": {
            "category": "Rule",
            "owner": "AWS",
            "provider": "VariableCheck",
            "version": "1"
        },
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
        "inputArtifacts": [],
        "region": "us-west-2"
    }
]
```

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta regla.

- Tutorial: Cómo crear una regla de verificación de variables para una canalización como una condición de entrada: en esta sección se proporciona un tutorial con los pasos para crear una condición de entrada con una regla de verificación de variables.
- Referencia de variables: en esta sección se proporciona información de referencia y ejemplos de las variables de canalización.
- Para obtener más información sobre las reglas y condiciones, consulte Condición y RuleExecutionen la Guía de la CodePipeline API. RuleTypeId

# Referencia del modelo de integración

Existen varias integraciones prediseñadas para servicios de terceros que ayudan a incorporar las herramientas existentes para los clientes en el proceso de lanzamiento de la canalización. Los socios o proveedores de servicios externos utilizan un modelo de integración para implementar los tipos de acción que se pueden usar en CodePipeline.

Utilice esta referencia cuando planifique o trabaje con tipos de acción que se administran con un modelo de integración compatible en CodePipeline.

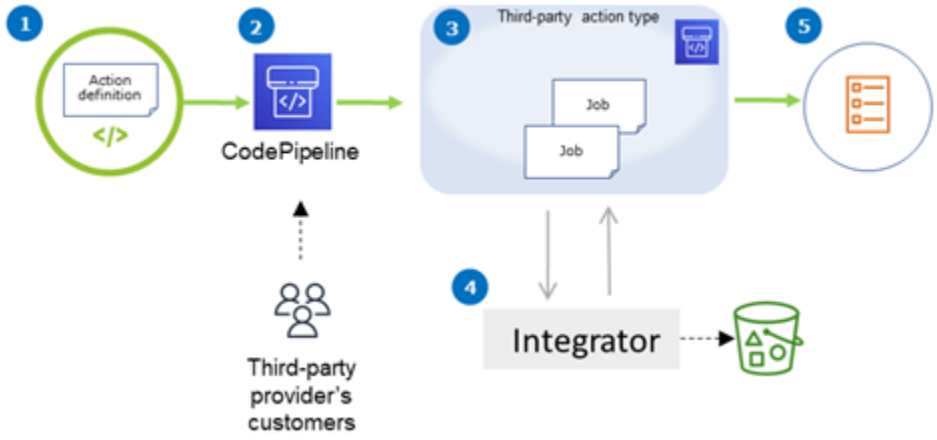
Para certificar tu tipo de acción de terceros como una integración con un socio CodePipeline, consulta la red de AWS socios (APN). Esta información es un complemento de la AWS CLI Reference.

Temas


- Cómo funcionan los tipos de acciones de terceros con el integrador
- Conceptos
- Modelos de integración compatibles
- Modelo de integración de Lambda
- Modelo de integración de procesos de trabajo


## Cómo funcionan los tipos de acciones de terceros con el integrador

Puede añadir tipos de acciones de terceros a las canalizaciones de los clientes para completar las tareas relacionadas con los recursos de los clientes. El integrador gestiona las solicitudes de trabajo y ejecuta la acción con ellas. CodePipeline En el siguiente diagrama, se muestra un tipo de acción de terceros creado para que los clientes lo utilicen en su proceso. Una vez que el cliente configura la acción, ésta se ejecuta y crea solicitudes de trabajo que administra el motor de acciones del integrador.




En el diagrama se muestran los siguientes pasos:

1. La definición de la acción se registra y está disponible en CodePipeline. La acción de terceros está disponible para los clientes del proveedor de terceros.
2. El cliente del proveedor elige y configura la acción en CodePipeline.
3. La acción se ejecuta y los trabajos se ponen en cola. CodePipeline Cuando el trabajo está listo CodePipeline, envía una solicitud de trabajo.
4. El integrador (el trabajador de las API de sondeo de terceros o de la función de Lambda) recoge la solicitud de trabajo, devuelve una confirmación y trabaja en los artefactos de las acciones.
5. El integrador devuelve el success/failure resultado (el trabajador utiliza success/failure las API o la función Lambda envía success/failure el resultado) con un resultado del trabajo y un token de continuación.

Para obtener información sobre los pasos que puede seguir para solicitar, ver y actualizar un tipo de acción, consulte Trabajar con tipos de acciones.

## Conceptos

En esta sección se utilizan los siguientes términos para los tipos de acciones de terceros:

Tipo de acción

Un proceso repetible que se puede reutilizar en canalizaciones que realizan las mismas cargas de trabajo de entrega continua. Los tipos de acciones se identifican mediante `Owner`, `Category`, `Provider` y `Version`. Por ejemplo:

```
            {

                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "CodeDeploy",
                "Version": "1"
            },
```

Todas las acciones del mismo tipo comparten la misma implementación.

Action

Una instancia única de un tipo de acción, uno de los procesos discretos que tienen lugar dentro de una etapa de una canalización. Por lo general, esto incluye los valores de usuario específicos de la canalización en la que se ejecuta esta acción.

Definición de la acción

El esquema de un tipo de acción que define las propiedades necesarias para configurar la acción y input/output los artefactos.

Action execution (Ejecución de acciones)

Conjunto de trabajos que se han ejecutado para determinar si la acción realizada por el cliente se ha realizado correctamente o no.

Motor de ejecución de acciones

Propiedad de la configuración de ejecución de acciones que define el tipo de integración utilizado por un tipo de acción. Los valores válidos son `JobWorker` y `Lambda`.

Integración

Describe un software ejecutado por un integrador para implementar un tipo de acción. CodePipeline admite dos tipos de integración correspondientes a los dos motores de acción compatibles `JobWorker` y`Lambda`.

Integrador

La persona propietaria de la implementación de un tipo de acción.

Trabajo

Un trabajo con el contexto del proceso y del cliente para ejecutar una integración. La ejecución de una acción se compone de una o más tareas.

Proceso de trabajo

El servicio que procesa las entradas del cliente y ejecuta un trabajo.

# Modelos de integración compatibles

CodePipeline tiene dos modelos de integración:

- Modelo de integración Lambda:  este modelo de integración es la forma preferida de trabajar con los tipos de acción en CodePipeline. El modelo de integración de Lambda utiliza una función de Lambda para procesar las solicitudes de trabajo cuando se ejecuta la acción.
- Modelo de integración de proceso de trabajo: el modelo de integración del proceso de trabajo es el modelo utilizado anteriormente para las integraciones de terceros. El modelo de integración de trabajadores laborales utiliza un trabajador laboral configurado para contactar con las CodePipeline API y procesar las solicitudes de trabajo cuando se ejecuta la acción.

A modo de comparación, en la siguiente tabla se describen las características de los dos modelos:

| | Modelo de integración de Lambda | Modelo de integración de procesos de trabajo |
|---|---|---|
| Descripción | El integrador escribe la integración como una función de Lambda, que se invoca CodePipeline siempre que hay un trabajo disponible para la acción. La función de Lambda no sondea los trabajos disponibl es, sino que espera hasta que se reciba la siguiente solicitud de trabajo. | El integrador escribe la integraci ón como un proceso de trabajo que consulta constantemente los puestos disponibles en la cartera del cliente. A continuación, el trabajador ejecuta el trabajo y envía el resultado del trabajo a través de las API. CodePipeline CodePipeline |
| Infraestructura | AWS Lambda | Implemente el código del proceso de trabajo en la infraestructura del integrador, como las instancia s de Amazon EC2. |

| | Modelo de integración de Lambda | Modelo de integración de procesos de trabajo |
|---|---|---|
| Esfuerzo de desarrollo | La integración solo contiene la lógica empresarial. | La integración debe interactuar con CodePipeline las API, además de contener la lógica empresarial. |
| Esfuerzo operativo | Menor esfuerzo operativo, ya que la infraestructura es solo AWS recursos. | Mayor esfuerzo operativo porque el proceso de trabajo necesita su hardware independiente. |
| Tiempo máximo de ejecución de un trabajo | Si la integración debe ejecutarse de forma activa durante más de 15 minutos, no se puede utilizar este modelo. Esta acción está destinada a los integradores que necesitan iniciar un proceso (por ejemplo, iniciar una creación a partir del artefacto de código del cliente) y devolver un resultado cuando finalice. No recomendamos que el integrador siga esperando a que finalice la compilación. En su lugar, devuelva una continuación. CodePipelinecrea un nuevo trabajo en otros 30 segundos si se recibe una continuación del código del integrador para comprobar el estado del trabajo hasta que finalice. | Con este modelo, se puede mantener la ejecución de jobs (hours/days) durante mucho tiempo. |

# Modelo de integración de Lambda

El modelo de integración de Lambda compatible incluye la creación de la función de Lambda y la definición del resultado para el tipo de acción de terceros.

## Actualice su función Lambda para gestionar la entrada desde CodePipeline

Puede crear una función de Lambda nueva. Puede añadir lógica empresarial a la función de Lambda, que se ejecuta siempre que haya un trabajo disponible en su proceso para su tipo de acción. Por ejemplo, dado el contexto del cliente y del proceso, es posible que desee empezar a desarrollar su servicio para el cliente.

Utilice los siguientes parámetros para actualizar la función Lambda de modo que gestione la entrada desde. CodePipeline

Formato:

- `jobId`:
  - El ID exclusivo del trabajo generado por el sistema.
  - Tipo: cadena
  - Patrón: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}
- `accountId`:
  - El ID de la AWS cuenta del cliente que se utilizará al realizar el trabajo.
  - Tipo: cadena
  - Patrón: [0-9]{12}
- `data`:
  - Otra información sobre un trabajo que una integración utiliza para completar el trabajo.
  - Contiene un mapa de lo siguiente:
    - `actionConfiguration`:
      - Los datos de configuración de la acción. Los campos de configuración de la acción son una asignación de pares clave-valor para que el cliente introduzca valores. Las claves vienen determinadas por los parámetros clave del archivo de definición del tipo de acción al configurar la acción. En este ejemplo, los valores los determina el usuario de la acción especificando la información en los campos `Username` y `Password`.
      - Tipo: asignación de cadena a cadena, presente de forma opcional

Ejemplo:

```
"configuration": {
    "Username": "MyUser",
    "Password": "MyPassword"
},
```

- `encryptionKey`:
  - Representa información sobre la clave utilizada para cifrar los datos del almacén de artefactos, como una AWS KMS clave.
  - Contenido: tipo de tipo de datos `encryptionKey`, disponible opcionalmente
- `inputArtifacts`:
  - Lista de información sobre un artefacto en el que se va a trabajar, como, por ejemplo, una prueba o un artefacto de compilación.
  - Contenido: lista de tipo de datos `Artifact`, disponible opcionalmente
- `outputArtifacts`:
  - Lista de información sobre el resultado de una acción.
  - Contenido: lista de tipo de datos `Artifact`, disponible opcionalmente
- `actionCredentials`:
  - Representa un objeto de credenciales AWS de sesión. Estas credenciales son credenciales temporales emitidas por AWS STS. Se pueden usar para acceder a los artefactos de entrada y salida del bucket de S3, que se usa para almacenar artefactos para la canalización CodePipeline.

    Estas credenciales también tienen los mismos permisos que la plantilla de declaraciones de política especificada en el archivo de definición del tipo de acción.
  - Contenido: tipo de tipo de datos `AWSSessionCredentials`, disponible opcionalmente
- `actionExecutionId`:
  - El ID externo de la ejecución de la acción.
  - Tipo: cadena
- `continuationToken`:
  - Un token generado por el sistema, como un ID de implementación, que necesita un trabajo para continuar con el trabajo de forma asíncrona.

    - Tipo: cadena, presente de forma opcional

Tipos de datos:

- `encryptionKey`:
  - `id`:
    - El ID utilizado para identificar la clave. Para una AWS KMS clave, puedes usar el identificador de la clave, el ARN de la clave o el ARN del alias.
    - Tipo: cadena
  - `type`:
    - El tipo de clave de cifrado, como una AWS KMS clave.
    - Tipo: cadena
    - Valores válidos: `KMS`
- `Artifact`:
  - `name`:
    - El nombre del artefacto.
    - Tipo: cadena, presente de forma opcional
  - `revision`:
    - El ID de revisión del artefacto. Según el tipo de objeto, puede ser un ID de confirmación (GitHub) o un ID de revisión (Amazon S3).
    - Tipo: cadena, presente de forma opcional
  - `location`:
    - La ubicación de un artefacto.
    - Contenido: tipo de tipo de datos `ArtifactLocation`, disponible opcionalmente
- `ArtifactLocation`:
  - `type`:
    - El tipo de artefacto en la ubicación.
    - Tipo: cadena, presente de forma opcional
    - Valores válidos: `S3`
  - `s3Location`:
    - Ubicación del bucket de S3 que contiene una revisión.

    - Contenido: tipo de tipo de datos `S3Location`, disponible opcionalmente
- `S3Location`:
  - `bucketName`:
    - Nombre del bucket de S3.
    - Tipo: cadena
  - `objectKey`:
    - La clave del objeto en el bucket de S3 que identifica de forma exclusiva el objeto en el bucket.
    - Tipo: cadena
- `AWSSessionCredentials`:
  - `accessKeyId`:
    - La clave de acceso secreta de la sesión.
    - Tipo: cadena
  - `secretAccessKey`:
    - La clave de acceso secreta de la sesión.
    - Tipo: cadena
  - `sessionToken`:
    - Ll token de la sesión.
    - Tipo: cadena

Ejemplo:

```
{
    "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"
    }
}
```

## Devuelve los resultados de tu función de Lambda a CodePipeline

El recurso del proceso de trabajo del integrador debe devolver una carga útil válida en caso de éxito, fracaso o continuación.

Formato:

- `result`: el resultado del trabajo.
  - Obligatorio
  - Valores válidos (no distingue entre mayúsculas y minúsculas):

  - `Success`: indica que un trabajo se ha realizado correctamente y es terminal.
  - `Continue`: indica que un trabajo se ha realizado correctamente y debe continuar, por ejemplo, si se vuelve a invocar al trabajador del trabajo para ejecutar la misma acción.
  - `Fail`: indica que se ha producido un error en un trabajo y es terminal.
- `failureType`: un tipo de fallo que se va a asociar a un trabajo fallido.

  La categoría `failureType` de acciones de los socios describe el tipo de error que se produjo al ejecutar el trabajo. Los integradores configuran el tipo junto con el mensaje de error cuando devuelven el resultado de un error de trabajo a. CodePipeline
  - Opcional. Obligatorio si el resultado es `Fail`.
  - Debe ser nulo si `result` es `Success` o `Continue`
  - Valores válidos:
    - ConfigurationError
    - JobFailed
    - PermissionsError
    - RevisionOutOfSync
    - RevisionUnavailable
    - SystemUnavailable
- `continuation`: el estado de continuación se pasará al siguiente trabajo dentro de la ejecución de la acción actual.
  - Opcional. Obligatorio si el resultado es `Continue`.
  - Debe ser nulo si `result` es `Success` o `Fail`.
  - Propiedades:
    - `State`: un hash del estado que se va a aprobar.
- `status`: estado de la ejecución de la acción.
  - Opcional.
  - Propiedades:
    - `ExternalExecutionId`: un ID de ejecución externo o un ID de confirmación opcional para asociarlo al trabajo.
    - `Summary`: un resumen opcional de lo ocurrido. En los escenarios de error, se convierte en el mensaje de error que ve el usuario.

- `outputVariables`: conjunto de key/value pares que se pasarán a la siguiente ejecución de la acción.
  - Opcional.
  - Debe ser nulo si `result` es `Continue` o `Fail`.

Ejemplo:

```
{
    "result": "success",
    "failureType": null,
    "continuation": null,
    "status": {
        "externalExecutionId": "my-commit-id-123",
        "summary": "everything is dandy"
    },
    "outputVariables": {
        "FirstOne": "Nice",
        "SecondOne": "Nicest",
        ...
    }
}
```

## Utilice los tokens de continuación para esperar los resultados de un proceso asíncrono

El token `continuation` forma parte de la carga útil y el resultado de la función de Lambda. Es una forma de pasar el estado del trabajo CodePipeline e indicar que el trabajo debe continuar. Por ejemplo, cuando un integrador inicia una compilación para el cliente en su recurso, no espera a que se complete la compilación, sino que indica CodePipeline que no tiene un resultado final devolviendo el `result` as `continue` y devolviendo el identificador único de la compilación al `continuation` token CodePipeline as.

> **Note**
>
> Las funciones de Lambda se pueden ejecutar durante un máximo de 15 minutos. Si el trabajo necesita ejecutarse durante más tiempo, puedes usa los tokens de continuación.

El CodePipeline equipo invoca al integrador después de 30 segundos con el mismo `continuation` token en su carga útil para comprobar si está completo. Si la compilación se completa, el integrador devuelve el success/fail resultado final; de lo contrario, continúa.

## Proporcione CodePipeline los permisos para invocar la función Lambda del integrador en tiempo de ejecución

Agrega permisos a la función Lambda de su integrador para proporcionar al CodePipeline servicio permisos para invocarlo mediante el principal del servicio:. CodePipeline `codepipeline.amazonaws.com` Puede agregar permisos mediante CloudFormation o la línea de comandos. Para ver un ejemplo, consulta Trabajar con tipos de acciones.

# Modelo de integración de procesos de trabajo

Una vez designado el flujo general de trabajo, puede crear el proceso de trabajo. Aunque los detalles específicos de la acción de terceros determinan lo que se necesita para el proceso de trabajo, la mayoría de los procesos de trabajo para acciones de terceros incluyen la siguiente funcionalidad:

- Encuesta para ver si hay trabajos en CodePipeline usoPollForThirdPartyJobs.
- Reconocer los trabajos y devolver los resultados al CodePipeline uso `AcknowledgeThirdPartyJobPutThirdPartyJobSuccessResult`,, `yPutThirdPartyJobFailureResult`.
- Recuperar artefactos and/or al colocarlos en el bucket de Amazon S3 para la canalización. Para descargar artefactos del bucket de Amazon S3 debe crear un cliente de Amazon S3 que utilice la versión 4 de Signature (Sig V4). Se requiere el Sig V4 para AWS KMS.

  Para subir artefactos al bucket de Amazon S3, también debe configurar la `PutObject` solicitud de Amazon S3 para utilizar el cifrado mediante AWS Key Management Service (AWS KMS). AWS KMS usos AWS KMS keys. Para saber si se debe utilizar la clave Clave administrada de AWS o una clave gestionada por el cliente para cargar artefactos, el trabajador debe examinar los datos de la tarea y comprobar la propiedad de la clave de cifrado. Si la propiedad está definida, debes usar esa ID de clave gestionada por el cliente al realizar la configuración AWS KMS. Si la propiedad de la clave es nula, utilice la Clave administrada de AWS. CodePipeline usa el, Clave administrada de AWS a menos que se configure lo contrario.

  Para ver un ejemplo que muestre cómo crear los AWS KMS parámetros en Java o.NET, consulte Especificar los AWS Key Management Service en Amazon S3 mediante los AWS SDK. Para

obtener más información sobre el bucket de Amazon S3 para CodePipeline, consulteCodePipeline conceptos.

## Elegir y configurar una estrategia de administración de permisos para el proceso de trabajo

Si quiere crear un empleado para su acción de terceros CodePipeline, necesita una estrategia que integre la administración de usuarios y permisos.

La estrategia más sencilla consiste en añadir la infraestructura que necesita para su trabajador mediante la creación de instancias de Amazon EC2 con un rol de instancia AWS Identity and Access Management (IAM), lo que le permitirá ampliar fácilmente los recursos que necesita para la integración. Puede utilizar la integración integrada AWS para simplificar la interacción entre su trabajador y. CodePipeline

Obtenga más información acerca de Amazon EC2 y determine si es la opción correcta para su integración. Para más información, consulte Amazon EC2 - Alojamiento de servidores virtuales. Para obtener información sobre cómo configurar una instancia de Amazon EC2, consulte Introducción a las instancias de Amazon EC2 Linux.

Otra estrategia que puede considerar es utilizar federación de identidades con IAM para integrar los recursos y el sistema de proveedor de identidad que ya tiene. Esta estrategia resulta útil si ya tiene un proveedor de identidad corporativo o una configuración que admita usuarios que utilicen proveedores de identidad web. La federación de identidades le permite conceder un acceso seguro a AWS los recursos CodePipeline, incluso sin tener que crear o administrar usuarios de IAM. Puede utilizar características y políticas sobre los requisitos de seguridad de contraseñas y la rotación de credenciales. Puede utilizar aplicaciones de ejemplo como plantillas para su propio diseño. Para obtener información, consulte Administrar federación.

Para dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en AWS IAM Identity Center:

  Cree un conjunto de permisos. Siga las instrucciones de Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center .
- Usuarios gestionados en IAM a través de un proveedor de identidades:

  Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.

- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o agregue un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

A continuación se muestra una política de ejemplo que puede crear para utilizarla con un proceso de trabajo de terceros. Esta política es solo un ejemplo y se ofrece "tal cual".

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codepipeline:PollForThirdPartyJobs",
        "codepipeline:AcknowledgeThirdPartyJob",
        "codepipeline:GetThirdPartyJobDetails",
        "codepipeline:PutThirdPartyJobSuccessResult",
        "codepipeline:PutThirdPartyJobFailureResult"
      ],
      "Resource": [
        "arn:aws:codepipeline:us-east-2::actionType:ThirdParty/Build/Provider/1/"

      ]
    }
  ]
}
```

# Referencia del archivo de definiciones de imágenes

Esta sección se incluye solo como referencia. Para obtener más información sobre cómo crear una canalización con acciones de código fuente o de implementación para contenedores, consulte Creación de una canalización, etapas y acciones.

AWS CodePipeline Los trabajadores que realizan acciones de contenedor, como una acción de origen de Amazon ECR o acciones de implementación de Amazon ECS, utilizan archivos de definiciones para asignar el URI de la imagen y el nombre del contenedor a la definición de la tarea. Cada archivo de definiciones es un JSON-formatted archivo que utiliza el proveedor de la acción de la siguiente manera:

- Las implementaciones estándar de Amazon ECS requieren un archivo `imagedefinitions.json` como entrada para la acción de implementación. Para ver un tutorial que utiliza la acción de implementación estándar de Amazon ECS en CodePipeline, consulteTutorial: implementación estándar de Amazon ECS con CodePipeline. Para ver otro ejemplo de tutorial en el que se utiliza la acción de implementación estándar de Amazon ECS CodePipeline junto con la ECRBuildAndPublish acción, consulteTutorial: Cree y envíe una imagen de Docker a Amazon ECR con CodePipeline (tipo V2).
- blue/green Las implementaciones de Amazon ECS requieren un `imageDetail.json` archivo como entrada para la acción de implementación. Para ver un tutorial con un ejemplo de blue/green implementación, consulteTutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR.
  - Las acciones de código fuente de Amazon ECR generan un archivo `imageDetail.json` que se suministra como salida de la acción de código fuente.

Temas

- Archivo imagedefinitions.json para las acciones de implementación estándar de
- Detail.json archivo de imagen para las acciones de blue/green implementación de Amazon ECS

## Archivo imagedefinitions.json para las acciones de implementación estándar de

Un documento de definiciones de imágenes es un archivo JSON que describe el nombre, la imagen y la etiqueta del contenedor del servicio de Amazon ECS. Si está implementando aplicaciones

basadas en contenedores, debe generar un archivo de definiciones de imágenes para proporcionar al trabajador el contenedor CodePipeline de Amazon ECS y la identificación de la imagen para recuperarlos del repositorio de imágenes, como Amazon ECR.

> **Note**
>
> El nombre predeterminado del archivo es `imagedefinitions.json`. Si utiliza otro nombre de archivo, debe proporcionarlo al crear la etapa de implementación de la canalización.

Cree el archivo `imagedefinitions.json` teniendo en cuenta lo siguiente:

- El archivo debe usar codificación. UTF-8
- El límite de tamaño máximo de archivo para el archivo de definiciones de imágenes es de 100 KB.
- Debe crear el archivo como un artefacto de código fuente o de compilación para que sea un artefacto de entrada para la acción de implementación. En otras palabras, asegúrate de que el archivo esté cargado en tu ubicación de origen, como tu CodeCommit repositorio, o de que se genere como un artefacto de salida integrado.

El archivo `imagedefinitions.json` proporciona el nombre del contenedor y el URI de la imagen. Debe crearse con el siguiente conjunto de pares clave-valor.

| Clave | Valor |
| --- | --- |
| name | *container_name* |
| imageURI | *imageUri* |

> **Note**
>
> El campo de nombre se usa para el nombre de la imagen del contenedor, es decir, el nombre de la imagen de Docker.

Esta es la estructura JSON, donde el nombre del contenedor es `sample-app`, la URI de la imagen es `ecs-repo` y la etiqueta es `latest`:

```
[
  {
    "name": "sample-app",
    "imageUri": "11111EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest"
  }
]
```

También puede construir el archivo para que incluya varios pares contenedor-imagen.

Estructura JSON:

```
[
  {
    "name": "simple-app",
    "imageUri": "httpd:2.4"
  },
  {
    "name": "simple-app-1",
    "imageUri": "mysql"
  },
  {
    "name": "simple-app-2",
    "imageUri": "java1.8"
  }
]
```

Antes de crear la canalización, siga los pasos que se indican a continuación para configurar el archivo `imagedefinitions.json`.

1. Como parte de la planificación de la implementación de aplicaciones basadas en contenedores para la canalización, planifique la etapa de código fuente y la etapa de compilación, si procede.
2. Seleccione una de las siguientes opciones:

    a. Si la canalización se crea de forma que salta la etapa de compilación, debe crear manualmente el archivo JSON y cargarlo en el repositorio de origen a fin de que la acción de origen pueda proporcionar el artefacto. Cree el archivo con un editor de texto y asígnele un nombre, o utilice el nombre de archivo `imagedefinitions.json` predeterminado. Envíe el archivo de definiciones de imágenes al repositorio de código fuente.

> **Note**
>
> Recuerde comprimir el archivo JSON si el repositorio de origen es un bucket de .

b. Si tu canalización tiene una fase de compilación, agrega un comando a tu archivo buildspec que genere el archivo de definiciones de imágenes en tu repositorio de origen durante la fase de compilación. En el siguiente ejemplo, se utiliza el comando printf para crear un archivo `imagedefinitions.json`. Incluya este comando en la sección `post_build` del archivo `buildspec.yml`:

```
printf '[{"name":"container_name","imageUri":"image_URI"}]' >
imagedefinitions.json
```

Debe incluir el archivo de definiciones de imágenes como artefacto de salida en el archivo `buildspec.yml`.

3. Al crear la canalización en la consola, introduzca el nombre del archivo de definiciones de imágenes en el campo Image Filename (Nombre del archivo de imágenes) de la página Deploy (Implementación) del asistente Create Pipeline (Crear canalización).

Para ver un tutorial paso a paso sobre cómo crear una canalización que utilice Amazon ECS como proveedor de implementación, consulte Tutorial: Continuous Deployment with. CodePipeline

## Detail.json archivo de imagen para las acciones de blue/green implementación de Amazon ECS

Un documento `imageDetail.json` es un archivo JSON que describe el URI de una imagen de Amazon ECS. Si va a implementar aplicaciones basadas en contenedores para una blue/green implementación, debe generar el `imageDetail.json` archivo para proporcionar a Amazon ECS y al trabajador CodeDeploy laboral la identificación de la imagen para que la recupere del repositorio de imágenes, como Amazon ECR.

> **Note**
>
> El nombre del archivo debe ser `imageDetail.json`.

Para ver una descripción de la acción y el resto de parámetros, consulte Referencia de acciones de implementación de Amazon Elastic Container Service y CodeDeploy azul-verde.

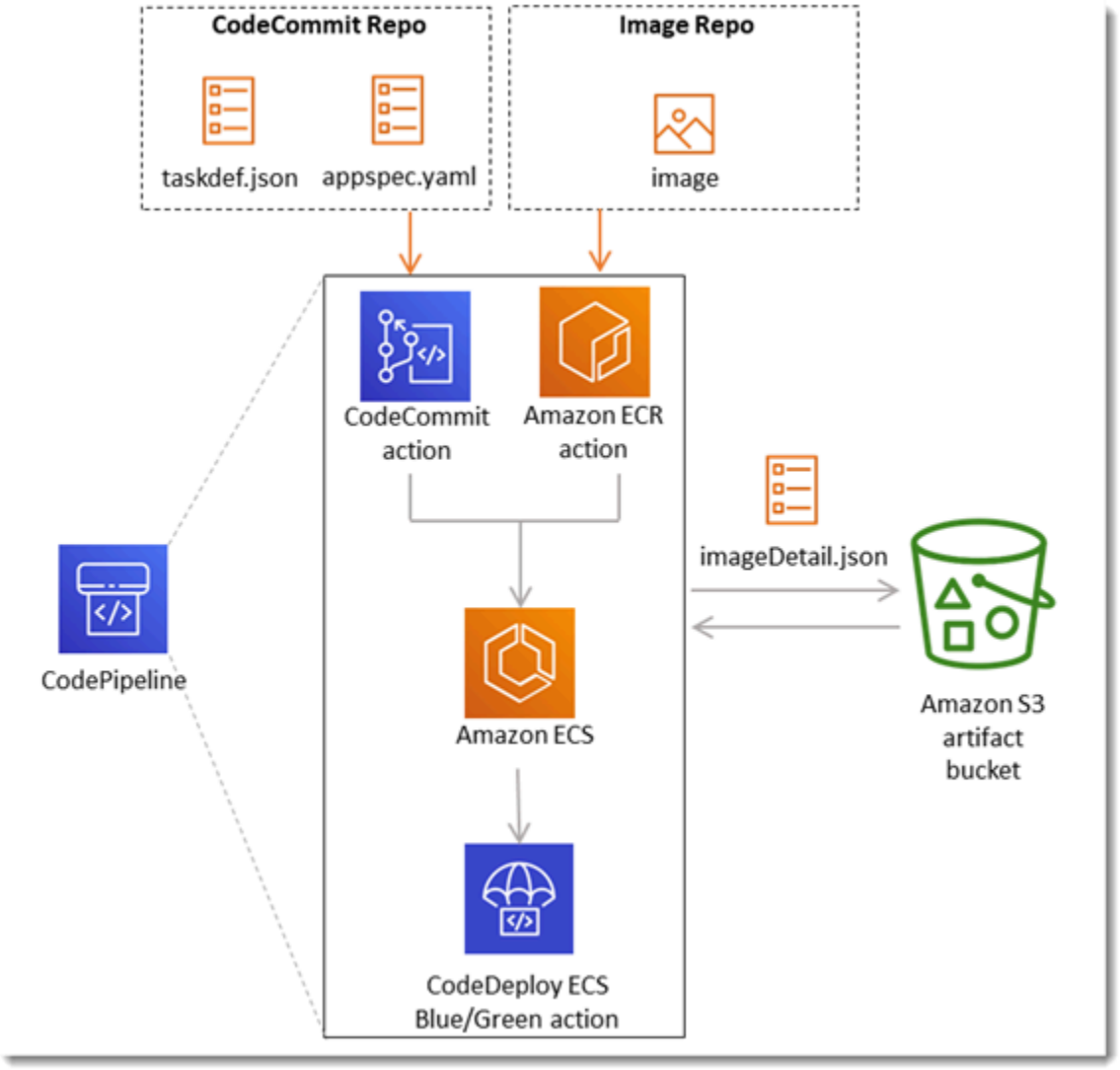
Debe crear el archivo `imageDetail.json` como un artefacto de código fuente o de compilación para que sea un artefacto de entrada para la acción de implementación. Puede utilizar uno de estos métodos para proporcionar el archivo `imageDetail.json` en la canalización:

- Incluya el `imageDetail.json` archivo en su ubicación de origen para que se proporcione en proceso como entrada para la acción de implementación de Amazon ECS blue/green .

  > **Note**
  >
  > Recuerde comprimir el archivo JSON si el repositorio de origen es un bucket de .

- Las acciones de origen de Amazon ECR generan automáticamente un archivo `imageDetail.json` como artefacto de entrada para la acción siguiente.

  > **Note**
  >
  > Dado que la acción de origen de Amazon ECR crea este archivo, no es necesario que las canalizaciones con una acción de origen de Amazon ECR proporcionen de forma manual un archivo `imageDetail.json`.
  > Para obtener un tutorial sobre la creación de una canalización que incluya una etapa de origen de Amazon ECR, consulte Tutorial: Creación de una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR.




El archivo `imageDetail.json` proporciona el URI de la imagen. Debe crearse con el siguiente par clave-valor.

| Clave | Valor |
| --- | --- |
| ImageURI | *image_URI* |

imageDetail.json

A continuación, se muestra la estructura JSON, donde el URI de la imagen es `ACCOUNTID.dkr.ecr.us-west-2.amazonaws.com/dk-image-repo@sha256:example3`:

```
{
```

```
"ImageURI": "ACCOUNTID.dkr.ecr.us-west-2.amazonaws.com/dk-image-
repo@sha256:example3"
}
```

imageDetail.json (generated by ECR)

La acción de origen de Amazon ECR genera automáticamente un archivo `imageDetail.json` cada vez que se envía un cambio al repositorio de imágenes. El archivo `imageDetail.json` generado por las acciones de origen de Amazon ECR se suministra como artefacto de salida de la acción de origen a la siguiente acción de la canalización.

A continuación, se muestra la estructura JSON, donde el nombre del repositorio es `dk-image-repo`, el URI de la imagen es `ecs-repo` y la etiqueta de la imagen es `latest`:

```
{
    "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"
    ]
}
```

El archivo `imageDetail.json` asigna el URI de la imagen y nombre del contenedor a la definición de tareas de Amazon ECS, tal y como se indica a continuación:

- `ImageSizeInBytes`: el tamaño, en bytes, de la imagen en el repositorio.
- `ImageDigest`: el resumen `sha256` del manifiesto de la imagen.
- `Version`: la versión de la imagen.
- `ImagePushedAt`: la fecha y la hora en que se envió la imagen más reciente al repositorio.
- `RegistryId`: el identificador de AWS cuenta asociado al registro que contiene el repositorio.
- `RepositoryName`: el nombre del repositorio de Amazon ECR al que se envió la imagen.
- `ImageURI`: el URI de la imagen.

- `ImageTags`: la etiqueta utilizada para la imagen.

Antes de crear la canalización, siga los pasos que se indican a continuación para configurar el archivo `imageDetail.json`.

1. Como parte de la planificación del blue/green despliegue de aplicaciones basadas en contenedores para tu canalización, planifica la fase de origen y la fase de creación, si procede.
2. Seleccione una de las siguientes opciones:

   a. Si tu canalización ha omitido la fase de creación, debes crear manualmente el archivo JSON y subirlo a tu repositorio de origen, por ejemplo, para que la acción de origen pueda proporcionar el artefacto. CodeCommit Cree el archivo con un editor de texto y asígnele un nombre, o utilice el nombre de archivo `imageDetail.json` predeterminado. Envíe el `imageDetail.json` archivo al repositorio de código fuente.

   b. Si la canalización tiene una etapa de compilación, haga lo siguiente:

      i. Agrega un comando a tu archivo de especificaciones de construcción que genere el archivo de definiciones de imágenes en tu repositorio de origen durante la fase de compilación. En el siguiente ejemplo, se utiliza el comando printf para crear un archivo `imageDetail.json`. Incluya este comando en la sección `post_build` del archivo buildspec.yml:

      ```
      printf '{"ImageURI":"image_URI"}' > imageDetail.json
      ```

      Debe incluir el archivo `imageDetail.json` como artefacto de salida en el archivo `buildspec.yml`.

      ii. Añada el archivo `imageDetail.json` como artefacto en el archivo `buildspec.yml`.

      ```
      artifacts:
        files:
          - imageDetail.json
      ```

# Referencia de variables

Esta sección se incluye solo como referencia. Para obtener información sobre la creación de variables, consulte Trabajar con variables.

Las variables le permiten configurar las acciones de canalización con valores que se determinan en el momento de la ejecución de la canalización o de la ejecución de la acción.

Algunos proveedores de acciones producen un conjunto definido de variables. Puede elegir entre las claves de variables predeterminadas para ese proveedor de acciones, como el ID de confirmación.

> **Important**
>
> Al pasar parámetros del secreto, no introduzca el valor directamente. El valor se representa como texto no cifrado y, por lo tanto, se puede leer. Por razones de seguridad, no utilice texto sin formato con secretos. Le recomendamos encarecidamente que lo utilice AWS Secrets Manager para almacenar secretos.

Para consultar ejemplos paso a paso del uso de variables:

- Para ver un tutorial con una variable a nivel de canalización que se transfiere en el momento de la ejecución de la canalización, consulte Tutorial: Uso de variables a nivel de canalización.
- Para ver un tutorial con una acción de Lambda que usa variables de una acción ascendente (CodeCommit) y genera variables de salida, consulte. Tutorial: Uso de variables con acciones de invocación de Lambda
- Para ver un tutorial con una CloudFormation acción que hace referencia a variables de salida de pila desde una CloudFormation acción ascendente, consulte. Tutorial: Cree una canalización que utilice variables de AWS CloudFormation acciones de despliegue
- Para ver un ejemplo de una acción de aprobación manual con el texto del mensaje que hace referencia a variables de salida que se resuelven en el ID de CodeCommit confirmación y el mensaje de confirmación, consulteEjemplo: Usar variables en aprobaciones manuales.
- Para ver un ejemplo de CodeBuild acción con una variable de entorno que se resuelve con el nombre de la GitHub sucursal, consulteEjemplo: usa una BranchName variable con variables de CodeBuild entorno.

- CodeBuild las acciones producen como variables todas las variables de entorno que se exportaron como parte de la compilación. Para obtener más información, consulte CodeBuild variables de salida de la acción.

Límites de variables

Para obtener información sobre los límites, consulte Cuotas en AWS CodePipeline.

> **Note**
>
> Cuando escriba la sintaxis de variable en los campos de configuración de acciones, no exceda el límite de 1000 caracteres de los campos de configuración. Cuando se supera este límite, se devuelve un error de validación.

Temas


- Conceptos
- Casos de uso de variables
- Configuración de variables
- Resolución de variables
- Reglas para variables
- Variables disponibles para acciones de canalización


# Conceptos

En esta sección se enumeran los términos y conceptos clave relacionados con variables y espacios de nombres.

## Variables

Las variables son pares clave-valor que se pueden utilizar para configurar dinámicamente acciones en la canalización. Actualmente hay tres maneras de hacer que estas variables estén disponibles:

- Hay un conjunto de variables que están implícitamente disponibles al inicio de cada ejecución de canalización. Este conjunto incluye actualmente `PipelineExecutionId`, el ID de la ejecución de la canalización actual.

- Las variables a nivel de canalización se definen cuando la canalización se crea y se resuelven en el tiempo de ejecución de la canalización.

  Las variables a nivel de canalización se especifican cuando se crea la canalización y se pueden proporcionar valores en el momento de la ejecución de la canalización.
- Existen tipos de acción que producen conjuntos de variables cuando se ejecutan. Para ver las variables generadas por una acción, inspecciona el `outputVariables` campo que forma parte de la ListActionExecutions API. Para obtener una lista de los nombres de clave disponibles por proveedor de acciones, consulte Variables disponibles para acciones de canalización. Para ver qué variables produce cada tipo de acción, consulta la CodePipeline Referencia de la estructura de acciones.

Para hacer referencia a estas variables en la configuración de acción, debe utilizar la sintaxis de referencia de variable con el espacio de nombres correcto.

Para ver un flujo de trabajo de variables de ejemplo, consulte Configuración de variables.

## Espacios de nombres

Para garantizar que se pueda hacer referencia a las variables de forma única, deben asignarse a un espacio de nombres. Después de tener un conjunto de variables asignado a un espacio de nombres, se puede hacer referencia a las mismas en una configuración de acción mediante el espacio de nombres y la clave de variable con la siguiente sintaxis:

```
#{namespace.variable_key}
```

Hay tres tipos de espacios de nombres en los que se pueden asignar variables:

- El espacio de nombres reservado codepipeline

  Este es el espacio de nombres asignado al conjunto de variables implícitas disponibles al inicio de cada ejecución de canalización. Este espacio de nombres es `codepipeline`. Ejemplo de referencia de variable:

  ```
  #{codepipeline.PipelineExecutionId}
  ```
- El espacio de nombres de las variables a nivel de canalización

Este es el espacio de nombres asignado a variables a nivel de canalización. El espacio de nombres de las variables a nivel de canalización es `variables`. Ejemplo de referencia de variable:

```
#{variables.variable_name}
```

- Espacio de nombres asignado a la acción

  Se trata de un espacio de nombres que se asigna a una acción. Todas las variables producidas por la acción caen bajo este espacio de nombres. Para que las variables producidas por una acción estén disponibles para su uso en una configuración de acción descendente, debe configurar la acción de producción con un espacio de nombres. Los espacios de nombres deben ser únicos en toda la definición de canalización y no pueden entrar en conflicto con ningún nombre de artefacto. Aquí hay una referencia de variable de ejemplo para una acción configurada con un espacio de nombres de `SourceVariables`.

  ```
  #{SourceVariables.VersionId}
  ```

# Casos de uso de variables

A continuación, se incluyen algunos de los casos de uso más comunes de variables a nivel de canalización y le ayuda a determinar cómo puede usar las variables para sus necesidades específicas.

- Las variables a nivel de canalización son para CodePipeline los clientes que desean usar la misma canalización cada vez con pequeñas variaciones en las entradas de la configuración de la acción. Cualquier desarrollador que inicie una canalización añade el valor de la variable en la interfaz de usuario cuando se inicia la canalización. Con esta configuración, solo se transfieren parámetros para esa ejecución.
- Con las variables a nivel de canalización, puede pasar entradas dinámicas a las acciones de la canalización. Puedes migrar tus canalizaciones parametrizadas CodePipeline sin tener que mantener diferentes versiones de la misma canalización ni crear canalizaciones complejas.
- Puede usar variables a nivel de canalización para transferir parámetros de entrada que le permitan reutilizar una canalización en cada ejecución, por ejemplo, cuando quiere especificar qué versión quiere implementar en un entorno de producción, de forma que no tenga que duplicar las canalizaciones.

- Puede usar una única canalización para implementar recursos en varios entornos de creación e implementación. Por ejemplo, en el caso de una canalización con un CodeCommit repositorio, la implementación desde una sucursal y un entorno de implementación de destino específicos se puede realizar con CodeBuild CodeDeploy los parámetros transmitidos a nivel de canalización.

> **Note**
>
> En el caso de Amazon ECR, Amazon S3 o CodeCommit las fuentes, también puedes crear una anulación de fuente utilizando la entrada de transformación de entrada EventBridge para usar la entrada `revisionValue` en el evento de canalización, donde `revisionValue` se deriva de la variable de evento de origen de tu clave de objeto, confirmación o ID de imagen. Para obtener más información, consulte el paso opcional para la entrada de transformación de entrada, que se incluye en los procedimientos de Acciones y recursos de origen de Amazon ECR EventBridge, Conexión a acciones de origen de Amazon S3 con un origen habilitado para eventos o CodeCommit acciones de origen y EventBridge.

# Configuración de variables

Puede configurar las variables a nivel de canalización o a nivel de acción en la estructura de canalización.

## Configuración de variables a nivel de canalización

Puede agregar una o varias variables a nivel de canalización. Puede hacer referencia a este valor en la configuración de CodePipeline las acciones. Puede añadir los nombres de las variables, los valores predeterminados y las descripciones al crear la canalización. Las variables se resuelven en el momento de la ejecución.

> **Note**
>
> Si no se define un valor predeterminado para una variable a nivel de canalización, la variable se considera obligatoria. Debe especificar las anulaciones para todas las variables obligatorias al iniciar una canalización; de lo contrario, la ejecución de la canalización fallará y se producirá un error de validación.

Las variables se proporcionan a nivel de canalización mediante el atributo de variables de la estructura de canalización. En el siguiente ejemplo, la variable `Variable1` tiene un valor de `Value1`.

```
      "variables": [
           {
               "name": "Variable1",
               "defaultValue": "Value1",
               "description": "description"
           }
       ]
```

Para ver un ejemplo en la estructura del JSON de la canalización, consulte Creación de una canalización, etapas y acciones.

Para ver un tutorial con una variable a nivel de canalización que se transfiere en el momento de la ejecución de la canalización, consulte Tutorial: Uso de variables a nivel de canalización.

Tenga en cuenta que no se admite el uso de variables a nivel de canalización en ningún tipo de acción de origen.

> **Note**
>
> Si el espacio de nombres `variables` ya se utiliza en algunas de las acciones del proceso, debe actualizar la definición de la acción y elegir otro espacio de nombres para la acción conflictiva.

## Configuración de variables a nivel de acción

Una acción se configura para generar variables declarando un espacio de nombres para la acción. La acción ya debe ser uno de los proveedores de acciones que generan variables. De lo contrario, las variables disponibles son variables de nivel de canalización.

El espacio de nombres se declara mediante:

- En la páginaEdit action (Editar acción) de la consola, introduciendo un espacio de nombres en Variable namespace (Espacio de nombres de variable).
- Introduciendo un espacio de nombres en el campo de parámetros `namespace` en la estructura de canalización JSON.

En este ejemplo, se agrega el `namespace` parámetro a la acción de CodeCommit origen con el nombre`SourceVariables`. Esto configura la acción para producir las variables disponibles para ese proveedor de acciones, como `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
        }
    ]
},
```

A continuación, configure la acción descendente para utilizar las variables producidas por la acción anterior. Esto se hace del siguiente modo:

- En la página Edit action (Editar acción) de la consola, introduciendo la sintaxis de variable (para la acción descendente) en los campos de configuración de acción.
- Introduciendo la sintaxis de variable (para la acción descendente) en los campos de configuración de acciones en la estructura de canalización JSON

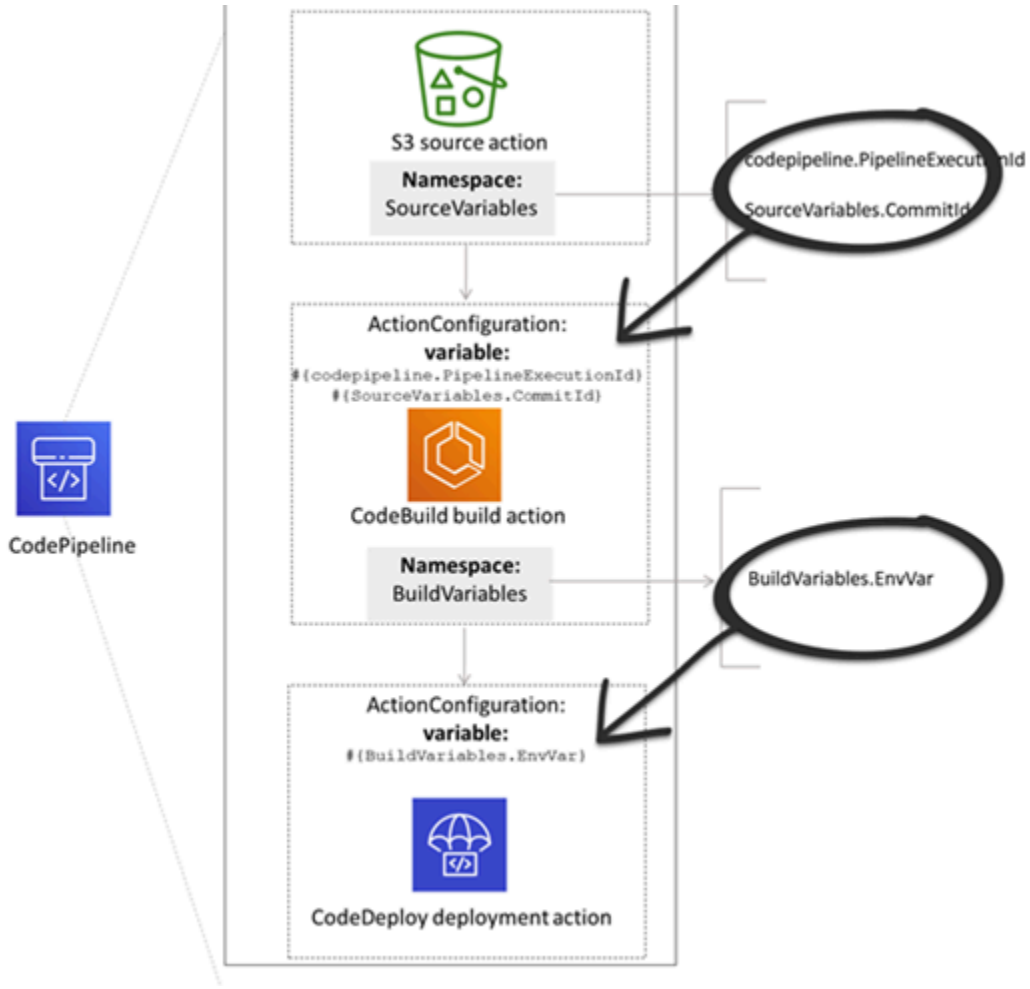
En este ejemplo, el campo de configuración de la acción de compilación muestra variables de entorno que se actualizan en la ejecución de la acción. El ejemplo especifica el espacio de nombres y la variable para el ID de ejecución con `#{codepipeline.PipelineExecutionId}` y el espacio de nombres y la variable para el ID de confirmación con **`#{SourceVariables.CommitId}`**.

```
{
    "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
        }
    ]
},
```

# Resolución de variables

Cada vez que se ejecuta una acción como parte de una ejecución de canalización, las variables que produce están disponibles para su uso en cualquier acción que se garantice que ocurra después de

la acción de producción. Para utilizar estas variables en una acción de consumo, puede agregarlas a la configuración de la acción de consumo utilizando la sintaxis mostrada en el ejemplo anterior. Antes de realizar una acción agotadora, CodePipeline resuelve todas las referencias a variables presentes en la configuración antes de iniciar la ejecución de la acción.




# Reglas para variables

Las siguientes reglas le ayudan con la configuración de variables:

- Especifique el espacio de nombres y la variable de una acción a través de una nueva propiedad de acción o editando una acción.
- Cuando se utiliza el asistente de creación de canalizaciones, la consola genera un espacio de nombres para cada acción creada con el asistente.
- Si no se especifica el espacio de nombres, no se puede hacer referencia a las variables producidas por esa acción en ninguna configuración de acción.

- Para hacer referencia a variables producidas por una acción, la acción de referencia debe producirse después de la acción que produce las variables. Esto significa que está en una etapa posterior a la acción que produce las variable o en la misma etapa pero en un orden de ejecución más alto.

# Variables disponibles para acciones de canalización

El proveedor de la acción determina qué variables puede generar la acción.

Para ver los procedimientos paso a paso para administrar variables, consulte Trabajar con variables.

## Acciones con claves de variables definidas

A diferencia de un espacio de nombres que puede elegir, las siguientes acciones utilizan claves de variables que no se pueden editar. Por ejemplo, para el proveedor de acciones de Amazon S3, solo están disponibles las claves de variables `ETag` y `VersionId`.

Cada ejecución también tiene un conjunto de variables de CodePipeline-generated canalización que contienen datos sobre la ejecución, como el identificador de versión de la canalización. Estas variables las puede consumir cualquier acción de la canalización.

Temas

- CodePipeline variable de ID de ejecución
- Variables de salida de acciones de Amazon ECR
- CloudFormation StackSets variables de salida de acción
- CodeCommit variables de salida de la acción
- CodeStarSourceConnection variables de salida de acción
- GitHub variables de salida de acción GitHub (a través de la aplicación OAuth) action)
- Variables de salida de acciones de S3

## CodePipeline variable de ID de ejecución

CodePipeline variable de ID de ejecución

| Proveedor | Clave de variable | Ejemplo de valor | Ejemplo de sintaxis de variables |
|---|---|---|---|
| codepipeline | `PipelineExecutionId` | 8abc75f0-fbf8-4f4c-bfEXAMPLE | `#{codepipeline.PipelineExecutionId}` |

## Variables de salida de acciones de Amazon ECR

Variables de Amazon ECR

| Clave de variable | Ejemplo de valor | Ejemplo de sintaxis de variables |
|---|---|---|
| `ImageDigest` | sha256:EXAMPLE1122334455 | `#{SourceVariables.ImageDigest}` |
| `ImageTag` | más reciente | `#{SourceVariables.ImageTag}` |
| `ImageURI` | 11111 EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest | `#{SourceVariables.ImageURI}` |
| `RegistryId` | EXAMPLE12233 | `#{SourceVariables.RegistryId}` |
| `RepositoryName` | my-image-repo | `#{SourceVariables.RepositoryName}` |

## CloudFormation StackSets variables de salida de acción

CloudFormation StackSets variables

| Clave de variable | Ejemplo de valor | Ejemplo de sintaxis de variables |
| --- | --- | --- |
| `OperationId` | 11111111-2bbb-111-2bbb-1111 1example | `#{DeployVariables. OperationId}` |
| `StackSetId` | my-stackset:1111aaaa-1111-2 222-2bbb-11111example | `#{DeployVariables. StackSetId}` |

## CodeCommit variables de salida de la acción

CodeCommit variables

| Clave de variable | Ejemplo de valor | Ejemplo de sintaxis de variables |
| --- | --- | --- |
| `AuthorDate` | 2019-10-29 T03:32:21Z | `#{SourceVariables. AuthorDate}` |
| `BranchName` | desarrollo | `#{SourceVariables. BranchName}` |
| `CommitId` | exampleb01f91b31 | `#{SourceVariables. CommitId}` |
| `CommitMessage` | Se ha corregido un error (tamaño máximo de 100 KB) | `#{SourceVariables. CommitMessage}` |
| `CommitterDate` | 2019-10-29 T03:32:21Z | `#{SourceVariables. CommitterDate}` |
| `RepositoryName` | mi CodeCommitRepo | `#{SourceVariables. RepositoryName}` |

## CodeStarSourceConnection variables de salida de acción

Las variables de salida disponibles dependen del tipo de disparador (push, pull request o tag). Las variables de la tabla siguiente marcadas como Todas se generan para cada tipo de disparador. Las variables marcadas con un tipo de desencadenador específico solo se generan cuando ese desencadenador inicia la ejecución del proceso.

**`CodeStarSourceConnection`**variables (Bitbucket Cloud GitHub, GitHub Enterprise Repository y GitLab.com)

| Clave de variable | Ejemplo de valor | Ejemplo de sintaxis de variables | Tipo de desencadenador |
|---|---|---|---|
| `AuthorDate` | 29 de octubre de 2019 T03:32:21Z | `#{SourceVariables.AuthorDate}` | Todos |
| `AuthorDisplayName` | Jane Doe | `#{SourceVariables.AuthorDisplayName}` | Todos |
| `AuthorEmail` | jane.doe@example.com | `#{SourceVariables.AuthorEmail}` | Todos |
| `AuthorId` | jane doe | `#{SourceVariables.AuthorId}` | Todos |
| `BranchName` | desarrollo | `#{SourceVariables.BranchName}` | Empuja, Tag |
| `CommitId` | exampleb01f91b31 | `#{SourceVariables.CommitId}` | Todos |

| Clave de variable | Ejemplo de valor | Ejemplo de sintaxis de variables | Tipo de desencadenador |
|---|---|---|---|
| `CommitMessage` | Se ha corregido un error (tamaño máximo de 100 KB) | `#{SourceVariables.CommitMessage}` | Todos |
| `ConnectionArn` | arn:aws:codestar-connections:region ::conexión/ *account-id connection-id* | `#{SourceVariables.ConnectionArn}` | Todos |
| `DestinationBranchName` | main | `#{SourceVariables.DestinationBranchName}` | Solicitud de extracción |
| `FullRepositoryName` | username/GitHubRepo | `#{SourceVariables.FullRepositoryName}` | Todos |
| `ProviderType` | GitHub | `#{SourceVariables.ProviderType}` | Todos |
| `PullRequestId` | 1 | `#{SourceVariables.PullRequestId}` | Solicitud de extracción |
| `PullRequestTitle` | Añadir nueva función | `#{SourceVariables.PullRequestTitle}` | Solicitud de extracción |

| Clave de variable | Ejemplo de valor | Ejemplo de sintaxis de variables | Tipo de desencadenador |
|---|---|---|---|
| SourceBranchName | rama de características | #{SourceVariables.SourceBranchName} | Solicitud de extracción |
| TagName | v1.0.0 | #{SourceVariables.TagName} | Tag |

## GitHub variables de salida de acción GitHub (a través de la aplicación OAuth) action)

GitHub variables (acción GitHub (a través de la aplicación OAuth))

| Clave de variable | Ejemplo de valor | Ejemplo de sintaxis de variables |
|---|---|---|
| AuthorDate | 29 de octubre de 2019 T03:32:21 Z | #{SourceVariables.AuthorDate} |
| BranchName | main | #{SourceVariables.BranchName} |
| CommitId | exampleb01f91b31 | #{SourceVariables.CommitId} |
| CommitMessage | Se ha corregido un error (tamaño máximo de 100 KB) | #{SourceVariables.CommitMessage} |
| CommitterDate | 2019-10-29 T03:32:21Z | #{SourceVariables.CommitterDate} |
| CommitUrl | | #{SourceVariables.CommitUrl} |

| Clave de variable | Ejemplo de valor | Ejemplo de sintaxis de variables |
| --- | --- | --- |
| `RepositoryName` | mi GitHubRepo | `#{SourceVariables.RepositoryName}` |

### Variables de salida de acciones de S3

Variables de S3

| Clave de variable | Ejemplo de valor | Ejemplo de sintaxis de variables |
| --- | --- | --- |
| `ETag` | example28be1c3 | `#{SourceVariables.ETag}` |
| `VersionId` | exampleta_IUQCv | `#{SourceVariables.VersionId}` |

## Acciones con claves de variables configuradas por el usuario

Para CodeBuild las CloudFormation acciones de Lambda, las claves variables las configura el usuario.

Temas

- CloudFormation variables de salida de acciones
- CodeBuild variables de salida de la acción
- Variables de salida de acciones de Lambda

### CloudFormation variables de salida de acciones

CloudFormation variables

| Clave de variable | Ejemplo de sintaxis de variables |
| --- | --- |
| En el caso de las CloudFormation acciones, las variables se producen a partir de cualquier valor designado en la | `#{DeployVariables.StackName}` |

| Clave de variable | Ejemplo de sintaxis de variables |
| --- | --- |
| `Outputs` sección de una plantilla de pila. Tenga en cuenta que los únicos modos de CloudFormation acción que generan resultados son los que dan lugar a la creación o actualización de una pila, como la creación de pilas, las actualizaciones de pilas y la ejecución de conjuntos de cambios. Los modos de acción correspondientes que generan variables son:<br>• CREATE_UPDATE<br>• CHANGE_SET_EXECUTE<br>• CHANGE_SET_REPLACE<br>• REPLACE_ON_FAILURE<br><br>Para obtener más información sobre estos modos de acción, consulte Referencia de la acción de implementación de CloudFormation. Para ver un tutorial que muestra cómo crear una canalización con una acción de CloudFormation despliegue en una canalización que usa variables CloudFormation de salida, consulteTutorial: Cree una canalización que utilice variables de AWS CloudFormation acciones de despliegue. | |

## CodeBuild variables de salida de la acción

CodeBuild variables

| Clave de variable | Ejemplo de sintaxis de variables |
| --- | --- |
| En el CodeBuild caso de las acciones, las variables se producen a partir de los valores generados por las variables de entorno exportadas. Configura una variable de CodeBuild entorno editando tu CodeBuild acción CodePipeline o añadiendo la variable de entorno a la especificación de construcción. | `#{BuildVariables.EnvVar}` |

| Clave de variable | Ejemplo de sintaxis de variables |
| --- | --- |
| Añade instrucciones a tu CodeBuild especificación de compilación para añadir la variable de entorno en la sección de variables exportadas. Consulta env/exported -variables en la Guía del usuario. AWS CodeBuild | |

## Variables de salida de acciones de Lambda

Variables de Lambda

| Clave de variable | Ejemplo de sintaxis de variables |
| --- | --- |
| La acción Lambda generará como variables todos los pares clave-valor que se incluyen en la `outputVariables` sección de la solicitud de la API. PutJobSuccessResult<br><br>Para ver un tutorial con una acción de Lambda que usa variables de una acción ascendente (CodeCommit) y genera variables de salida, consulte. Tutorial: Uso de variables con acciones de invocación de Lambda | `#{TestVariables.testRunId}` |

# Trabajar con patrones glob en la sintaxis

Al especificar los archivos o las rutas que se utilizan en los artefactos de canalización o en las ubicaciones de origen, puede especificar el artefacto en función del tipo de acción. Por ejemplo, para la acción de S3, se especifica la clave del objeto de S3.

Para los desencadenadores, puede especificar filtros. Puede utilizar patrones glob para especificar los filtros. A continuación se muestran algunos ejemplos.

Cuando la sintaxis es "glob", se hace coincidir la representación en cadena de la ruta mediante un lenguaje de patrones limitado con una sintaxis similar a la de las expresiones regulares. Por ejemplo:

- `*.java` Especifica una ruta que representa un nombre de archivo que termina en .java
- `*.*` Especifica los nombres de archivo que contienen un punto
- `*.{java,class}` Especifica los nombres de archivo que terminan en .java o .class
- `foo.?` Especifica los nombres de archivo que comienzan por foo. y una extensión de un solo carácter

Para interpretar los patrones globales, se utilizan las siguientes reglas:

- Para especificar cero o más caracteres de un componente de nombre en los límites del directorio, utilice `*`.
- Para especificar cero o más caracteres de un componente de nombre que cruza los límites del directorio, utilice `**`.
- Para especificar un carácter de un componente de nombre, utilice `?`.
- Para escapar de caracteres que de otro modo se interpretarían como caracteres especiales, utilice el carácter de barra invertida (\).
- Para especificar un único carácter de un conjunto de caracteres, utilice `[ ]`.
- Para especificar un único archivo que esté en la raíz de la ubicación de compilación o de la ubicación del repositorio de origen, utilice `my-file.jar`.
- Para especificar un único archivo en un subdirectorio, use `directory/my-file.jar` o `directory/subdirectory/my-file.jar`.
- Para especificar todos los archivos, utilice `"**"`. El patrón glob de `**` indica que debe coincidir con cualquier número de subdirectorios.

- Para especificar todos los archivos y directorios de un directorio denominado `directory`, utilice `"directory/**"`. El patrón glob de `**` indica que debe coincidir con cualquier número de subdirectorios.
- Para especificar todos los archivos de un directorio denominado `directory`, pero no ninguno de sus subdirectorios, utilice `"directory/*"`.
- Dentro de una expresión de corchete, los caracteres `*`, `?` y `\` coinciden con ellos mismos. El carácter (-) coincide consigo mismo si es el primer carácter dentro de los corchetes o el primer carácter después de `!` durante la negación.
- Los caracteres `{ }` son un grupo de subpatrones, donde el grupo coincide si cualquier subpatrón del grupo coincide. El carácter `","` se usa para separar los subpatrones. Los grupos no pueden estar anidados.

# Actualizar canalizaciones de sondeo para utilizar el método de detección de cambios recomendado

Si tiene una canalización que utiliza sondeos para reaccionar a los cambios de origen, puede actualizarla para utilizar el método de detección recomendado. Para obtener una guía de migración con instrucciones para actualizar sus procesos de sondeo para utilizar el método recomendado de detección de cambios basado en eventos, consulte. Migrar los canales de sondeo para utilizar la detección de cambios basada en eventos

# Actualizar una acción fuente GitHub (a través de la aplicación OAuth) a una acción fuente GitHub (a través de GitHub la aplicación)

En AWS CodePipeline, hay dos versiones compatibles de la acción GitHub fuente:

- Recomendado:  La acción GitHub (a través de GitHub la aplicación) utiliza la autenticación GitHub basada en la aplicación respaldada por un recurso. CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas Instala una aplicación de AWS CodeStar Connections en tu GitHub organización para que puedas gestionar el acceso a ella. GitHub
- No se recomienda:  la acción GitHub (a través de la aplicación OAuth) usa tokens de OAuth para autenticarse GitHub y usa un webhook independiente para detectar los cambios. Este ya no es el método recomendado.

> **Note**
>
> Las conexiones no están disponibles en las regiones Asia Pacífico (Hong Kong), Asia Pacífico (Hyderabad), Asia Pacífico (Yakarta), Asia Pacífico (Melbourne), Asia Pacífico (Osaka), África (Ciudad del Cabo), Oriente Medio (Bahréin), Oriente Medio (Emiratos Árabes Unidos), Europa (España), Europa (Zúrich), Israel (Tel Aviv) o AWS GovCloud (US-West). Para hacer referencia a otras acciones disponibles, consulte Integraciones de productos y servicios con CodePipeline. Para ver consideraciones sobre esta acción en la región de Europa (Milán), consulte la nota que aparece en CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.

Utilizar la acción (mediante la GitHub aplicación) en lugar de la acción GitHub (mediante la GitHub aplicación OAuth) presenta algunas ventajas importantes:

- Con las conexiones, ya CodePipeline no se necesitan apps de OAuth ni tokens de acceso personal para acceder a tu repositorio. Cuando creas una conexión, instalas una GitHub aplicación que administra la autenticación en tu GitHub repositorio y otorga permisos a nivel de organización. Debe autorizar los tokens de OAuth como usuario para acceder al repositorio. Para obtener más información sobre el OAuth-based GitHub acceso en contraposición al App-based GitHub acceso,

consultehttps://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps.

- Cuando gestionas GitHub (mediante la GitHub aplicación) las acciones en la CLI o CloudFormation, ya no tendrás que guardar tu token de acceso personal como secreto en Secrets Manager. Ya no tienes que hacer referencia de forma dinámica al secreto almacenado en la configuración de tu CodePipeline acción. En su lugar, se agrega el ARN de conexión a la configuración de la acción. Para ver una acción de configuración de ejemplo, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server y acciones GitLab.com GitLab autogestionadas.
- Cuando creas un recurso de conexión para usarlo con tu acción GitHub (a través de la GitHub aplicación) CodePipeline, puedes usar el mismo recurso de conexión para asociar otros servicios compatibles, como CodeGuru Reviewer, a tu repositorio.
- En GitHub (mediante GitHub la aplicación), puedes clonar repositorios para acceder a los metadatos de git en CodeBuild acciones posteriores, mientras que en GitHub (mediante la aplicación OAuth) solo puedes descargar la fuente.
- Un administrador instala la aplicación en los repositorios de su organización. Ya no tiene que hacer un seguimiento de los tokens de OAuth que dependen de la persona que creó el token.

Todas las aplicaciones instaladas en una organización tienen acceso al mismo conjunto de repositorios. Para cambiar quién puede acceder a cada repositorio, modifique la política de IAM para cada conexión. Para ver un ejemplo, consulte Ejemplo: Una política de ámbito reducido para utilizar conexiones con un repositorio especificado.

Puedes seguir los pasos de este tema para eliminar tu acción fuente GitHub (mediante la aplicación OAuth) y añadir una acción fuente GitHub (mediante la GitHub aplicación) desde la consola. CodePipeline

Temas

- Paso 1: Sustituye tu acción (a través de la aplicación OAuth) GitHub
- Paso 2: Crear una conexión a GitHub
- Paso 3: Guarde la acción GitHub de origen

# Paso 1: Sustituye tu acción (a través de la aplicación OAuth) GitHub

Usa la página de edición del pipeline para reemplazar tu acción (a través de la aplicación OAuth) por una GitHub acción GitHub (a través GitHub de la aplicación).

Para reemplazar tu acción (a través de la aplicación OAuth) GitHub

1. Inicia sesión en la CodePipeline consola.
2. Seleccione su canalización y, a continuación, elija Editar. Elija Editar etapa en la etapa de fuente. Aparece un mensaje en el que se recomienda actualizar la acción.
3. En Action provider, selecciona GitHub (a través de GitHub la aplicación).
4. Realice una de las siguientes acciones:
   - En Conexión, si aún no has creado una conexión con tu proveedor, selecciona Conectar a GitHub. Continúe con el paso 2: crear una conexión a GitHub.
   - En Conexión, si ya ha creado una conexión con su proveedor, seleccione la conexión. Continúe con el Paso 3: Guardar la acción de origen para la conexión.

# Paso 2: Crear una conexión a GitHub

Tras elegir crear la conexión, se muestra la GitHub página Conectarse a.

Para crear una conexión a GitHub

1. En la configuración de GitHub conexión, el nombre de la conexión aparece en Nombre de la conexión.

   En GitHub Aplicaciones, elige la instalación de una aplicación o elige Instalar una nueva aplicación para crear una.

   > **Note**
   >
   > Se instala una aplicación para todas las conexiones a un proveedor en particular. Si ya has instalado la GitHub aplicación, elígela y omite este paso.

2. Si GitHub aparece la página de autorización, inicie sesión con sus credenciales y, a continuación, elija continuar.
3. En la página de instalación de la aplicación, aparece un mensaje que indica que la AWS CodeStar aplicación está intentando conectarse a tu GitHub cuenta.

> **Note**
>
> Solo instalas la aplicación una vez para cada GitHub cuenta. Si instaló la aplicación previamente, puede elegir Configurar para dirigirse a una página de modificación para la instalación de la aplicación o puede utilizar el botón Atrás para volver a la consola.

4. En la página Instalar AWS CodeStar, seleccione Instalar.
5. En la GitHub página Conectarse a, se muestra el identificador de conexión de la nueva instalación. Elija Conectar.

## Paso 3: Guarde la acción GitHub de origen

Complete las actualizaciones en la página Editar acción para guardar la nueva acción fuente.

Para guardar la acción GitHub de origen

1. En Repositorio, introduzca el nombre del repositorio de terceros. En Ramificación, introduzca la ramificación en la que desea que la canalización detecte los cambios de origen.

> **Note**
>
> En Repositorio, escriba `owner-name/repository-name` como se muestra en este ejemplo:
>
> ```
> my-account/my-repository
> ```

2. En Formato del artefacto de salida, debe elegir el formato de los artefactos.
   - Para almacenar los artefactos de salida de la GitHub acción mediante el método predeterminado, elija CodePipeline predeterminado. La acción accede a los archivos desde el GitHub repositorio y almacena los artefactos en un archivo ZIP en el almacén de artefactos de la canalización.

- Para almacenar un archivo JSON que contiene una referencia URL al repositorio de manera que las acciones posteriores puedan ejecutar comandos Git directamente, elija Clonación completa. Esta opción solo la pueden utilizar las acciones posteriores de CodeBuild .

  Si eliges esta opción, tendrás que actualizar los permisos de tu rol de servicio del CodeBuild proyecto, tal y como se muestra en. Añade CodeBuild GitClone permisos para las conexiones a Bitbucket, Enterprise Server o GitHub GitHub GitLab.com Para ver un tutorial que muestra cómo utilizar la opción Clonación completa, consulte Tutorial: Utilice un clon completo con una fuente de GitHub canalización.

3. En Artefactos de salida, puede conservar el nombre del artefacto de salida para esta acción, por ejemplo `SourceArtifact`. Seleccione Listo para cerrar la página Editar acción.
4. Seleccione Listo para cerrar la página de edición de etapa. Seleccione Guardar para cerrar la página de edición de la canalización.

# Cuotas en AWS CodePipeline

CodePipeline tiene cuotas para el número de canales, etapas, acciones y webhooks que puede tener una AWS cuenta en cada AWS región.

Estas cuotas se aplican a cada región y pueden aumentarse. Puede tardar hasta dos semanas procesar las solicitudes de aumento de cuota.

| Recurso | Predeterminado |
| --- | --- |
| Tiempo que transcurre hasta que finaliza el tiempo de espera de una acción<br>(Se trata de tiempos de espera configurables. Consulte la siguiente tabla para ver los tiempos de espera no configurables) | CloudFormation acción de despliegue: 3 días<br><br>CodeDeploy y acciones de despliegue de CodeDeploy ECS (blue/green): 5 días<br><br>AWS Lambda acción de invocación: 24 horas<br><br>**Note**<br>Mientras se ejecuta la acción, contacta CodePipeline periódica mente con Lambda para obtener un estado. La función de Lambda responde con un estado, en el que la ejecución de la acción es satisfactoria, produce un error o está en curso. Si la función de Lambda no ha enviado ninguna respuesta después de 20 minutos, se agota el tiempo de espera de la acción. Si, durante los 20 minutos, la función Lambda responde que la acción aún está en curso, CodePipeline reinicia el temporizador de 20 minutos y vuelve a intentarlo. Si no se ejecuta |

| Recurso | Predeterminado |
| --- | --- |
| | correctamente después de 24 horas, CodePipeline establece el estado de la acción de invocación de Lambda en fallida.<br>Lambda tiene un tiempo de espera independiente para las funciones de Lambda que no está relaciona do con el tiempo de espera de la acción. CodePipeline<br><br>Acción de implementación de Amazon S3: 90 minutos<br><br>**Note**<br>Si se agota el tiempo de espera para cargar en S3 durante el implementación de un archivo ZIP de gran tamaño, la acción no se realizará correctamente y se generará un error de tiempo de espera. Intente dividir el archivo ZIP en archivos más pequeños.<br><br>Tiempo de espera predeterminado a nivel de cuenta para la acción de aprobación manual: 7 días<br><br>**Note**<br>El tiempo de espera predeterm inado para la acción de aprobació n manual se puede anular para una |

| Recurso | Predeterminado |
| --- | --- |
| | acción específica en la canalizac ión, y se puede configurar hasta 86 400 minutos (60 días) con un valor mínimo de 5 minutos. Para obtener más información, consulta ActionDeclaration en la CodePipel ine Referencia de la API de . Al configurarse, este tiempo de espera se aplica a la acción. De lo contrario, se utiliza el nivel de cuenta predeterminado.<br><br>Todas las demás acciones: 1 hora<br><br>**Note**<br>El tiempo de espera de la acción de implementación de Amazon ECS se puede configurar hasta una hora (el tiempo de espera predeterminado). |
| Número máximo de canalizaciones totales por región en una cuenta AWS | 1 000<br><br>**Note**<br>Las canalizaciones configuradas para los sondeos o detección de cambios basada en eventos se cuentan para esta cuota. |

| Recurso | Predeterminado |
| --- | --- |
| Número máximo de canalizaciones configuradas para sondear los cambios de origen, por región de AWS | 300<br><br>**Note**<br>Esta es una cuota fija y no se puede cambiar. Si alcanza el límite de canalizaciones de sondeo, puede configurar canalizaciones adicional es que utilicen la detección de cambios basada en eventos. Para obtener más información, consulte Acciones de origen y métodos de detección de cambios[1]. |
| Número máximo de webhooks por región en una cuenta AWS | 300 |
| Número de acciones personalizadas por región en una cuenta AWS | 50 |

[1]Siga las instrucciones a continuación, en función del proveedor de código fuente, para actualizar las canalizaciones de sondeo a fin de utilizar la detección de cambios basada en eventos:

- Para actualizar una acción CodeCommit de origen, consulteMigre las canalizaciones de sondeo (CodeCommit o la fuente de Amazon S3) (consola).
- Para actualizar una acción de origen de Amazon S3, consulte Migre las canalizaciones de sondeo (CodeCommit o la fuente de Amazon S3) (consola).
- Para actualizar una acción GitHub de origen, consulteMigre los canales de sondeo a webhooks (mediante la aplicación OAuthGitHub ) (acciones fuente) (consola).

Las siguientes cuotas AWS CodePipeline se aplican a la disponibilidad regional, a las restricciones de nomenclatura y a los tamaños de artefactos permitidos. Estas cuotas son fijas y no pueden modificarse.

Para obtener una lista de los puntos finales del CodePipeline servicio de cada región, consulte los AWS CodePipeline puntos finales y las cuotas en la AWS Referencia general.

Para obtener información sobre necesidades estructurales, consulte CodePipeline referencia de estructura de tuberías.

| | |
|---|---|
| AWS Regiones en las que puedes crear una canalización | Este de EE. UU. (Norte de Virginia)<br>Este de EE. UU. (Ohio)<br>Oeste de EE. UU. (Norte de California)<br>Oeste de EE. UU. (Oregón)<br>Canadá (centro)<br>Europa (Fráncfort)<br>Europa (Zúrich)*<br>Europa (Estocolmo)<br>Europa (Milán)*<br>Europa (España) *<br>Europa (Irlanda)<br>Europa (Londres)<br>Europa (París)<br>Israel (Tel Aviv) *<br>África (Ciudad del Cabo)*<br>Asia-Pacífico (Hong Kong)*<br>Asia-Pacífico (Mumbai)<br>Asia Pacífico (Hyderabad) *<br>Asia-Pacífico (Tokio) |

| | |
|---|---|
| | Asia-Pacífico (Seúl)<br>Asia-Pacífico (Osaka)<br>Asia-Pacífico (Singapur)<br>Asia-Pacífico (Sídney)<br>Asia Pacífico (Yakarta) *<br>Asia Pacífico (Melbourne) *<br>Asia Pacífico (Malasia) *<br>Asia Pacífico (Nueva Zelanda) *<br>América del Sur (São Paulo)<br>Medio Oriente (Baréin)*<br>Oriente Medio (Emiratos Árabes Unidos) *<br>AWS GovCloud (US-West)<br>AWS GovCloud (US-East) |
| Caracteres permitidos en un nombre de acción | Los nombres de acción no pueden superar los 100 caracteres. Los caracteres permitidos son:<br>Letras minúsculas de la `a` a la `z`, ambas incluidas.<br>Letras mayúsculas de la `A` a la `Z`, ambas incluidas.<br>Número del `0` al 9 inclusive.<br>Caracteres especiales: `.` (punto), `@` (arroba), `-` (signo menos) y `_` (guion bajo).<br>Los demás caracteres, como los espacios, no están permitidos. |

| | |
|---|---|
| Caracteres permitidos en los tipos de acción | Los nombres de tipos de acción no pueden superar los 25 caracteres. Los caracteres permitidos son:<br><br>Letras minúsculas de la "a" a la "z", ambas incluidas.<br><br>Letras mayúsculas de la "A" a la "Z", ambas incluidas.<br><br>Números del 0 al 9, ambos incluidos.<br><br>Caracteres especiales: . (punto), @ (arroba), - (signo menos) y _ (guion bajo).<br><br>Los demás caracteres, como los espacios, no están permitidos. |
| Caracteres permitidos en los nombres de artefactos | Los nombres de artefactos no pueden superar los 100 caracteres. Los caracteres permitidos son:<br><br>Letras minúsculas de la a a la z, ambas incluidas.<br><br>Letras mayúsculas de la A a la Z, ambas incluidas.<br><br>Número del 0 al 9 inclusive.<br><br>Caracteres especiales: - (signo menos) y _ (guion bajo).<br><br>Los demás caracteres, como los espacios, no están permitidos. |

| | |
|---|---|
| Caracteres permitidos en nombres de acciones de socios | Los nombres de acciones de los socios deben seguir las mismas convenciones de nomenclatura y restricciones que los demás nombres de acciones de CodePipeline. En concreto, no pueden tener más de 100 caracteres. Los caracteres permitidos son:<br><br>Letras minúsculas de la "a" a la "z", ambas incluidas.<br><br>Letras mayúsculas de la "A" a la "Z", ambas incluidas.<br><br>Números del 0 al 9, ambos incluidos.<br><br>Caracteres especiales: . (punto), @ (arroba), - (signo menos) y _ (guion bajo).<br><br>Los demás caracteres, como los espacios, no están permitidos. |
| Caracteres permitidos en los nombres de canalización | Los nombres de canalización no pueden superar los 100 caracteres. Los caracteres permitidos son:<br><br>Letras minúsculas de la "a" a la "z", ambas incluidas.<br><br>Letras mayúsculas de la "A" a la "Z", ambas incluidas.<br><br>Números del 0 al 9, ambos incluidos.<br><br>Caracteres especiales: . (punto), @ (arroba), - (signo menos) y _ (guion bajo).<br><br>Los demás caracteres, como los espacios, no están permitidos. |

| | |
|---|---|
| Caracteres permitidos en los nombres de etapas | Los nombres de etapas no pueden superar los 100 caracteres. Los caracteres permitidos son:<br><br>Letras minúsculas de la "a" a la "z", ambas incluidas.<br><br>Letras mayúsculas de la "A" a la "Z", ambas incluidas.<br><br>Números del 0 al 9, ambos incluidos.<br><br>Caracteres especiales: . (punto), @ (arroba), - (signo menos) y _ (guion bajo).<br><br>Los demás caracteres, como los espacios, no están permitidos. |
| Tiempo que transcurre hasta que finaliza el tiempo de espera de una acción | CodeBuild acción de construcción: 36 horas<br><br>Acción de prueba: 8 horas<br><br>Acciones personalizadas: 24 horas<br><br>Acción de invocación de Step Functions: 7 días<br><br>Tiempo de espera de compilación para la acción de Comandos: 55 minutos |
| Longitud máxima permitida de la clave de configuración de la acción (por ejemplo, las claves de configuración de CodeBuild son `ProjectName` , `PrimarySource` y `EnvironmentVariables` ) | 50 caracteres |

| | |
|---|---|
| Longitud máxima del valor de configuración de la acción (por ejemplo, el valor de la `RepositoryName` configuración en la configuración de la CodeCommit acción debe ser inferior a 1000 caracteres):<br><br>`"RepositoryName": "my-repo-name-less-than-1000-characters" )` | 1.000 caracteres |
| Número máximo de acciones por canalización | 1 000 |
| Número máximo de ejecuciones de canalización simultáneas por canalización (modo EN COLA\|PARALELO) | 50 |
| Número máximo de ejecuciones de canalización simultáneas por ejecución de canalización en modo PARALELO | 5 |
| Número máximo de archivos para un objeto de Amazon S3 | 100 000 |
| Número máximo de acciones en paralelo de una etapa | 100 |
| Número máximo de acciones secuenciales de una etapa | 100 |

| | |
|---|---|
| Tamaño máximo de los artefactos de una etapa de código fuente | Artefactos almacenados en buckets de Amazon S3: 7 GB.<br><br>Artefactos almacenados en CodeCommit nuestros GitHub repositorios: 1 GB<br><br>Excepción: si se utilizan AWS Elastic Beanstalk para implementar aplicaciones, el tamaño máximo del artefacto es siempre de 512 MB.<br><br>Excepción: si se utilizan CloudFormation para implementar aplicaciones, el tamaño máximo del artefacto es siempre de 256 MB.<br><br>Excepción: si se utiliza la acción `CodeDeployToECS` para implementar las aplicaciones, el tamaño de artefacto máximo es siempre de 3 MB. |
| Tamaño máximo del archivo JSON de definiciones de imágenes que se utiliza en canalizaciones que implementan contenedores e imágenes de Amazon ECS | 100 KB |
| Tamaño máximo de los artefactos de entrada para las acciones CloudFormation | 256 MB |
| Tamaño máximo de artefactos de entrada para la acción `CodeDeployToECS` | 3 MB |
| Tamaño máximo de artefactos de entrada para la acción `Step Functions` | La acción Step Functions se ejecuta en Lambda y, por lo tanto, tiene cuotas de tamaño de artefacto que son las mismas que las cuotas de tamaño de artefacto de las funciones de Lambda. Para obtener más información, consulte Cuotas de Lambda en la Guía para desarrolladores de Lambda. |

| | |
|---|---|
| Tamaño máximo del objeto JSON que se puede almacenar en la propiedad `ParameterOverrides` | CloudFormation En el caso de una acción de CodePipeline despliegue con el proveedor, la `ParameterOverrides` propiedad se utiliza para almacenar un objeto JSON que especifica los valores del archivo de configuración de la CloudFormation plantilla. El límite máximo de tamaño del objeto JSON que se puede almacenar en la propiedad `ParameterOverrides` es de 1 kilobyte. |
| Número de acciones de una etapa | 1 como mínimo, 50 como máximo |
| Número de artefactos permitidos para cada acción | Para conocer el número de artefactos de entrada y salida permitidos para cada acción, consulte Artefactos de entrada y salida para cada tipo de acción |
| Número de meses en los que se retiene la información del historial de ejecución de la canalización | 12 |
| Número de fases de una canalización | 2 como mínimo, 50 como máximo |
| Etiquetas de canalización | Las etiquetas distinguen entre mayúsculas y minúsculas. Máximo de 50 por recurso. |

| | |
|---|---|
| Nombres de clave de etiqueta de canalización | Cualquier combinación de letras Unicode, números, espacios y caracteres permitidos de UTF-8 entre 1 y 128 caracteres de longitud. Los caracteres permitidos son + - = . _ : / @<br><br>Los nombres de clave de etiqueta deben ser únicos y cada clave puede tener un solo valor. Una etiqueta no puede:<br><br>• comience con AWS:<br>• contener únicamente espacios<br>• terminar con un espacio<br>• contener emojis o cualquiera de los siguiente s caracteres: ? ^ * [ \ ~ ! # $ % & * ( ) > < \| " ' |
| Valores de etiqueta de canalización | Cualquier combinación de letras Unicode, números, espacios y caracteres permitidos de UTF-8 entre 1 y 256 caracteres de longitud. Los caracteres permitidos son + - = . _ : / @<br><br>Una clave solo puede tener un valor, pero muchas claves pueden tener el mismo valor. Una etiqueta no puede:<br><br>• comience con AWS:<br>• contener únicamente espacios<br>• terminar con un espacio<br>• contener emojis o cualquiera de los siguiente s caracteres: ? ^ * [ \ ~ ! # $ % & * ( ) > < \| " ' |

| | |
|---|---|
| Desencadenadores | Hay un máximo de 50 desencadenadores en una definición de canalización entre las configuraciones `push` y `pull request`.<br><br>Hay un máximo de tres filtros por cada desencadenador de inserción y por cada desencadenador de solicitud de extracción.<br><br>**Note**<br>No se permiten duplicados para los filtros de la misma matriz de tipos de eventos.<br><br>Puede agregar hasta 8 patrones, ramificaciones y rutas de archivo de inclusión y 8 de exclusión para cada tipo de evento (inserción, solicitud de extracción).<br><br>Los caracteres permitidos en patternvalues incluyen todos los tipos de caracteres.<br><br>Hay una longitud máxima de 255 caracteres para los patrones de inclusión y exclusión.<br><br>Para nombres de etiquetas, hay una longitud máxima de 255 caracteres.<br><br>El tamaño máximo de la matriz `triggers` no debe superar los 200 KB |

| | |
|---|---|
| Filtros de desencadenadores | Rutas de archivo:<br>• Número de patrones: se pueden agregar hasta 8 patrones de inclusión y 8 de exclusión.<br>• Tamaño del patrón: cada patrón de inclusión o exclusión puede tener un tamaño máximo de 255 caracteres.<br><br>Ramificaciones:<br>• Número de patrones: se pueden agregar hasta 8 patrones de inclusión y 8 de exclusión.<br>• Tamaño del patrón: cada patrón de inclusión o exclusión puede tener un tamaño máximo de 255 caracteres.<br><br>Solicitudes de extracción:<br><br>Ramificaciones:<br>• Número de patrones: se pueden agregar hasta 8 patrones de inclusión y 8 de exclusión.<br>• Tamaño del patrón: cada patrón de inclusión o exclusión puede tener un tamaño máximo de 255 caracteres. |

| | |
|---|---|
| Unicidad de los nombres | En una sola AWS cuenta, cada canalización que crees en una AWS región debe tener un nombre único. Puede reutilizar los nombres para canalizaciones en diferentes regiones de AWS .<br><br>Los nombres de etapas deben ser únicos en una canalización.<br><br>Los nombres de acciones deben ser únicos en una etapa. |

| | |
|---|---|
| Cuotas para las variables de salida y los espacios de nombres | Hay un límite de tamaño máximo de 122 880 bytes para todas las variables de salida combinadas para una acción concreta.<br><br>Hay un límite de tamaño máximo de 100 KB para la configuración de acciones resueltas totales para una acción concreta.<br><br>Los nombres de las variables de salida distinguen mayúsculas de minúsculas.<br><br>Los espacios de nombres distinguen mayúsculas de minúsculas.<br><br>Los caracteres permitidos son:<br><br>• Letras minúsculas de la "a" a la "z", ambas incluidas.<br>• Letras mayúsculas de la "A" a la "Z", ambas incluidas.<br>• Números del 0 al 9, ambos incluidos.<br>• Caracteres especiales: ^ (intercalación), @ (arroba), - (signo menos), _ (guion bajo), [ (corchete izquierdo), ] (corchete derecho), * (asterisco), $ (dólar).<br><br>Los demás caracteres, como los espacios, no están permitidos. |

| | |
|---|---|
| Cuotas de variables a nivel de canalización | Hay un máximo de 50 variables a nivel de canalización por canalización.<br><br>Los nombres de las variables a nivel de canalización deben ser:<br><br>• 128 caracteres de longitud máxima<br>• Letras minúsculas de la "a" a la "z", ambas incluidas.<br>• Letras mayúsculas de la "A" a la "Z", ambas incluidas.<br>• Números del 0 al 9, ambos incluidos.<br>• Caracteres especiales `@\-_]+`<br><br>Los demás caracteres, como los espacios, no están permitidos.<br><br>Para los valores de variables, hay una longitud máxima de 1000 caracteres<br><br>Para los valores de variables, se permiten todos los caracteres.<br><br>Hay una longitud máxima de 200 caracteres para las descripciones de las variables. |

* Debe habilitar esta región antes de poder utilizarla.

# Apéndice A: acciones de GitHub origen (a través de la aplicación OAuth)

Este apéndice proporciona información sobre la acción GitHub (a través de la aplicación OAuth) en. CodePipeline

> **Note**
>
> Si bien no recomendamos usar la acción GitHub (a través de la aplicación OAuth), las canalizaciones existentes con la acción GitHub (a través de la aplicación OAuth) seguirán funcionando sin ningún impacto. En el caso de una canalización con una acción GitHub (a través de una aplicación OAuth), CodePipeline usa OAuth-based tokens para conectarse a tu repositorio. GitHub Por el contrario, la GitHub acción (a través de GitHub la aplicación) usa un recurso de conexión para asociar AWS recursos a tu GitHub repositorio. El recurso de conexión usa tokens basados en aplicaciones para conectarse. Para obtener más información sobre cómo actualizar tu canalización a la GitHub acción recomendada que usa una conexión, consultaActualizar una acción fuente GitHub (a través de la aplicación OAuth) a una acción fuente GitHub (a través de GitHub la aplicación). Para obtener más información sobre el OAuth-based GitHub acceso en contraposición al GitHub acceso basado en aplicaciones, consulte. https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps

Para integrarlo GitHub, CodePipeline usa una aplicación de GitHub OAuth para tu canalización. CodePipelineusa webhooks para gestionar la detección de cambios en tu canalización con la acción fuente GitHub (a través de la aplicación OAuth).

> **Note**
>
> Cuando configuras una acción de origen GitHub (mediante una GitHub aplicación) en CloudFormation, no incluyes ninguna información sobre el GitHub token ni añades ningún recurso de webhook. Los recursos de conexiones se configuran tal y como se muestra en AWS::CodeStarConnections: :Connection de la Guía del CloudFormation usuario.

Esta referencia contiene las siguientes secciones para la acción GitHub (mediante la aplicación OAuth):

- Para obtener información sobre cómo añadir una acción fuente y un webhook GitHub (mediante una aplicación OAuth) a una canalización, consulta. Añadir una acción fuente GitHub (mediante la aplicación OAuth)
- Para obtener información sobre los parámetros de configuración y ejemplos de YAML/JSON fragmentos de una acción fuente GitHub (mediante una aplicación de OAuth), consulte. GitHub (a través de la aplicación OAuth): referencia de la acción fuente

> **Important**
>
> Al crear CodePipeline webhooks, no utilices tus propias credenciales ni reutilices el mismo token secreto en varios webhooks. Para garantizar una seguridad óptima, genere un token secreto único para cada webhook que vaya a crear. El token secreto es una cadena arbitraria que tú proporcionas y que se GitHub utiliza para calcular y firmar las cargas útiles de los webhooks a las que se envían CodePipeline, a fin de proteger la integridad y la autenticidad de las cargas útiles de los webhooks. Usar sus propias credenciales o reutilizar el mismo token en varios webhooks puede provocar vulnerabilidades de seguridad.

> **Note**
>
> Si se ha proporcionado un token secreto, aparecerá redactado en la respuesta.

Temas

- Añadir una acción fuente GitHub (mediante la aplicación OAuth)
- GitHub (a través de la aplicación OAuth): referencia de la acción fuente

# Añadir una acción fuente GitHub (mediante la aplicación OAuth)

Añades acciones de origen GitHub (a través de la aplicación OAuth) de la siguiente manera: CodePipeline

- Utiliza el asistente de creación de canalizaciones (Creación de una canalización personalizada (consola)) de la CodePipeline consola o la página de edición de acciones para elegir la opción de GitHub proveedor. La consola crea un webhook que inicia su canalización cuando cambia la fuente.
- Usar la CLI para agregar la configuración de la acción `GitHub` y crear recursos adicionales de la siguiente manera:
  - Utilizar el ejemplo de configuración de acción `GitHub` en GitHub (a través de la aplicación OAuth): referencia de la acción fuente para crear la acción como se muestra en Crear una canalización (CLI).
  - Se desactivan las comprobaciones periódicas y se crea la detección de cambios de forma manual, ya que el método de detección de cambios consiste de forma predeterminada en iniciar la canalización sondeando la fuente. Migra su proceso de sondeo a webhooks para realizar acciones GitHub (a través de la aplicación OAuth).

# GitHub (a través de la aplicación OAuth): referencia de la acción fuente

> **Note**
>
> Si bien no recomendamos usar la acción GitHub (a través de la aplicación OAuth), las canalizaciones existentes con la acción GitHub (a través de la aplicación OAuth) seguirán funcionando sin ningún impacto. En el caso de una canalización con una acción fuente GitHub (a través de una aplicación OAuth), CodePipeline usa OAuth-based tokens para conectarse a tu repositorio. GitHub Por el contrario, la nueva GitHub acción (a través de GitHub la aplicación) usa un recurso de conexión para asociar AWS recursos a tu GitHub repositorio. El recurso de conexión usa tokens basados en aplicaciones para conectarse. Para obtener más información sobre cómo actualizar tu canalización a la GitHub acción recomendada que usa una conexión, consultaActualizar una acción fuente GitHub (a través de la aplicación OAuth) a una acción fuente GitHub (a través de GitHub la aplicación).

Activa la canalización cuando se realiza una nueva confirmación en el GitHub repositorio y la rama configurados.

Para integrarlo GitHub, CodePipeline usa una aplicación de OAuth o un token de acceso personal para tu canalización. Si utilizas la consola para crear o editar tu canalización, CodePipeline crea un GitHub webhook que inicia tu canalización cuando se produce un cambio en el repositorio.

Debes haber creado una GitHub cuenta y un repositorio antes de conectar la canalización mediante una GitHub acción.

Si quieres limitar el acceso CodePipeline a los repositorios, crea una GitHub cuenta y concede a la cuenta acceso solo a aquellos repositorios con los que quieras integrarte. CodePipeline Usa esa cuenta cuando configures el uso de GitHub repositorios CodePipeline para las etapas de origen de las canalizaciones.

Para obtener más información, consulta la documentación para GitHub desarrolladores en el GitHub sitio web.

Temas


- Tipo de acción
- Parámetros de configuración
- Artefactos de entrada
- Artefactos de salida
- Variables de salida
- Declaración de acción (GitHub ejemplo)
- Conectarse a GitHub (OAuth)
- Véase también


## Tipo de acción

- Categoría: `Source`
- Propietario: `ThirdParty`
- Proveedor: `GitHub`
- Versión: `1`

# Parámetros de configuración

Propietario

Obligatorio: sí

El nombre del GitHub usuario u organización propietario del GitHub repositorio.

Repo

Obligatorio: sí

El nombre del repositorio en el que se van a detectar los cambios de origen.

Rama

Obligatorio: sí

El nombre de la ramificación donde se van a detectar los cambios de origen.

OAuthToken

Obligatorio: sí

Representa el token de GitHub autenticación que CodePipeline permite realizar operaciones en tu GitHub repositorio. La entrada siempre se muestra como una máscara de cuatro asteriscos. Representa uno de los siguientes valores:

- Cuando usas la consola para crear la canalización, CodePipeline usa un token de OAuth para registrar la GitHub conexión.
- Cuando lo usas AWS CLI para crear la canalización, puedes pasar tu token de acceso GitHub personal a este campo. Sustituye los asteriscos (****) por el token de acceso personal del que has copiado. GitHub Cuando ejecute `get-pipeline` para ver la configuración de la acción, aparecerá la máscara de cuatro asteriscos de este valor.
- Cuando utilices una CloudFormation plantilla para crear la canalización, primero debes almacenar el token como secreto en. AWS Secrets Manager El valor de este campo debe incluirse como una referencia dinámica al secreto almacenado en Secrets Manager, como `{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}`.

Para obtener más información sobre los GitHub alcances, consulta la referencia de la API para GitHub desarrolladores en el GitHub sitio web.

PollForSourceChanges

Obligatorio: no

`PollForSourceChanges`controla si CodePipeline sondea el GitHub repositorio en busca de cambios en el código fuente. Le recomendamos que utilice webhooks para detectar cambios de código fuente en su lugar. Para obtener más información acerca de la configuración de webhooks, consulte Migre las canalizaciones de sondeo a webhooks GitHub (a través de la aplicación OAuth) (acciones fuente) (CLI) o Actualiza las canalizaciones para las acciones fuente de los eventos push GitHub (a través de la aplicación OAuth) (CloudFormation plantilla).

> **Important**
>
> Si tiene la intención de configurar webhooks, debe establecer `PollForSourceChanges` en `false` evitar ejecuciones de canalizaciones duplicadas.

Los valores válidos para este parámetro son:

- `True`: Si se establece, CodePipeline sondea tu repositorio para ver si hay cambios en el código fuente.

  > **Note**
  >
  > Si lo omites`PollForSourceChanges`, de CodePipeline forma predeterminada sondea tu repositorio para ver si hay cambios en el código fuente. Este comportamiento es el mismo que si `PollForSourceChanges` se establece en `true`.

- `False`: Si se establece, CodePipeline no sondea tu repositorio para ver si hay cambios en el código fuente. Utilice esta opción si desea configurar un webhook para detectar cambios de código fuente.

## Artefactos de entrada

- Número de artefactos: `0`
- Descripción: los artefactos de entrada no se aplican a este tipo de acción.

## Artefactos de salida

- Número de artefactos: 1

- Descripción: el artefacto de salida de esta acción es un archivo ZIP que contiene el contenido del repositorio configurado y la ramificación en la confirmación especificada como la revisión de origen para la ejecución de la canalización. Los artefactos generados desde el repositorio son los artefactos de salida de la GitHub acción. El identificador de confirmación del código fuente se muestra CodePipeline como la revisión de origen para la ejecución de la canalización activada.

## Variables de salida

Cuando se configura, esta acción produce variables a las que se puede hacer referencia mediante la configuración de acción de una acción descendente en la canalización. Esta acción produce variables que se pueden ver como variables de salida, incluso si la acción no tiene un espacio de nombres. Configure una acción con un espacio de nombres para que esas variables estén disponibles para la configuración de las acciones posteriores.

Para obtener más información sobre las variables en CodePipeline, consulteReferencia de variables.

CommitId

El ID de GitHub confirmación que activó la ejecución de la canalización. Los ID de confirmación son el volumen de dispositivo raíz completo de la confirmación.

CommitMessage

El mensaje de descripción, si lo hay, asociado a la confirmación que desencadenó la ejecución de la canalización.

CommitUrl

La dirección URL de la confirmación que activó la canalización.

RepositoryName

El nombre del GitHub repositorio en el que se realizó la confirmación que activó la canalización.

BranchName

El nombre de la rama del GitHub repositorio en el que se realizó el cambio de fuente.

AuthorDate

Fecha en la que se creó la confirmación, en formato de marca temporal.

CommitterDate

Fecha en la que se ha confirmado la confirmación, en formato de marca temporal.

# Declaración de acción (GitHub ejemplo)

## 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"
        }
    ]
},
```

## Conectarse a GitHub (OAuth)

La primera vez que utilices la consola para añadir un GitHub repositorio a una canalización, se te pedirá que autorices el CodePipeline acceso a tus repositorios. El token requiere los siguientes GitHub alcances:

- El ámbito `repo`, que se utiliza para controlar totalmente la lectura y la extracción artefactos de los repositorios públicos y privados en una canalización.
- El ámbito `admin:repo_hook`, que se utiliza para el control total de enlaces de repositorio.

Cuando utilices la CLI o una CloudFormation plantilla, debes proporcionar el valor de un token de acceso personal en GitHub el que ya hayas creado.

## Véase también

Los recursos relacionados siguientes pueden serle de ayuda cuando trabaje con esta acción.

- Referencia de recursos para la Guía del AWS CloudFormation usuario AWS::CodePipeline: :Webhook: incluye definiciones de campos, ejemplos y fragmentos del recurso en. CloudFormation
- Referencia de recursos para la Guía del AWS CloudFormation usuario AWS::CodeStar:: GitHubRepository — Incluye definiciones de campo, ejemplos y fragmentos del recurso en. CloudFormation
- Tutorial: Crea una canalización que compile y pruebe tu aplicación para Android con AWS Device Farm— Este tutorial proporciona un archivo de especificaciones de compilación de ejemplo y una

aplicación de ejemplo para crear una canalización con una fuente. GitHub Crea y prueba una aplicación de Android con y. CodeBuild AWS Device Farm

# AWS CodePipeline Historial de documentos de la Guía del usuario

En la siguiente tabla se describen los cambios importantes de cada versión de la Guía del CodePipeline usuario. Para recibir notificaciones sobre los cambios en esta documentación, puede suscribirse a una fuente RSS.

- Versión de la API: 2015-07-09
- Última actualización de la documentación: 23 de mayo de 2025

| Cambio | Descripción | Fecha |
| --- | --- | --- |
| Se corrigió la documentación sobre el comportamiento de los filtros activadores de tuberías | Se corrigió la documentación de los ejemplos de filtros de activación para describir con precisión la relación entre las inclusiones y las exclusiones. Las exclusiones tienen prioridad sobre las inclusiones en los filtros de ruta de archivos y ramas. Para obtener más información, consulte Ejemplos de filtros de activación. | 16 de marzo de 2026 |
| Support agregado para Azure DevOps | Support agregado para configurar conexiones para que AWS los recursos interactúen con Azure DevOps. Para obtener más información, consulte Connections for Azure DevOps. | 5 de agosto de 2025 |

| | | |
|---|---|---|
| Se ha actualizado la documentación del archivo de especificaciones de implementación para la acción de implementación de EC2. | Se ha agregado nueva información sobre el uso del archivo de especificaciones de implementación para la acción de implementación de EC2, con ejemplos y opciones de configuración actualizados. Consulte la página de referencia de la acción en Referencia de la acción de implementación de Amazon EC2. | 16 de mayo de 2025 |
| Se ha agregado documentación para la acción de implementación de Lambda. | Se ha agregado compatibilidad para la nueva acción de implementación de Lambda, con ejemplos y opciones de configuración actualizados. Consulte la página de referencia de la acción en Referencia de la acción de implementación de AWS Lambda. | 16 de mayo de 2025 |
| Se han agregado campos para el tipo de computación, el tipo de entorno y el tipo de variable de entorno a la acción Commands. | Se ha agregado información sobre nuevos campos que permiten especificar el entorno de compilación y el tipo de computación para la acción de comandos. También se ha agregado un campo nuevo para el tipo de variable de entorno. Consulte la página de referencia de acciones en CodePipeline Comands action. | 8 de mayo de 2025 |

| | | |
|---|---|---|
| Se ha agregado un nuevo tema donde se documenta la política para el rol de servicio predeterminado. | Se agregó información sobre la política de roles de servicio mínimos y se actualizó la tabla de enlaces a permisos de rol de servicio adicionales para cada acción realizada . CodePipeline Consulte la página de referencia de las nuevas reglas en la política CodePipeline de roles de servicio. | 26 de marzo de 2025 |
| Se ha agregado la nueva acción de invocación de `CodePipeline` . | Se ha agregado información acerca de la nueva acción de invocación de `CodePipeline` . Consulte la página de referencia de acciones en CodePipeline invoke action reference. | 14 de marzo de 2025 |
| Se ha agregado una nueva regla `CodeBuild` . | Se ha agregado información acerca de la nueva regla `CodeBuild` , que está disponible para ejecutar proyectos de compilación como regla para las condiciones de etapa. Consulte la página de referencia de la nueva CodeBuild regla en Rule reference. | 14 de marzo de 2025 |

| | | |
|---|---|---|
| Compartir conexiones en AWS CodeConnections | Para las canalizaciones con conexiones en uso AWS CodeConnections, puede usar conexiones configuradas para compartir entre Cuentas de AWS ellas. Puede configura r las conexiones compartid as en AWS Resource Access Manager. Para obtener más información, consulte Uso de una conexión compartida con otra Cuenta de AWS. | 6 de marzo de 2025 |
| EventBridge entrada, entrada de transformación, entrada para revisiones de fuentes | Para las canalizaciones con CodeCommit fuentes de Amazon ECR y Amazon S3, puede utilizar la entrada de transformación de EventBridge entrada para las revisiones de la fuente, donde `revisionV alue` se deriva de la variable de evento de origen de su clave de objeto (S3), commit (CodeCommit) o ID de imagen (ECR). Consulte las acciones y EventBridge recursos de origen de Amazon ECR, Cómo conectarse a las acciones de origen de Amazon S3 con una fuente habilitada para eventos y las acciones de CodeCommi t origen y EventBridge. | 3 de marzo de 2025 |

| | | |
|---|---|---|
| Cree una anulación de especificaciones para la acción AWS CodeBuild | Puedes optar por anular la especificación de construcción de la CodeBuild acción y, en su lugar, introducir los comandos directamente. Consulte la página de referencia de la acción en Referencia de la acción de compilación y prueba de AWS CodeBuild. Consulte también Creación de una canalización, etapas y acciones. | 3 de marzo de 2025 |
| Se ha agregado una nueva acción de implementación de EC2. | Se ha agregado información acerca de la nueva acción de implementación de EC2. Consulte la página de referencia de la acción en Referencia de la acción de implementación de Amazon EC2. Para ver un tutorial, consulte el tutorial: Implementación en instancias de EC2 con. CodePipeline | 21 de febrero de 2025 |
| Se ha agregado una nueva acción de implementación de EKS. | Se ha agregado información acerca de la nueva acción de implementación de EKS. Consulte la página de referencia de la acción en Referencia de la acción de implementación de EKS. Para ver un tutorial, consulte Tutorial: Implementación en Amazon EKS con CodePipeline. | 20 de febrero de 2025 |

| | | |
|---|---|---|
| Nuevas CloudWatch métricas y dimensiones para CodePipeline | Se agregaron métricas para las canalizaciones en CloudWatch las métricas. Consulta las CodePipeline CloudWatch métricas. | 13 de febrero de 2025 |
| Se ha agregado una nueva regla `Commands`. | Se ha agregado información acerca de la nueva regla `Commands`, que está disponible para ejecutar comandos del intérprete de comandos como regla para las condiciones de etapa. Puede consultar la página de referencia de la regla en Referencia de la regla de comandos. | 17 de diciembre de 2024 |

| | | |
|---|---|---|
| Se han ampliado los ejemplos y la información de referencia sobre los desencadenadores. | Se han agregado descripciones para los filtros de eventos de solicitud de extracción por proveedor en Eventos de solicitud de extracción para desencadenadores por proveedor. Se han agregado ejemplos ampliados para los desencadenadores, con información más detallada sobre las inclusiones y exclusiones en los eventos de inserción y eventos de solicitud de extracción en Eventos de solicitud de extracción para desencadenadores por proveedor. Se ha agregado más información de referencia para JSON sobre las inclusiones y exclusiones en Desencadenadores. | 17 de diciembre de 2024 |
| Se han agregado nuevas acciones al catálogo de acciones. | Ahora puede utilizar las acciones `ECRBuildAndPublish` e `InspectorScan`. Para obtener más información, consulta las páginas de referencia ECRBuildAndPublishy las InspectorScanacciones. | 22 de noviembre de 2024 |

| | | |
|---|---|---|
| Nueva configuración automática para el reintento de etapas en caso de fallo | Puede configurar una etapa para reintentar automáticamente una etapa fallida o acciones fallidas en la etapa. Para obtener más información, consulte Configuración de una etapa para el reintento automático en caso de fallo. | 15 de octubre de 2024 |
| Nuevo resultado de `Skip` para las condiciones de entrada | Se agregó información sobre el resultado de `Skip` que está disponible para las condiciones de entrada. Puede usar las reglas `VariableCheck` y `LambdaInvoke` con esta configuración. Consulte los pasos en Creación de condiciones de entrada con el resultado Omitir. Para ver una lista de aspectos que tener en cuenta con respecto a las condiciones con resultados Omitir, consulte Consideraciones sobre la configuración de los resultados para las condiciones de etapa. | 15 de octubre de 2024 |

| | | |
|---|---|---|
| Nuevos pasos en la consola para crear canalizaciones a partir de plantillas estáticas | En la CodePipeline consola, puedes usar un nuevo asistente de creación de canalizaciones para elegir entre varias plantillas estáticas con las que generar los recursos de canalización CloudFormation. Para obtener más información, consulte Creación de una canalización a partir de plantillas estáticas. | 9 de octubre de 2024 |
| Nueva acción `Commands` | Se ha agregado información sobre la nueva acción `Commands` que está disponible para ejecutar comandos del intérprete de comandos como una acción en su canalización. Consulta la página de referencia de la nueva acción en Acción de comandos y los permisos de la función de servicio en Añadir permisos a la función de CodePipeline servicio. Para ver un tutorial, consulte Tutorial: Creación de una canalización que ejecute comandos mediante computación. | 3 de octubre de 2024 |

| | | |
|---|---|---|
| Nueva regla `VariableCheck` para las condiciones | Se ha agregado información sobre la regla `VariableCheck` para las condiciones de etapa. Consulte la página de referencia de la nueva regla en VariableCheck. Para ver un tutorial, consulte Tutorial: Cómo crear una regla de verificación de variables para una canalización como una condición de entrada. | 27 de septiembre de 2024 |
| Actualizaciones de las conexiones para GitHub | Se agregó información sobre el uso del token GitHub de acceso del usuario con las conexiones a GitHub (acciones GitHub de la versión 2). Los tokens de acceso de los usuarios se utilizan en los CodeBuild proyectos. Consulte GitHub las conexiones y el tutorial: Utilice un clon completo con una fuente de GitHub canalización. | 16 de septiembre de 2024 |
| Actualizaciones para reestructurar la referencia de JSON en canalizaciones y la tabla de contenidos de la guía | La guía se ha reestructurado e incluye cambios en los títulos de algunas secciones para mejorar la usabilidad de las secciones de referencia y de las tareas. | 16 de agosto de 2024 |

| | | |
|---|---|---|
| Actualización del campo del token secreto en la respuesta para las acciones `PutWebhook` y `ListWebhooks` | Se ha actualizado el campo del token secreto para las acciones `PutWebhook` y `ListWebhooks` . Si se ha proporcionado un token secreto, aparecerá redactado en la respuesta. Consulte las notas añadidas al apéndice A: acciones fuente de la GitHub versión 1. Para ver las actualizaciones relacionadas en la Guía de CodePipeline API, consulte PutWebhooky ListWebhooks. | 6 de agosto de 2024 |
| Se ha agregado nuevo contenido para las condiciones y reglas de etapas | Ahora puede configurar las condiciones y reglas de etapas para canalizaciones de tipo V2. Consulte Conceptos, Funcionamiento de las condiciones de las etapas y Configuración de las condiciones de una etapa. Se ha agregado un capítulo de referencia sobre las reglas que proporciona información de referencia. Consulte la referencia de CodePipeline reglas. | 30 de julio de 2024 |

| | | |
|---|---|---|
| Se ha agregado nueva información de referencia para los tipos de canalización y las características relacionadas | Hay disponible un nuevo script de análisis para evaluar el costo de pasar a canalizaciones de tipo V2. Consulte ¿Qué tipo de canalización es el adecuado para mí? Se ha agregado una tabla de referencia que proporciona enlaces a toda la documentación del CodePipeline servicio por función. Consulte la referencia CodePipeline de funciones. | 11 de julio de 2024 |
| Actualiza la acción de origen de S3 para agregar una nueva opción para las anulaciones de origen | Hay disponible una nueva opción para las anulaciones de origen denominada `S3_OBJECT_KEY` para la acción de origen de S3. Se ha agregado un nuevo parámetro `AllowOverrideForS3ObjectKey` para la acción de origen de S3. Consulte la página de referencia de Acción de origen de Amazon S3 e Iniciar una canalización con una anulación de revisión de código fuente. | 7 de junio de 2024 |

| | | |
|---|---|---|
| Actualiza la acción de origen de S3 para agregar nuevas variables de salida | Hay disponibles nuevas variables de salida denominadas `BucketName` y `ObjectKey` para la acción de origen de S3. Consulte la página de referencia de Acción de origen de Amazon S3. | 5 de junio de 2024 |
| Actualizaciones de las acciones `CloudFormationStackSet` y `CloudFormationStackInstances` | Se ha agregado el parámetro `CallAs` para la acción `CloudFormationStackSet` y `CloudFormationStackInstances`. Consulte la página de referencia de la acción. | 2 de mayo de 2024 |
| Compatibilidad con reversiones a nivel de etapa | Puede revertir una etapa de forma manual o automática a una ejecución de canalización previa realizada correctamente para la etapa. Consulte Configuración de la reversión de etapas y Conceptos. | 26 de abril de 2024 |
| Actualizaciones de la disponibilidad regional StackSets y de las acciones de Step Functions | Las acciones Step Functions StackSets y Step Functions ya están disponibles en todas las regiones en las CodePipeline que están disponibles. Consulte CloudFormation StackSets la referencia de acciones y la referencia de acciones de AWS Step Functions. | 27 de marzo de 2024 |

| | | |
|---|---|---|
| Actualizaciones a políticas administradas | `AWSCodePipeline_FullAccess` Se actualizó la política AWS gestionada. Consulte las políticas administradas por AWS para AWS CodePipeline. | 15 de marzo de 2024 |
| Compatibilidad con el tiempo de espera configurable para realizar acciones de aprobación manual | Se ha agregado información sobre las cuotas para el nuevo campo de tiempo de espera configurable para las acciones de aprobación manual. Para obtener más información, consulte Cuotas. | 15 de febrero de 2024 |
| Compatibilidad para el filtrado de desencadenadores por ramificaciones y rutas de archivo | Compatibilidad adicional para la configuración de desencadenadores que permite filtrar el estado de las solicitudes de extracción, las ramificaciones y las rutas de archivo para canalizaciones de tipo V2. Para obtener más información, consulte Filtros de desencadenadores en solicitudes de inserción o extracción de código, Desencadenadores y Cómo filtrar por nombres de ramificación para solicitudes de extracción a fin de iniciar la canalización y Cuotas. | 8 de febrero de 2024 |

| | | |
|---|---|---|
| Compatibilidad con los nuevos modos de ejecución de canalizaciones | Compatibilidad adicional para los modos de ejecución de canalizaciones PARALELO y EN COLA. Para obtener más información, consulte Configuración o cambio del modo de ejecución de una canalización, Procesamiento de ejecuciones en modo EN COLA, Procesamiento de ejecuciones en modo PARALELO y Cuotas. | 8 de febrero de 2024 |
| Actualizaciones de las páginas de la consola para ver los detalles de las acciones, revisar las acciones de aprobación manual y la página de listas de canalizaciones | Las actualizaciones de la consola están documenta das para el nuevo botón y cuadro de diálogo Ver detalles, el nuevo cuadro de diálogo de aprobació n manual y las nuevas columnas para las ejecucion es recientes de la página de listas de canalizaciones. Para obtener más información, consulte Ver canalizaciones (consola), Ver detalles de las acciones en una canalización y Administración de acciones de aprobación en canalizac iones. | 10 de enero de 2024 |

| | | |
|---|---|---|
| Support para GitLab autogestión | Support agregado para configurar conexiones para que AWS los recursos interactúen con los recursos GitLab autogestionados. Para obtener más información, consulte Conexiones GitLab autogestionadas. | 28 de diciembre de 2023 |
| Actualizaciones de las acciones `CloudFormationStackSet` y `CloudFormationStackInstances` | Se ha agregado el parámetro `ConcurrencyMode` para la acción `CloudFormationStackSet` y `CloudFormationStackInstances`. Consulte la página de referencia de la acción. | 19 de diciembre de 2023 |
| Actualizaciones de los parámetros AWS Device Farm de acción en CodePipeline | Se CodePipeline han actualizado los parámetros de la AWS Device Farm acción en. Para obtener más información, consulte la Referencia de la acción de AWS Device Farm. | 18 de diciembre de 2023 |
| Support agregado para mensajes de error detallados para la CloudFormation acción en CodePipeline | CloudFormation Los mensajes de error de acción ahora pueden mostrar detalles sobre los recursos que fallaron. Para obtener más información, consulte la Referencia de la acción de CloudFormation. | 15 de diciembre de 2023 |

| | | |
|---|---|---|
| Las actualizaciones para iniciar una canalización con la revisión de origen anulada CodePipeline | Ahora puede iniciar una canalización con una revisión de código fuente específica. Para obtener más información, consulte Iniciar una canalización con una anulación de revisión de código fuente. | 17 de noviembre de 2023 |
| Nuevas regiones compatibles | CodePipeline ya está disponible en las regiones de Asia Pacífico (Hyderabad), Asia Pacífico (Yakarta), Asia Pacífico (Melbourne), Asia Pacífico (Osaka), Oriente Medio (Emiratos Árabes Unidos), Europa (España) e Israel (Tel Aviv). Se ha actualizado el tema Referencia del bucket de marcadores de posición de eventos y el tema de Puntos de conexión de Servicio de AWS. | 13 de noviembre de 2023 |
| Actualizaciones de los campos de eventos en Amazon EventBridge | Ahora puedes ver los campos de eventos actualizados en Amazon EventBridge. Para obtener más información, consulte Supervisión de CodePipeline eventos. | 9 de noviembre de 2023 |

| | | |
|---|---|---|
| Actualizaciones para canalizaciones nuevas de tipo V2, activadores en etiquetas de Git y variables de canalización en CodePipeline | Ahora puedes elegir un tipo de canalización. CodePipeline En el caso de una canalización de tipo V2, ahora puedes usar una configuración de activación para iniciar la canalización en etiquetas de Git. Con las canalizaciones de tipo V2, también puedes usar variables a nivel de canalización para pasar los parámetros de entrada para la ejecución de una canalización. Para obtener más información, consulta Variables, Tutorial: Usar variables a nivel de canalización y Tutorial: Usar etiquetas de Git para iniciar la canalización. Para obtener más información acerca de los tipos de objetos de canalización, consulte Tipos de canalizaciones. | 24 de octubre de 2023 |
| CodePipeline permite volver a intentar todas las acciones en una fase fallida | En el caso de una etapa fallida CodePipeline, puede volver a intentar la etapa sin volver a ejecutar la canalización. Para ello, puede volver a intentar las acciones fallidas de una etapa o volver a intentar todas las acciones de la etapa empezando por la primera acción de la etapa. Para obtener más información, consulte. | 17 de octubre de 2023 |

| | | |
|---|---|---|
| Support for GitLab groups | Support agregado para configurar conexiones para que AWS los recursos interactúen con GitLab los grupos. Para obtener más información, consulte GitLab Conexiones. | 15 de septiembre de 2023 |
| CodePipeline admite conexiones a GitLab .com | Puede usar las conexione s para configurar AWS los recursos para interactuar con GitLab .com. También puede elegir la opción de clonación completa para usar los comandos y metadatos de Git para las acciones posteriores. Para obtener más información, consulte el tema de referencia sobre las GitLab conexiones y la estructura de CodeStarSourceConnection acciones. | 10 de agosto de 2023 |
| Actualización a la acción `CloudFormationStac kInstances` | Se agregó el parámetro `RegionConcurrencyT ype` para la acción `CloudFormationStac kInstances`. Consulte la página de referencia de la acción para ver la acción `CloudFormationStac kInstances`. | 8 de agosto de 2023 |

| | | |
|---|---|---|
| Actualización a la acción `CloudFormationStackSet` | Se agregó el parámetro `RegionConcurrencyType` para la acción `CloudFormationStackSet`. Consulte la página de referencia de la acción para ver la acción `CloudFormationStackSet`. | 24 de julio de 2023 |
| Actualizaciones a políticas administradas | `AWSCodePipeline_FullAccess` Se actualizó la política AWS gestionada. Consulte Políticas administradas de AWS para AWS CodePipeline. | 21 de junio de 2023 |
| Actualizaciones de los procedimientos de migración de los canales de sondeo | Los procedimientos para migrar (actualizar) las canalizaciones de sondeo para utilizar la detección de cambios basada en eventos se han actualizado con los pasos para las canalizaciones que utilizan un bucket de Amazon S3 habilitado para recibir notificaciones. EventBridge Para obtener más información, consulte Migrar los canales de sondeo para utilizar la detección de cambios basada en eventos. | 12 de junio de 2023 |

| | | |
|---|---|---|
| Actualizaciones de las políticas administradas | Las políticas AWS `AWSCodePipeline_FullAccess` gestionadas se `AWSCodePipeline_ReadOnlyAccess` han actualizado con un permiso adicional. Para obtener más información, consulte AWS CodePipeline las actualizaciones de las políticas AWS administradas. | 16 de mayo de 2023 |
| Actualizaciones de las políticas administradas | Las políticas AWS administradas ya `AWSCodePipelineFullAccess` `AWSCodePipelineReadOnlyAccess` están en desuso. Utilice las políticas `AWSCodePipeline_FullAccess` y `AWSCodePipeline_ReadOnlyAccess` . Consulte Actualizaciones de AWS CodePipeline a las políticas administradas de AWS. | 17 de noviembre de 2022 |

| | | |
|---|---|---|
| Actualizaciones de los procedimientos que utilizan CloudTrail | Todos los procedimientos de la consola, los comandos CLI de muestra y CloudFormation los fragmentos y plantillas de muestra para una canalización con una fuente S3 se han actualizado con la opción de elegir Escribir y seleccionar false para los eventos de administración en. CloudTrail Consulte los ejemplos actualizados en Cómo iniciar una canalización, en el tutorial: Crear una canalización con CloudFormation, Editar canalizaciones para usar eventos push y Actualizar canalizaciones de sondeo. | 27 de abril de 2022 |
| Nueva integración compatible con Snyk | Puedes usar la acción de invocación de Snyk CodePipeline para automatizar el análisis de seguridad de tu código fuente abierto. Para obtener más información, consulte la referencia de acciones de Snyk y las integraciones. | 10 de junio de 2021 |
| Nueva región compatible Europa (Milán) | CodePipeline ya está disponible en Europa (Milán). Se han actualizado los temas Límites y Puntos de conexión de Servicio de AWS. | 27 de enero de 2021 |

| | | |
|---|---|---|
| La detección de cambios se puede desactivar para las acciones de origen con conexiones | Puede usar la CLI o el SDK para actualizar una acción de `CodeStarSourceConnection` origen y desactivar la detección automática de cambios en el repositorio de origen. El tema de referencia sobre la estructura de CodeStarSourceConnection acciones se ha actualizado con una descripción del `DetectChanges` parámetro. | 8 de enero de 2021 |
| CodePipeline ahora admite acciones CloudFormation StackSets de despliegue | En un nuevo tutorial, Tutorial: Crear una canalización que se utilice CloudFormation StackSets como proveedor de despliegues, se indican los pasos a seguir CloudFormation StackSets para crear y actualizar los conjuntos de pilas y las instancias de pila con la canalización. También se ha añadido el tema de referencia sobre la estructura de CloudFormation StackSets acciones. | 30 de diciembre de 2020 |
| Nueva región compatible Asia-Pacífico (Hong Kong) | CodePipeline ya está disponible en Asia Pacífico (Hong Kong). Se han actualizado los temas Límites y Puntos de conexión de Servicio de AWS. | 22 de diciembre de 2020 |

| | | |
|---|---|---|
| Consulta los patrones de EventBridge eventos actualizados en CodePipeline | Se han agregado a Monitoring CodePipeline events los patrones y estados de eventos actualizados para los eventos en proceso, fase y nivel de acción. | 21 de diciembre de 2020 |
| Consulta las ejecuciones de canalizaciones entrantes en CodePipeline | Puede utilizar la consola de la o la CLI para ver las ejecucion es entrantes. Para obtener más información, consulte Ver una ejecución entrante (consola) y Ver el estado de una ejecución entrante (CLI). | 16 de noviembre de 2020 |
| La acción CodeCommit de origen CodePipeline es compatible con la opción de clonación completa | Cuando usas una acción de CodeCommit origen, puedes elegir la opción de clonación completa para usar los comandos y metadatos de Git para CodeBuild las acciones posteriores. Para obtener más información, consulta la referencia de la CodeCommit acción y el tutorial: Usa la clonación completa con una fuente de CodeCommit canalización. | 11 de noviembre de 2020 |

| | | |
|---|---|---|
| CodePipeline admite conexiones a GitHub un servidor GitHub empresarial | Puede usar las conexiones para configurar AWS los recursos con los que interactuar GitHub, GitHub Enterprise Cloud y GitHub Enterprise Server. También puede elegir la opción de clonación completa para usar los comandos y metadatos de Git para las acciones posteriores. Para obtener más información, consulte GitHub las conexiones, las conexiones de GitHub Enterprise Server y el tutorial: Utilice una clonación completa con una fuente de GitHub canalización. Si ya tienes una canalización con una acción de GitHub origen, consulta Actualizar una acción de origen GitHub (mediante una OAuth aplicación) a una acción de origen GitHub (mediante una GitHub aplicación). | 30 de septiembre de 2020 |
| La CodeBuild acción permite habilitar las compilaciones por lotes AWS CodePipeline | Para CodeBuild las acciones de tu proceso, puedes habilitar las compilaciones por lotes para ejecutar varias compilaciones en una sola ejecución. Para obtener más información, consulta la referencia sobre la estructura de CodeBuild acciones y Crear una canalización (consola). | 30 de julio de 2020 |

| | | |
|---|---|---|
| AWS CodePipeline ahora admite acciones AWS AppConfig de despliegue | En un nuevo tutorial, Tutorial: crear una canalización que se utilice AWS AppConfig como proveedor de despliegues, se indican los pasos a seguir AWS AppConfig para implementar los archivos de configuración con la canalización. También se ha añadido el tema de referencia sobre la estructura de AWS AppConfig acciones. | 25 de junio de 2020 |
| AWS CodePipeline ahora es compatible con Amazon VPC en AWS GovCloud (EE. UU.-Oeste) | Ahora puedes conectarte directamente a AWS CodePipeline través de un punto de conexión privado de Amazon VPC en AWS GovCloud (EE. UU., oeste). Para obtener más información, consulte Uso CodePipeline con Amazon Virtual Private Cloud. | 2 de junio de 2020 |

| | | |
|---|---|---|
| AWS CodePipeline ahora admite acciones de AWS Step Functions invocación | Ahora puedes crear una canalización CodePipeline que se utilice AWS Step Functions como proveedor de acciones de invocación. Un nuevo tutorial, Tutorial: Use una acción de AWS Step Functions invocación en una canalización, proporciona los pasos para iniciar la ejecución de una máquina de estados desde su canalización. También se ha incorporado el tema AWS Step Functions Action Structure Reference. | 28 de mayo de 2020 |
| Ver, mostrar y actualizar conexiones | Puede mostrar, eliminar y actualizar las conexiones en la consola. Consulte Listar conexiones en CodePipeline. | 21 de mayo de 2020 |
| Las conexiones permiten etiquetar los recursos en la CLI | Los recursos de conexiones ahora admiten el etiquetado en la AWS CLI. Las conexione s ahora se integran con AWS CodeGuru. Consulte Referenci a de permisos de IAM para conexiones. | 6 de mayo de 2020 |
| CodePipeline ahora está disponible en AWS GovCloud (EE. UU.-Oeste) | Ahora se puede utilizar CodePipeline en AWS GovCloud (EE. UU. al oeste). Para obtener más información, consulte Cuotas. | 8 de abril de 2020 |

| | | |
|---|---|---|
| El tema de cuotas muestra qué cuotas CodePipeline de servicio se pueden configurar | Se ha CodePipeline cambiado el formato del tema de las cuotas. En la documenta ción, se indica qué cuotas de servicio se pueden configura r y qué cuotas no. Consulte Cuotas en. AWS CodePipeline | 12 de marzo de 2020 |
| El tiempo de espera de la acción de implementación de Amazon ECS se puede configurar | El tiempo de espera de la acción de implementación de Amazon ECS se puede configurar hasta una hora (el tiempo de espera predeterm inado). Consulte las cuotas de AWS CodePipeline. | 5 de febrero de 2020 |
| Nuevos temas describen cómo detener una ejecución de canalización | Puede detener una ejecución de canalización en CodePipel ine. Puede especificar que la ejecución se detenga después de que se permita completar las acciones en curso o puede especificar que se detenga la ejecución inmediatamente y se abandonen las acciones en curso. Consulte Cómo se detienen las ejecuciones de canalización y Detener una ejecución de canalización en CodePipeline. | 21 de enero de 2020 |

| | | |
|---|---|---|
| CodePipeline admite conexiones | Puede usar las conexiones para configurar AWS los recursos para que interactúen con los repositorios de código externos. Cada conexión es un recurso que los servicios pueden utilizar, por ejemplo, CodePipeline para conectarse a un repositorio de terceros, como Bitbucket Cloud. Para obtener más información, consulta Cómo trabajar con conexiones en CodePipeline. | 18 de diciembre de 2019 |
| Se han actualizado los temas de seguridad, autenticación y control de acceso | La información de seguridad, autenticación y control de acceso de se CodePipeline ha organizado en un nuevo capítulo de seguridad. Para obtener más información, consulte Seguridad. | 17 de diciembre de 2019 |
| Los nuevos temas describen cómo puede usar variables en sus canalizaciones | Ahora puede configurar espacios de nombres para acciones y generar variables cada vez que se complete la ejecución de la acción. Puede configurar acciones posteriores para hacer referencia a estos espacios de nombres y variables. Consulte Trabajar con variables y Variables. | 14 de noviembre de 2019 |

| | | |
|---|---|---|
| Los nuevos temas describen cómo funcionan las ejecuciones de canalización, por qué las etapas se bloquean durante una ejecución y cuándo se reemplazan las ejecuciones de canalización | Se han agregado varios temas a la sección Bienvenida para describir cómo funcionan las ejecuciones de canalización, incluido el motivo por el que se bloquean las etapas durante una ejecución y qué sucede cuando se reemplazan las ejecuciones de canalización. Estos temas incluyen una lista de conceptos, un ejemplo de DevOps flujo de trabajo y recomendaciones sobre cómo debe estructurarse una canalización. Se han agregado los siguientes temas: términos de canalización, ejemplos de DevOps canalización y cómo funcionan las ejecuciones de canalizaciones. | 11 de noviembre de 2019 |
| CodePipeline admite las reglas de notificación | Ahora puede utilizar reglas de notificación para notificar a los usuarios acerca de cambios importantes en las canalizaciones. Para obtener más información, consulte Crear una regla de notificación. | 5 de noviembre de 2019 |

| | | |
|---|---|---|
| CodeBuild variables de entorno disponibles en CodePipeline | Puedes configurar variables de CodeBuild entorno en la acción de CodeBuild creación de tu canalización. Puede utilizar la consola o la CLI para añadir el parámetro `EnvironmentVariables` a la estructura de canalizac ión. Se ha actualizado el tema Crear una canalización (consola). También se han actualizado los ejemplos de configuración en la referencia de acciones de CodeBuild. | 14 de octubre de 2019 |
| Nueva región | CodePipeline ya está disponibl e en Europa (Estocolmo). Se han actualizado los temas Límites y Puntos de conexión de Servicio de AWS. | 5 de septiembre de 2019 |
| Especifique el control ACLs almacenado y en caché para las acciones de implement ación de Amazon S3 | Ahora puede especificar opciones predefinidas de ACL y control de caché al crear una acción de despliegue de Amazon S3 en CodePipel ine. Se han actualizado los siguientes temas: Creación de una canalización (consola) , Referencia de estructura de CodePipeline canalizac ión y Tutorial: creación de una canalización que utilice Amazon S3 como proveedor de despliegues. | 27 de junio de 2019 |

| | | |
|---|---|---|
| Ahora puede añadir etiquetas a los recursos de AWS CodePipeline | Ahora puedes usar el etiquetado para rastrear y administrar AWS CodePipeline recursos como canalizaciones, acciones personalizadas y webhooks. Se han agregado los siguientes temas nuevos: Etiquetar recursos, Usar etiquetas para controlar el acceso a CodePipeline los recursos, Etiquetar una canalización CodePipeline, Etiquetar una acción personalizada y Etiquetar un CodePipeline webhook. CodePipeline Los siguientes temas se han actualizado para mostrar cómo usar la CLI para etiquetar recursos: Crear una canalización (CLI), Crear una acción personalizada (CLI) y Crear un webhook para una GitHub fuente. | 15 de mayo de 2019 |

| | | |
|---|---|---|
| Ahora puede ver el historial de ejecución de acciones en AWS CodePipeline | Ahora puede ver detalles sobre las ejecuciones anteriores de todas las acciones de una canalización. Estos detalles incluyen las horas de inicio y finalización, la duración, el ID de ejecución de la acción, información sobre la ubicación de los artefactos de entrada y de salida e información sobre los recursos externos. El tema Visualización de los detalles y el historial de la canalización se ha actualizado para reflejar esta compatibilidad. | 20 de marzo de 2019 |
| AWS CodePipeline ahora es compatible con la publicación de aplicaciones en AWS Serverless Application Repository | Ahora puede crear una canalización CodePipeline que publique su aplicación sin servidor en. AWS Serverless Application Repository Un nuevo tutorial, Tutorial: Publicar aplicaciones en el AWS Serverless Application Repository, proporciona los pasos para crear y configurar una canalización a fin de entregar continuamente su aplicación sin servidor al. AWS Serverless Application Repository | 8 de marzo de 2019 |

| | | |
|---|---|---|
| AWS CodePipeline ahora admite acciones entre regiones en la consola | Ahora puede gestionar las acciones entre regiones en la AWS CodePipeline consola. El tema Agregar una acción entre regiones se ha actualizado con los pasos para agregar, editar o eliminar una acción que se encuentre en una región de AWS diferente a la de la canalización. Se han actualizado los temas Crear una canalización, Editar una canalización y Referencia de la estructura de la canalización de CodePipeline. | 14 de febrero de 2019 |
| AWS CodePipeline ahora es compatible con las implementaciones de Amazon S3 | Ahora puede crear una canalización CodePipeline que utilice Amazon S3 como proveedor de acciones de implementación. Un nuevo tutorial, Tutorial: Create a pipeline que utilice Amazon S3 como proveedor de despliegues, proporciona los pasos para implementar archivos de muestra en su bucket de Amazon S3 con CodePipeline. El tema Referencia de la estructura de la canalización de CodePipeline también se ha actualizado. | 16 de enero de 2019 |

| | | |
|---|---|---|
| AWS CodePipeline ahora es compatible con las implementaciones del Alexa Skills Kit | Ahora puedes usar CodePipeline un kit de habilidades de Alexa para el despliegue continuo de las habilidades de Alexa. Un nuevo tutorial, Tutorial: Crear una canalización que implemente una habilidad de Amazon Alexa, contiene los pasos para crear credenciales que permitan conectarse AWS CodePipeline a su cuenta de desarrollador del Alexa Skills Kit y, a continuación, crear una canalización que implemente una habilidad de muestra. El tema Referencia de la estructura de la canalización de CodePipeline se ha actualizado. | 19 de diciembre de 2018 |
| AWS CodePipeline ahora es compatible con los puntos de conexión de Amazon VPC con tecnología de AWS PrivateLink | Ahora puede conectarse directamente a AWS CodePipeline través de un punto de conexión privado en su VPC, manteniendo todo el tráfico dentro de su VPC y de la red. AWS Para obtener más información, consulte Uso CodePipeline con Amazon Virtual Private Cloud. | 6 de diciembre de 2018 |

| | | |
|---|---|---|
| AWS CodePipeline ahora admite acciones de origen y acciones de ECS-to-CodeDeploy despliegue de Amazon ECR | Ahora puede usar CodePipeline y CodeDeploy con Amazon ECR y Amazon ECS para el despliegue continuo de aplicaciones basadas en contenedores. Un nuevo tutorial, titulado Crear una canalización con una fuente e ECS-to-CodeDeploy implementación de Amazon ECR, contiene los pasos para usar la consola para crear una canalización que despliegue aplicaciones contenedoras almacenadas en un repositorio de imágenes en un clúster de Amazon ECS con enrutamiento de CodeDeploy tráfico. Se han actualizado los temas Crear una canalización y Referencia de la estructura de la canalización de CodePipeline . | 27 de noviembre de 2018 |

| | | |
|---|---|---|
| AWS CodePipeline ahora admite acciones interregionales en un proceso | En un tema nuevo, Añadir una acción interregional, se incluyen los pasos para utilizar AWS CLI o CloudFormation añadir una acción que se encuentre en una región distinta de la de tu proceso. Se han actualizado los temas Crear una canalización, Editar una canalización y Referencia de la estructura de la canalización de CodePipeline . | 12 de noviembre de 2018 |
| AWS CodePipeline ahora se integra con Service Catalog | Ahora puede añadir Service Catalog como acción de implementación a su canalización. Esto le permite configurar una canalización para publicar actualizaciones del producto en Service Catalog al realizar un cambio en el repositorio de origen. El tema Integraciones se ha actualizado para reflejar esta compatibilidad con Service Catalog. Se han agregado dos tutoriales de Service Catalog a la sección Tutoriales de AWS CodePipeline. | 16 de octubre de 2018 |

| | | |
|---|---|---|
| AWS CodePipeline ahora se integra con AWS Device Farm | Ahora puedes añadirla AWS Device Farm como acción de prueba a tu canalización. De este modo puede configurar una canalización para probar aplicaciones móviles. El tema Integraciones se ha actualiza do para reflejar esta compatibi lidad con. AWS Device Farm Se han agregado dos tutoriale s de AWS Device Farm a la sección Tutoriales de AWS CodePipeline. | 19 de julio de 2018 |
| AWS CodePipeline Las notificaciones de actualización de la Guía del usuario ahora están disponibles a través de RSS | La versión HTML de la Guía del CodePipeline usuario ahora admite una fuente RSS de las actualizaciones que se documentan en la página del historial de actualizaciones de la documentación. La fuente RSS incluye las actualiza ciones realizadas a partir del 30 de junio de 2018. Las actualizaciones anunciadas anteriormente siguen estando disponibles en la página Historial de actualizaciones de la documentación. Utilice el botón RSS del panel del menú superior para suscribirse a la fuente. | 30 de junio de 2018 |

# Actualizaciones anteriores

En la siguiente tabla se describen los cambios importantes en cada versión de la Guía del CodePipeline usuario del 30 de junio de 2018 y anteriores.

| Cambio | Descripción | Fecha de modificación |
| --- | --- | --- |
| Usa webhooks para detectar los cambios de origen en GitHub las canalizaciones | Al crear o editar una canalización en la consola, CodePipeline ahora crea un webhook que detecta los cambios en el repositorio de GitHub origen y, a continuación, inicia la canalización. Para obtener información sobre cómo migrar tu canalización, consulta Cómo configurar tus GitHub canalizaciones para usar webhooks para la detección de cambios. Para obtener más información, consulte Inicio de la ejecución de una canalización en CodePipeline. | 1 de mayo de 2018 |
| Temas actualizados | Al crear o editar una canalización en la consola, CodePipeline ahora crea una regla de Amazon CloudWatch Events y un AWS CloudTrail registro que detecta los cambios en el bucket de origen de Amazon S3 y, a continuación, inicia la canalización. Para obtener más información sobre cómo migrar una canalización, consulte Acciones de origen y métodos de detección de cambios.<br><br>Se Tutorial: Crear una canalización simple (bucket de S3) ha actualizado para mostrar cómo se crean la regla y el seguimiento de Amazon CloudWatch Events al seleccionar una fuente de Amazon S3. Creación de una canalización, etapas y accionesy Editar una canalización en CodePipeline también se han actualizado.<br><br>Para obtener más información, consulte Inicie una canalización en CodePipeline. | 22 de marzo de 2018 |
| Tema actualizado | CodePipeline ya está disponible en Europa (París). Se ha actualizado el tema Cuotas en AWS CodePipeline. | 21 de febrero de 2018 |

| Cambio | Descripción | Fecha de modificación |
| --- | --- | --- |
| Temas actualizados | Ahora puede usar CodePipeline Amazon ECS para el despliegue continuo de aplicaciones basadas en contenedores. Cuando cree una canalización, puede seleccionar Amazon ECS como proveedor de implement ación. Un cambio en el código del repositorio de control de origen activa la canalización y esta crea una nueva imagen de Docker, la envía al registro de contenedores e implementa la imagen actualizada en un servicio de Amazon ECS.<br><br>Los temas Integraciones de productos y servicios con CodePipeline, Creación de una canalización, etapas y acciones y CodePipeline referencia de estructura de tuberías se han actualizado para reflejar esta compatibi lidad con &ECS;. | 12 de diciembre de 2017 |
| Temas actualizados | Al crear o editar una canalización en la consola, CodePipel ine ahora crea una regla de Amazon CloudWatch Events que detecta los cambios en el CodeCommit repositorio y, a continuación, inicia automáticamente la canalización. Para obtener más información sobre cómo migrar una canalizac ión existente, consulte Acciones de origen y métodos de detección de cambios.<br><br>Se Tutorial: Crear una canalización simple (CodeCommit repositorio) ha actualizado para mostrar cómo se crean la regla y el rol de Amazon CloudWatch Events al seleccion ar un CodeCommit repositorio y una sucursal. Creación de una canalización, etapas y accionesy Editar una canalizac ión en CodePipeline también se han actualizado.<br><br>Para obtener más información, consulte Inicie una canalización en CodePipeline. | 11 de octubre de 2017 |

| Cambio | Descripción | Fecha de modificación |
| --- | --- | --- |
| Temas nuevos y actualizados | CodePipeline ahora ofrece soporte integrado para las notificaciones de cambios de estado de la canalización a través de Amazon CloudWatch Events y Amazon Simple Notification Service (Amazon SNS). Se ha agregado un nuevo tutorial, Tutorial: configurar una regla de CloudWatch eventos para recibir notificaciones por correo electrónico sobre los cambios en el estado de los oleoductos. Para obtener más información, consulte Supervisión de CodePipeline eventos. | 8 de septiembre de 2017 |
| Temas nuevos y actualizados | Ahora puedes añadirlas CodePipeline como objetivo para las acciones de Amazon CloudWatch Events. Las reglas de Amazon CloudWatch Events se pueden configurar para detectar los cambios en la fuente, de modo que la canalización comience tan pronto como se produzcan esos cambios, o se pueden configurar para ejecutar la canalización programada. Se ha añadido información sobre la opción de configuración de la acción de PollForSourceChanges origen. Para obtener más información, consulte Inicie una canalización en CodePipeline. | 5 de septiembre de 2017 |
| Nuevas regiones de | CodePipeline ya está disponible en Asia Pacífico (Seúl) y Asia Pacífico (Bombay). Los temas Cuotas en AWS CodePipeline y Regiones y puntos de conexión se han actualizado. | 27 de julio de 2017 |
| Nuevas regiones de | CodePipeline ahora está disponible en EE. UU. Oeste (Norte de California), Canadá (Centro) y Europa (Londres). Los temas Cuotas en AWS CodePipeline y Regiones y puntos de conexión se han actualizado. | 29 de junio de 2017 |

| Cambio | Descripción | Fecha de modificación |
| --- | --- | --- |
| Temas actualizados | Ahora puede ver detalles acerca de las ejecuciones anteriores de una canalización, no solo la ejecución más reciente. Estos detalles incluyen horas de inicio y finalización, duración e ID de ejecución. Los detalles están disponibles para un máximo de 100 ejecuciones de canalización durante el periodo de 12 meses más reciente. Los temas Vea las canalizaciones y los detalles en CodePipeline, referencia de permisos y Cuotas en AWS CodePipeline se han actualizado para reflejarlo. | 22 de junio de 2017 |
| Tema actualizado | Nouvola se ha agregado a la lista de acciones disponibles en Integraciones de acciones de prueba. | 18 de mayo de 2017 |
| Temas actualizados | En el AWS CodePipeline asistente, la página Paso 4: Beta ha cambiado su nombre a Paso 4: Implementar. El nombre predeterminado de la etapa creada en este paso ha cambiado de "Beta" a "Staging" (Ensayo). Numerosos temas y capturas de pantalla se han actualizado para reflejar estos cambios. | 7 de abril de 2017 |

| Cambio | Descripción | Fecha de modificación |
| --- | --- | --- |
| Temas actualizados | Ahora puedes añadirla AWS CodeBuild como acción de prueba a cualquier fase de una canalización. Esto te permite ejecutar pruebas unitarias AWS CodeBuild con tu código con mayor facilidad. Antes de esta versión, solo se podían AWS CodeBuild ejecutar pruebas unitarias como parte de una acción de compilación. Una acción de compilación requiere un artefacto de salida de la compilación, algo que las pruebas unitarias normalmente no generan.<br><br>Los temas Integraciones de productos y servicios con CodePipeline y Editar una canalización en CodePipeline se CodePipeline referencia de estructura de tuberías han actualizado para reflejar esta compatibilidad con AWS CodeBuild. | 8 de marzo de 2017 |
| Temas nuevos y actualizados | El índice se ha reorganizado incluyendo secciones para las canalizaciones, acciones y transiciones de etapa. Se ha añadido una nueva sección de CodePipeline tutoriales. A fin de facilitar su uso, Integraciones de productos y servicios con CodePipeline se ha dividido en temas más breves.<br><br>Una nueva sección, Autorización y control de acceso, proporciona información completa sobre el uso AWS Identity and Access Management (IAM) y ayuda CodePipeline a proteger el acceso a sus recursos mediante el uso de credenciales. Estas credenciales proporcionan los permisos necesarios para acceder a AWS los recursos, como colocar y recuperar artefactos de los buckets de Amazon S3 e integrar OpsWorks pilas en sus canalizaciones. | 8 de febrero de 2017 |

| Cambio | Descripción | Fecha de modificación |
| --- | --- | --- |
| Nueva región de | CodePipeline ya está disponible en Asia Pacífico (Tokio). Los temas Cuotas en AWS CodePipeline y Regiones y puntos de conexión se han actualizado. | 14 de diciembre de 2016 |
| Nueva región de | CodePipeline ya está disponible en Sudamérica (São Paulo). Los temas Cuotas en AWS CodePipeline y Regiones y puntos de conexión se han actualizado. | 7 de diciembre de 2016 |
| Temas actualizados | Ahora puedes añadirla AWS CodeBuild como acción de construcción a cualquier fase de una canalización. AWS CodeBuild es un servicio de compilación en la nube totalmente gestionado que compila el código fuente, ejecuta pruebas unitarias y produce artefactos listos para su despliegue. Puede usar un proyecto de compilación existente o crear uno en la CodePipeline consola. La salida del proyecto de compilación puede entonces implement arse como parte de una canalización.<br><br>Los temas Integraciones de productos y servicios con CodePipeline Autenticación y control de acceso se CodePipeline referencia de estructura de tuberías han actualizado para reflejar esta compatibilidad con AWS CodeBuild. Creación de una canalización, etapas y acciones<br><br>Ahora puede usarlo CodePipeline con el modelo CloudFormation de aplicaciones AWS sin servidor para entregar sus aplicaciones sin servidor de forma continua. El tema Integraciones de productos y servicios con CodePipeline se ha actualizado para reflejar esta compatibilidad.<br><br>Integraciones de productos y servicios con CodePipelinese ha reorganizado para agrupar AWS y asociar las ofertas por tipo de acción. | 1 de diciembre de 2016 |

| Cambio | Descripción | Fecha de modificación |
| --- | --- | --- |
| Nueva región de | CodePipeline ya está disponible en Europa (Frankfurt). Los temas Cuotas en AWS CodePipeline y Regiones y puntos de conexión se han actualizado. | 16 de noviembre de 2016 |
| Temas actualizados | AWS CloudFormation ahora se puede seleccionar como proveedor de despliegues en procesos, lo que le permite tomar medidas en relación con las CloudFormation pilas y los conjuntos de cambios como parte de la ejecución de una canalización. Los temas Integraciones de productos y servicios con CodePipeline Autenticación y control de acceso se CodePipeline referencia de estructura de tuberías han actualizado para reflejar esta compatibilidad con. Creación de una canalización, etapas y acciones AWS CloudFormation | 3 de noviembre de 2016 |
| Nueva región de | CodePipeline ya está disponible en la región de Asia Pacífico (Sídney). Los temas Cuotas en AWS CodePipeline y Regiones y puntos de conexión se han actualizado. | 26 de octubre de 2016 |
| Nueva región de | CodePipeline ya está disponible en Asia Pacífico (Singapur). Los temas Cuotas en AWS CodePipeline y Regiones y puntos de conexión se han actualizado. | 20 de octubre de 2016 |
| Nueva región de | CodePipeline ya está disponible en la región EE.UU. Este (Ohio). Los temas Cuotas en AWS CodePipeline y Regiones y puntos de conexión se han actualizado. | 17 de octubre de 2016 |
| Tema actualizado | Creación de una canalización, etapas y acciones se ha actualizado para reflejar la posibilidad de mostrar los identificadores de versión de las acciones personalizadas en las listas Source provider (Proveedor de código fuente) y Build provider (Proveedor de compilación). | 22 de septiembre de 2016 |

| Cambio | Descripción | Fecha de modificación |
| --- | --- | --- |
| Tema actualizado | La sección Incorporación de una acción de aprobación manual a una etapa se ha actualizado para reflejar una mejora que permite a los revisores de una acción de aprobación abrir el formulario Approve or reject the revision directamente desde una notificación por correo electrónico. | 14 de septiembre de 2016 |
| Temas nuevos y actualizados | Un nuevo tema que describe el modo de ver detalles sobre los cambios de código que fluyen actualmente por una canalización de publicación de software. Tener acceso rápido a esta información puede resultar útil para consultar acciones de aprobación manuales o solucionar problemas con la canalización.<br><br>Una nueva sección, Monitorear canalizaciones, ofrece un lugar centralizado donde encontrar todos los temas relacionados con el estado y la evolución de las canalizaciones. | 08 de septiembre de 2016 |
| Temas nuevos y actualizados | Una nueva sección, Incorporación de una acción de aprobación manual a una etapa, ofrece información sobre cómo configurar y utilizar acciones de aprobación manual en las canalizaciones. Los temas de esta sección ofrecen información conceptual sobre el proceso de aprobación, instrucciones para configurar los permisos de IAM requeridos, crear acciones de aprobación y aprobar o rechazar acciones de aprobación, y ejemplos de los datos JSON generados cuando se alcanza una acción de aprobación en una canalización. | 06 de julio de 2016 |
| Nueva región de | CodePipeline ya está disponible en la región Europa (Irlanda). Los temas Cuotas en AWS CodePipeline y Regiones y puntos de conexión se han actualizado. | 23 de junio de 2016 |

| Cambio | Descripción | Fecha de modificación |
| --- | --- | --- |
| Nuevo tema | Se ha añadido un nuevo tema, , para describir el modo de volver a intentar una acción fallida o un grupo de acciones fallidas en paralelo en una etapa. | 22 de junio de 2016 |
| Temas actualizados | Se han actualizado varios temasCreación de una canalización, etapas y acciones, como la autenticación y el control de accesoCodePipeline referencia de estructura de tuberías, yIntegraciones de productos y servicios con CodePipeline, para reflejar la compatibilidad con la configuración de un proceso para implementar código junto con libros de cocina y aplicaciones personalizados de Chef creados en AWS OpsWorks. CodePipeline Actualmente, el soporte para solo OpsWorks está disponible en la región EE. UU. Este (Virginia del Norte) (us-east-1). | 2 de junio de 2016 |
| Temas nuevos y actualizados | Se ha agregado un nuevo tema, Tutorial: Crear una canalización simple (CodeCommit repositorio). En este tema se proporciona un ejemplo de tutorial que muestra cómo utilizar un CodeCommit repositorio y una sucursal como ubicación de origen de una acción de origen en una canalización. Se han actualizado varios otros temas para reflejar esta integración con CodeCommit, incluidos la autenticación y el control de acceso, Integraciones de productos y servicios con CodePipelineTutorial: Crear una canalización de cuatro etapas, ySolución de problemas CodePipeline. | 18 de abril de 2016 |
| Nuevo tema | Se ha agregado un nuevo tema, Invoca un AWS Lambda función en una canalización en CodePipeline. Este tema contiene ejemplos de AWS Lambda funciones y pasos para añadir funciones Lambda a las canalizaciones. | 27 de enero de 2016 |
| Tema actualizado | Se ha añadido una nueva sección a Autenticación y control de acceso, Políticas basadas en recursos. | 22 de enero de 2016 |

| Cambio | Descripción | Fecha de modificación |
| --- | --- | --- |
| Nuevo tema | Se ha agregado un nuevo tema, Integraciones de productos y servicios con CodePipeline. La información sobre las integraciones con socios y otros se Servicios de AWS ha trasladado a este tema. También se han agregado enlaces a blogs y vídeos. | 17 de diciembre de 2015 |
| Tema actualizado | Se ha añadido una sección sobre detalles de la integración con Solano CI al tema Integraciones de productos y servicios con CodePipeline. | 17 de noviembre de 2015 |
| Tema actualizado | El CodePipeline complemento para Jenkins ahora está disponible a través del administrador de complementos de Jenkins como parte de la biblioteca de complementos para Jenkins. Los pasos para instalar el complemento se han actualizado en Tutorial: Crear una canalización de cuatro etapas. | 9 de noviembre de 2015 |
| Nueva región de | CodePipeline ya está disponible en la región de EE. UU. Oeste (Oregón). Se ha actualizado el tema Cuotas en AWS CodePipeline. Se han agregado enlaces a Regiones y puntos de conexión. | 22 de octubre de 2015 |
| Nuevo tema | Se han agregado dos nuevos temas,Configurar el cifrado del lado del servidor para los artefactos almacenados en Amazon S3 para CodePipeline y Cree una canalización en la CodePipeline que se utilicen los recursos de otra AWS inscrita. Se ha añadido una nueva sección a Autenticación y control de acceso. | 25 de agosto de 2015 |
| Tema actualizado | El tema Crea y añade una acción personalizada en CodePipeline se ha actualizado para reflejar los cambios en la estructura, como `inputArtifactDetails` y `outputArtifactDetails` . | 17 de agosto de 2015 |

| Cambio | Descripción | Fecha de modificación |
| --- | --- | --- |
| Tema actualizado | El tema Solución de problemas CodePipeline se ha actualizado con pasos revisados para la solución de problemas con el rol de servicio y Elastic Beanstalk. | 11 de agosto de 2015 |
| Tema actualizado | El tema Autenticación y control de acceso se ha actualizado con los últimos cambios en el rol de servicio de CodePipeline. | 6 de agosto de 2015 |
| Nuevo tema | Se ha agregado el tema Solución de problemas CodePipeline. Se han agregado pasos actualizados para los roles de IAM y Jenkins en Tutorial: Crear una canalización de cuatro etapas. | 24 de julio de 2015 |
| Actualización de tema | Se han agregado pasos actualizados para descargar los archivos de ejemplo en Tutorial: Crear una canalización simple (bucket de S3) y Tutorial: Crear una canalización de cuatro etapas. | 22 de julio de 2015 |
| Actualización de tema | Se ha agregado una solución temporal para los problemas de descarga de los archivos de ejemplo en Tutorial: Crear una canalización simple (bucket de S3). | 17 de julio de 2015 |
| Actualización de tema | Se ha agregado un enlace en Cuotas en AWS CodePipeline para remitir a la información sobre los límites que pueden cambiarse. | 15 de julio de 2015 |
| Actualización de tema | Se ha actualizado la sección de políticas administradas en Autenticación y Control de acceso. | 10 de julio de 2015 |
| Versión pública inicial | Esta es la primera versión pública de la Guía CodePipeline del usuario. | 9 de julio de 2015 |

# Referencia de características de CodePipeline

Esta sección es una referencia de alto nivel sobre las características de la documentación de CodePipeline. Para obtener información general conceptual sobre la estructura de la canalización, consulte CodePipeline referencia de estructura de tuberías.

Esta tabla se actualiza periódicamente con información de referencia sobre las características.

| Fecha de actualización de las tablas | Característica | Guía del usuario de CodePipeline | Guía de la API de CodePipeline | CloudFormationReferencia de de | Referencia de AWS CDK: constructos de L1 |
|---|---|---|---|---|---|
| 15 de octubre de 2024 | Reintento automático en caso de fallo de una etapa | Configuración de una etapa para el reintento automático en caso de fallo | RetryConfiguration | | |
| 15 de octubre de 2024 | Resultado Omitir para las condiciones beforeEntry | Creación de condiciones de entrada con el resultado Omitir y la regla `VariableCheck` (consola) | FailureConditions | | |
| 3 de octubre de 2024 | Acción de Comandos con la nueva | Referencia de la acción de Comandos | ActionDeclaration | | |

| Fecha de actualización de las tablas | Característica | Guía del usuario de CodePipeline | Guía de la API de CodePipeline | CloudFormationReferencia de de | Referencia de AWS CDK: constructos de L1 |
|---|---|---|---|---|---|
| | categoría de computación | | | | |
| 28 de agosto de 2024 | Tipo beforeEntry para condiciones de etapa | Configuración de las condiciones de una etapa | BeforeEntryConditions | AWS::CodePipeline::Pipeline BeforeEntryConditions | beforeEntry |
| 28 de agosto de 2024 | Tipo onSuccess para condiciones de etapa | Configuración de las condiciones de una etapa | SuccessConditions | AWS::CodePipeline::Pipeline SuccessConditions | onSuccess |
| 26 de abril de 2024 | Reversiones de etapa y tipo onFailure para condiciones de etapa | Configuración de la reversión de etapas | RollbackStage | onFailure | onFailure |
| 8 de febrero de 2024 | Modos de ejecución PARALELO y EN COLA | Configuración o cambio del modo de ejecución de una canalización | PipelineExecution | Execution Mode | Modo de ejecución |

| Fecha de actualización de las tablas | Característica | Guía del usuario de CodePipeline | Guía de la API de CodePipeline | CloudFormationReferencia de de | Referencia de AWS CDK: constructos de L1 |
|---|---|---|---|---|---|
| 17 de noviembre de 2023 | Anulaciones de origen | Iniciar una canalización con una anulación de revisión de código fuente | SourceRevisionOverride | N/A | N/A |
| 24 de octubre de 2023 | Tipos de canalización | ¿Qué tipo de canalización es el adecuado para mí? | PipelineDeclaration | PipelineType | enum PipelineType |
| 24 de octubre de 2023 | Variables a nivel de canalización | Tutorial: Uso de variables a nivel de canalización | PipelineVariable | AWS::CodePipeline::Pipeline::PipelineVariableDeclaration | Variables a nivel de canalización |

| Fecha de actualización de las tablas | Característica | Guía del usuario de CodePipeline | Guía de la API de CodePipeline | CloudFormationReferencia de de | Referencia de AWS CDK: constructos de L1 |
|---|---|---|---|---|---|
| 24 de octubre de 2023 | Desencadena y filtra rutas de archivo, ramificaciones y solicitudes de extracción | Automatización del inicio de las canalizaciones mediante desencadenadores y filtrado<br>Tutorial: filtrado de solicitudes de extracción por nombres de ramificación para iniciar una canalización (tipo V2) | ActionDeclaration | AWS::CodePipeline::Pipeline::PipelinePipelineTriggerDeclaration | Desencadenador |
| 17 de octubre de 2023 | Reintentar una etapa | Configuración del reintento de una etapa fallida o de acciones fallidas | ActionDeclaration | N/A | N/A |

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.