Manual do usuário

AWS CodePipeline



Versão da API 2015-07-09

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

AWS CodePipeline: Manual do usuário

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

As marcas comerciais e imagens comerciais da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestigie a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que é CodePipeline? 1
Integração e entrega contínuas 1
O que eu posso fazer com isso CodePipeline? 2
Uma rápida olhada em CodePipeline 3
Como faço para começar CodePipeline? 4
Conceitos 4
Pipelines 5
Execuções de pipeline 6
Operadores de estágio 8
Execuções de ação 8
Tipos de execução 9
Tipos de ação 9
Artefatos 9
Revisões de origem 10
Acionadores 10
Variáveis 11
Condições 11
Regras 12
DevOps exemplo de pipeline 12
Como funcionam as execuções de pipeline 14
Como as execuções de pipeline são iniciadas 15
Como as revisões de origem são processadas em execuções de pipeline 15
Como as substituições de fonte funcionam com o transformador de entrada EventBridge 16
Como as execuções de pipeline são interrompidas 17
Como as execuções são processadas no modo SUPERSEDED 21
Como as execuções são processadas no modo QUEUED 22
Como as execuções são processadas no modo PARALLEL 24
Gerenciar o fluxo do pipeline 24
Artefatos de entrada e saída 27
Como funcionam as condições do estágio? 30
Regras para condições de estágio 33
Tipos de pipeline 34
Que tipo de pipeline é ideal para mim? 34
Introdução 40

Inscreva-se para um Conta da AWS 40
Aplique uma política gerenciada para acesso administrativo ao CodePipeline 41
Instale o AWS CLI 42
Abra o console para CodePipeline 43
Próximas etapas 43
Integrações de produtos e serviços 44
Integrações com tipos de CodePipeline ação 44
Integrações de ações de origem 44
Integrações de ações de compilação 52
Integrações de ações de teste 54
Implantar integrações de ações 56
Integração da ação de aprovação ao Amazon Simple Notification Service 63
Integrações de ações de invocação 63
Integrações gerais com CodePipeline 65
Exemplos da comunidade 68
Publicações no blog 68
Tutoriais 72
Tutorial: Implante em instâncias do Amazon EC2 com CodePipeline 73
Pré-requisitos 73
Etapa 1: Criar instâncias do Amazon EC2 Linux 74
Etapa 2: adicionar permissões de bucket do artefato à função de instância do EC2 76
Etapa 3: Adicionar um arquivo de script ao repositório 77
Etapa 4: Criação do pipeline 78
Etapa 5: teste seu pipeline 82
Tutorial: crie e envie uma imagem do Docker para o Amazon ECR com CodePipeline (tipo V2) 83
Pré-requisitos 84
Etapa 1: Adicionar um Dockerfile ao repositório de origem 84
Etapa 2: Adicionar um arquivo imagedefinitions.json ao repositório de origem 86
Etapa 3: Criação do pipeline 87
Etapa 4: Testar o pipeline 92
Tutorial: Implante no Amazon EKS com CodePipeline 93
Pré-requisitos 93
Etapa 1: (Opcional) Criar um cluster no Amazon EKS 94
Etapa 2: Configurar o cluster privado no Amazon EKS 96
Etapa 3: atualizar a política da função de CodePipeline serviço no IAM 97

Etapa 4: criar uma entrada de acesso para a função CodePipeline de serviço 100
Etapa 5: Crie um repositório de origem e adicione os arquivos de configuração do gráfico Helm 101
Etapa 6: Criação do pipeline 101
Tutorial: Criar um pipeline que executa comandos com computação (tipo V2) 104
Pré-requisitos 104
Etapa 1: Crie arquivos de origem e envie para seu GitHub repositório 104
Etapa 2: Criar o pipeline 105
Etapa 3: executar o pipeline e verificar os comandos de compilação 109
Tutorial: Usar tags do Git para iniciar o pipeline 110
Pré-requisitos 111
Etapa 1: abrir CloudShell e clonar seu repositório 111
Etapa 2: Criar um pipeline para acionar as tags do Git 112
Etapa 3: Marcar as confirmações para lançamento 116
Etapa 4: Lançar alterações e visualizar logs 117
Tutorial: Filtrar nomes de ramificação para solicitações pull a fim de iniciar o pipeline (tipo V2) 118
Pré-requisitos 118
Etapa 1: criar um pipeline para iniciar uma solicitação pull para ramificações especificadas 119
Etapa 2: crie e mescle uma pull request GitHub.com para iniciar suas execuções de pipeline 121
Tutorial: Usar variáveis no nível do pipeline 123
Pré-requisitos 123
Etapa 1: Criar seu pipeline e compilar o projeto 123
Etapa 2: Lançar alterações e visualizar logs 127
Tutorial: Criar um pipeline simples (bucket do S3) 127
Criar um bucket do S3. 129
Crie instâncias do Windows Server Amazon EC2 e instale o agente CodeDeploy 131
Crie um aplicativo no CodeDeploy 133
Criar o primeiro pipeline 135
Adicionar outro estágio 138
Habilitar e desabilitar transições entre estágios 145
Limpar os recursos 146
Tutorial: criar um pipeline simples (CodeCommit repositório) 147
Crie um CodeCommit repositório 148

Fazer download, confirmar e enviar o código ... 149
Crie uma instância Linux do Amazon EC2 e instale o agente CodeDeploy ... 152
Crie um aplicativo no CodeDeploy ... 154
Criar o primeiro pipeline ... 155
Atualize o código em seu CodeCommit repositório ... 159
Limpar os recursos ... 161
Outras fontes de leitura ... 161
Tutorial: Criar um pipeline de quatro estágios ... 162
Concluir os pré-requisitos ... 163
Criar um pipeline ... 168
Adicionar mais estágios ... 169
Limpar os recursos ... 173
Tutorial: configurar uma regra de CloudWatch eventos para receber notificações por e-mail sobre mudanças no estado do pipeline ... 174
Configurar uma notificação por e-mail usando o Amazon SNS ... 175
Crie uma regra de notificação de CloudWatch eventos para CodePipeline ... 176
Limpar os recursos ... 177
Tutorial: crie e teste um aplicativo Android com AWS Device Farm ... 178
Configure CodePipeline para usar seus testes do Device Farm ... 179
Tutorial: teste um aplicativo iOS com AWS Device Farm ... 184
Configure CodePipeline para usar seus testes do Device Farm (exemplo do Amazon S3) ... 185
Tutorial: Criar um pipeline que realiza a implantação no Service Catalog ... 190
Opção 1: Implantar no Service Catalog sem um arquivo de configuração ... 192
Opção 2: Implantar no Service Catalog com um arquivo de configuração ... 197
Tutorial: criar um pipeline com AWS CloudFormation ... 202
Exemplo 1: criar um AWS CodeCommit oleoduto com AWS CloudFormation ... 202
Exemplo 2: Crie um pipeline do Amazon S3 com AWS CloudFormation ... 204
Tutorial: criar um pipeline que usa variáveis de ações de AWS CloudFormation implantação ... 208
Pré-requisitos: criar um AWS CloudFormation função de serviço e um CodeCommit repositório ... 209
Etapa 1: Baixe, edite e faça o upload da amostra AWS CloudFormation modelo ... 210
Etapa 2: Criar o pipeline ... 211
Etapa 3: adicionar um CloudFormation ação de implantação para criar o conjunto de alterações ... 214
Etapa 4: Adicionar uma ação de aprovação manual ... 215

Etapa 5: adicionar uma ação CloudFormation de implantação para executar o conjunto de alterações ... 215
Etapa 6: Adicionar uma ação CloudFormation de implantação para excluir a pilha ... 216
Tutorial: Implantação padrão do Amazon ECS com CodePipeline ... 217
Pré-requisitos ... 218
Etapa 1: Adicionar um arquivo de especificação de compilação ao repositório de origem ... 221
Etapa 2: Criar uma pipeline de implantação contínua ... 223
Etapa 3: adicionar permissões do Amazon ECR à função CodeBuild ... 225
Etapa 4: Testar o pipeline ... 225
Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR ... 226
Pré-requisitos ... 228
Etapa 1: Criar uma imagem e enviá-la a um repositório do Amazon ECR ... 228
Etapa 2: Crie arquivos de definição de tarefas e de AppSpec origem e envie para um CodeCommit repositório ... 229
Etapa 3: Criar o Application Load Balancer e os grupos de destino ... 233
Etapa 4: Criar um cluster e um serviço do Amazon ECS ... 236
Etapa 5: Crie seu CodeDeploy aplicativo e grupo de implantação (plataforma de computação ECS) ... 239
Etapa 6: Criar o pipeline ... 240
Etapa 7: Realizar uma alteração no pipeline e verificar a implantação ... 245
Tutorial: Criar um pipeline que implanta uma skill do Amazon Alexa ... 245
Pré-requisitos ... 246
Etapa 1: Criar um perfil de segurança do LWA dos serviços de desenvolvedor da Alexa ... 246
Etapa 2: Crie arquivos de origem de habilidades da Alexa e envie para seu repositório CodeCommit ... 246
Etapa 3: Usar os comandos da CLI do ASK para criar um token de atualização ... 248
Etapa 4: Criar o pipeline ... 249
Etapa 5: Realizar uma alteração em qualquer arquivo de origem e verificar a implantação ... 251
Tutorial: Criar um pipeline que usa o Amazon S3 como um provedor de implantação ... 252
Opção 1: Implantar arquivos estáticos de sites no Amazon S3 ... 253
Opção 2: Implantar arquivos de arquivamento compilados no Amazon S3 a partir de um bucket de origem do S3 ... 258
Tutorial: Publique aplicativos no AWS Serverless Application Repository ... 264
Antes de começar ... 265
Etapa 1: Criar um arquivo buildspec.yml ... 265

Etapa 2: Criar e configurar o pipeline 265
Etapa 3: Implantar o aplicativo de publicação 268
Etapa 4: Criar a ação de publicação 268
Tutorial: implantações da função Lambda com CodePipeline 269
Pré-requisitos 269
Etapa 1: Criar a função do Lambda de amostra 270
Etapa 2: Carregar o arquivo de função no repositório 271
Etapa 3: Criação do pipeline 271
Etapa 4: Testar o pipeline 274
Saiba mais 275
Tutorial: Usar variáveis com ações de invocação do Lambda 275
Pré-requisitos 276
Etapa 1: criar uma função do Lambda 276
Etapa 2: Adicionar uma ação de invocação do Lambda e uma ação de aprovação manual ao pipeline 279
Tutorial: Use uma AWS Step Functions ação de invocação 281
Pré-requisito: criar ou escolher um pipeline simples 281
Etapa 1: Criar a máquina de estado de exemplo 282
Etapa 2: Adicionar uma ação de invocação do Step Functions ao pipeline 282
Tutorial: Crie um pipeline que use AppConfig como provedor de implantação 284
Pré-requisitos 284
Etapa 1: Crie seus AWS AppConfig recursos 284
Etapa 2|: Fazer upload de arquivos para o bucket de origem do S3 285
Etapa 3: Criar o pipeline 286
Etapa 4: Realizar uma alteração em qualquer arquivo de origem e verificar a implantação .. 288
Tutorial: use o clone completo com uma fonte de GitHub pipeline 288
Pré-requisitos 289
Etapa 1: Criar um arquivo README 289
Etapa 2: Criar seu pipeline e criar o projeto 290
Etapa 3: atualizar a política CodeBuild de função de serviço para usar conexões 294
Etapa 4: Visualizar os comandos do repositório na saída da compilação 294
Tutorial: use o clone completo com uma fonte de CodeCommit pipeline 294
Pré-requisitos 295
Etapa 1: Criar um arquivo README 295
Etapa 2: Criar seu pipeline e criar o projeto 296
Etapa 3: atualizar a política da função CodeBuild de serviço para clonar o repositório 299

Etapa 4: Visualizar os comandos do repositório na saída da compilação 299
Tutorial: criar um pipeline com ações AWS CloudFormation StackSets de implantação 300
Pré-requisitos 301
Etapa 1: faça o upload da amostra AWS CloudFormation modelo e arquivo de parâmetros 301
Etapa 2: Criar o pipeline 303
Etapa 3: Visualizar a implantação inicial 306
Etapa 4: adicionar uma CloudFormationStackInstances ação 306
Etapa 5: Visualizar os recursos do conjunto de pilhas para a implantação 307
Etapa 6: Fazer uma atualização no conjunto de pilhas 308
Crie uma regra de verificação de variável para um pipeline como uma condição de entrada 308
Pré-requisitos 309
Etapa 1: Crie um arquivo de origem de amostra e adicione-o ao seu GitHub repositório 310
Etapa 2: Criar o pipeline 310
Etapa 2: editar o estágio de compilação para adicionar a condição e a regra 314
Etapa 3: executar o pipeline e visualizar as variáveis resolvidas 315
Melhores práticas e casos de uso 318
Exemplos de como usar CodePipeline 318
Use CodePipeline com o Amazon S3, AWS CodeCommit e AWS CodeDeploy 318
Use CodePipeline com provedores de ação terceirizados (GitHub e Jenkins) 319
Use CodePipeline para compilar, criar e testar código com CodeBuild 319
Use CodePipeline com o Amazon ECS para entrega contínua de aplicativos baseados em contêineres para a nuvem 320
Use CodePipeline com o Elastic Beanstalk para entrega contínua de aplicativos web para a nuvem 320
Use CodePipeline com AWS Lambda para entrega contínua Lambda-based e sem servidor de aplicativos 320
Use CodePipeline com CloudFormation modelos para entrega contínua na nuvem 320
Use CodePipeline com a Amazon VPC 321
Disponibilidade 321
Crie um VPC endpoint para CodePipeline 322
Solucionar problemas da configuração da VPC 323
Defina CI/CD pipelines com estágios e ações 324
Criar um pipeline, estágios e ações 325
Criar um pipeline personalizado (console) 326
Criar um pipeline (CLI) 339

Criar um pipeline usando modelos estáticos 346
Editar um pipeline 352
Editar um pipeline (console) 353
Editar um pipeline (AWS CLI) 356
Visualizar pipelines e detalhes 361
Visualizar pipelines (console) 361
Visualizar detalhes da ação em um pipeline (console) 366
Visualizar o ARN do pipeline e o ARN do perfil de serviço (console) 369
Visualizar detalhes e histórico do pipeline (CLI) 370
Exibir resultados das regras aplicadas às condições de estágio no histórico de execuções . 371
Excluir um pipeline 374
Excluir um pipeline (console) 374
Excluir um pipeline (CLI) 374
Criar um pipeline que usa recursos de outra conta 376
Pré-requisito: criar um AWS KMS chave de criptografia 378
Etapa 1: Configurar as políticas e funções da conta 379
Etapa 2: Editar o pipeline 386
Migrar pipelines de sondagem para usar a detecção de alterações baseada em eventos 389
Como migrar os pipelines de sondagem 389
Visualizar os pipelines de sondagem em sua conta 391
Migre canais de votação com uma fonte CodeCommit 396
Migrar pipelines de sondagem com uma origem do S3 habilitada para eventos 419
Migre canais de votação com uma origem e uma trilha do S3 CloudTrail 438
Migre canais de pesquisa para uma ação de origem GitHub (por meio do aplicativo OAuth) para conexões 474
Migre canais de pesquisa para uma ação de origem GitHub (por meio do aplicativo OAuth) para webhooks 477
Crie a função CodePipeline de serviço 495
Crie a função CodePipeline de serviço (console) 496
Crie a função CodePipeline de serviço (CLI) 496
Marcando atributos 499
Marcar um pipeline 500
Marcar pipelines (console) 500
Marcar pipelines (CLI) 502
Criar uma regra de notificação 504
..... 508

Conecte-se a provedores de origem primários usando ações de origem 511
Ações de origem do Amazon ECR e EventBridge 511
Crie uma EventBridge regra para uma fonte do Amazon ECR (console) 512
Crie uma EventBridge regra para uma fonte do Amazon ECR (CLI) 514
Crie uma EventBridge regra para uma fonte do Amazon ECR (CloudFormation modelo) 518
Conectando-se às ações de origem do Amazon S3 que usam eventos EventBridge 523
Criar pipelines com uma origem do S3 habilitada para eventos (CLI) 524
Crie pipelines com uma fonte S3 habilitada para eventos (CloudFormation modelo) 528
Conectando-se às ações de origem do Amazon S3 que usam EventBridge e AWS CloudTrail 543
Crie uma EventBridge regra para uma fonte do Amazon S3 (console) 544
Crie uma EventBridge regra para uma fonte (CLI) do Amazon S3 548
Crie uma EventBridge regra para uma fonte do Amazon S3 (CloudFormation modelo) 554
CodeCommit ações de origem e EventBridge 567
Crie uma EventBridge regra para uma CodeCommit fonte (console) 568
Crie uma EventBridge regra para uma CodeCommit fonte (CLI) 570
Crie uma EventBridge regra para uma CodeCommit fonte (CloudFormation modelo) 575
Adicione provedores de origem terceirizados aos pipelines usando AWS CodeConnections 584
DevOps Conexões Azure 584
Criar uma conexão com o Azure DevOps (console) 586
Crie uma conexão com o Azure DevOps (CLI) 590
Conexões do Bitbucket Cloud 592
Criar uma conexão com o Bitbucket Cloud (console) 594
Criar uma conexão com o Bitbucket Cloud (CLI) 597
GitHub conexões 599
Crie uma conexão com GitHub (console) 601
Crie uma conexão com GitHub (CLI) 604
GitHub Conexões do Enterprise Server 606
Crie uma conexão com o GitHub Enterprise Server (console) 607
Crie um host e uma conexão com o GitHub Enterprise Server (CLI) 611
GitLab.com conexões 614
Crie uma conexão com GitLab.com (console) 617
Crie uma conexão com GitLab.com (CLI) 621
Conexões para GitLab autogerenciamento 623
Crie uma conexão com o GitLab autogerenciado (console) 625
Crie um host e uma conexão com o GitLab autogerenciado (CLI) 628

Usar uma conexão compartilhada com outra conta ... 631
Automatizar a inicialização de pipelines usando gatilhos e filtragem ... 632
Considerações sobre filtros de gatilho ... 634
Eventos pull request para acionadores por provedor ... 635
Exemplos de filtros de gatilho ... 636
Adicionar gatilho em code push sem filtro ... 645
Adicionar gatilho com tipos de eventos code push ou pull request ... 645
Adicione filtros para tipos de evento push e pull request (console) ... 647
Adicionar filtros para tipos de evento push e pull request (CLI) ... 649
Adicione filtros para os tipos de eventos de push e pull request (CloudFormation modelos) ... 652
Adicionar acionador para desativar a detecção de alterações ... 653
Iniciar e parar pipelines manualmente ... 655
Inicie um funil em CodePipeline ... 655
Iniciar um pipeline manualmente ... 657
Iniciar um pipeline de acordo com uma programação ... 658
Iniciar um pipeline com uma substituição da revisão de origem ... 662
Interromper a execução de um pipeline ... 665
Interromper a execução de um pipeline (console) ... 666
Interromper uma execução de entrada (console) ... 669
Interromper a execução de um pipeline (CLI) ... 670
Interromper uma execução de entrada (CLI) ... 671
Exibir histórico e definir o modo para execuções de pipeline ... 673
Visualizar execuções ... 673
Visualizar o histórico de execução do pipeline (console) ... 673
Visualizar o status de execução (console) ... 675
Visualizar uma execução de entrada (console) ... 677
Visualizar revisões de origem de execução do pipeline (console) ... 678
Visualizar execuções da ação (console) ... 680
Visualizar informações sobre artefatos e armazenamento de artefatos da ação (console) ... 681
Visualizar detalhes e histórico do pipeline (CLI) ... 681
Definir ou alterar o modo de execução do pipeline ... 693
Considerações para visualizar modos de execução ... 694
Considerações para alternar entre os modos de execução ... 697
Definir ou alterar o modo de execução do pipeline (console) ... 698
Definir o modo de execução do pipeline (CLI) ... 699

Reverter ou repetir os estágios ... 703
Configurar a repetição do estágio para um estágio com falha ou ações com falha ... 703
Considerações para repetição de estágio ... 704
Repetir manualmente um estágio com falha ... 704
Configurar a repetição automática em caso de falha para um estágio ... 709
Configuração da reversão de estágio ... 714
Considerações para reversões ... 715
Reverter um estágio manualmente ... 715
Configurar um estágio para reversão automática ... 720
Exibir o status de reversão na lista de execução ... 724
Exibir detalhes do status da reversão ... 727
Configurar condições para um estágio ... 732
Casos de uso de condições de estágio ... 733
Considerações sobre resultados configurados para condições de estágio ... 733
Considerações sobre regras configuradas para condições de estágio ... 734
Criar condições de entrada ... 735
Criação de condições de entrada - exemplo de CloudWatchAlarm regra (console) ... 735
Criar condições de entrada com resultado Ignorar e regra `VariableCheck` (console) ... 736
Criar condições de entrada (CLI) ... 738
Criar condições de entrada (CFN) ... 740
Criar condições “em caso de falha” ... 741
Criar condições “em caso de falha” (console) ... 741
Criar condições onFailure com um exemplo de resultado de repetição (console) ... 742
Criar condições “em caso de falha” (CLI) ... 743
Criar condições “em caso de falha” (CFN) ... 745
Criar condições “em caso de sucesso” ... 746
Criar condições “em caso de sucesso” (console) ... 747
Criar condições “em caso de sucesso” (CLI) ... 748
Criar uma condição “em caso de sucesso” (CFN) ... 750
Excluir condições de estágio ... 751
Condições de estágio de substituição ... 752
Usar tipos de ação, ações personalizadas e ações de aprovação ... 754
Como trabalhar com tipos de ação ... 754
Solicitar um tipo de ação ... 756
Adicionar um tipo de ação disponível a um pipeline (console) ... 762
Visualizar um tipo de ação ... 764

Atualizar um tipo de ação ... 765
Criar uma ação personalizada para um pipeline ... 767
Criar uma ação personalizada ... 769
Criar um operador de trabalho para a ação personalizada ... 773
Adicionar uma ação personalizada a um pipeline ... 781
Marque uma ação personalizada em CodePipeline ... 784
Adicionar tags a uma ação personalizada ... 784
Visualizar tags para uma ação personalizada ... 785
Editar tags para uma ação personalizada ... 785
Remover tags de uma ação personalizada ... 786
Invocar uma função do Lambda em um pipeline ... 786
Etapa 1: Criar um pipeline ... 789
Etapa 2: Criar a função do Lambda ... 790
Etapa 3: adicionar a função Lambda a um pipeline no console CodePipeline ... 794
Etapa 4: Testar o pipeline com a função do Lambda ... 795
Etapa 5: Próximas etapas ... 796
Exemplo de evento JSON ... 797
Funções de exemplo adicionais ... 799
Adicionar uma ação de aprovação manual a uma fase ... 812
Opções de configuração de ações de aprovação manual ... 812
Visão geral da configuração e do fluxo de trabalho das ações de aprovação ... 814
Conceda permissões de aprovação a um usuário do IAM em CodePipeline ... 815
Conceder permissões do Amazon SNS a um perfil de serviço ... 818
Adicionar uma ação de aprovação manual ... 819
Aprovar ou rejeitar uma ação de aprovação ... 823
Formato dos dados JSON para notificações de aprovação manual ... 828
Adicionar uma ação entre regiões a um pipeline ... 829
Gerencie ações entre regiões em um pipeline (console) ... 831
Adicionar uma ação entre regiões a um pipeline (CLI) ... 834
Adicione uma ação entre regiões a um pipeline (CloudFormation) ... 839
Trabalhar com variáveis ... 842
Configurar ações de variáveis ... 843
Visualizar variáveis de saída ... 847
Exemplo: Usar variáveis em aprovações manuais ... 850
Exemplo: use uma BranchName variável com variáveis de CodeBuild ambiente ... 850
Trabalhar com transições de estágio ... 853

Habilitar ou desabilitar transições (console) ... 853
Habilitar ou desabilitar transições (CLI) ... 855
Monitorar pipelines ... 857
Monitoramento de CodePipeline eventos ... 858
Tipos de detalhes ... 860
Pipeline-level eventos ... 862
Stage-level eventos ... 870
Action-level eventos ... 875
Crie uma regra que envie uma notificação sobre um evento de pipeline ... 883
Referência do bucket de espaço reservado para eventos ... 887
Nomes de bucket de espaço reservado para eventos por região ... 888
Registro de chamadas de API do AWS CloudTrail com ... 892
CodePipeline informações em CloudTrail ... 892
Entendendo as entradas do arquivo de CodePipeline log ... 893
CodePipeline CloudWatch métricas ... 895
PipelineDuration ... 896
FailedPipelineExecutions ... 896
Solução de problemas ... 897
Erro de pipeline: um pipeline configurado com AWS Elastic Beanstalk retorna uma mensagem de erro: “Falha na implantação. A função fornecida não tem permissões suficientes: Service:AmazonElasticLoadBalancing” ... 898
Erro de implantação: um pipeline configurado com um AWS Elastic Beanstalk a ação de implantação trava em vez de falhar se a permissão "DescribeEvents" estiver ausente ... 899
Erro de pipeline: uma ação de origem retorna a mensagem de permissões insuficientes: “Não foi possível acessar o nome do CodeCommit `repositório`. Verifique se a função do IAM do pipeline tem permissões suficientes para acessar o repositório" ... 899
Erro de pipeline: uma ação de compilação ou de teste do Jenkins é executada por muito tempo e falha devido a falta de credenciais ou de permissões ... 900
Erro de pipeline: um pipeline criado em um AWS Região usando um bucket criado em outro AWS A região retorna um "InternalError" com o código "JobFailed” ... 900
Erro de implantação: um arquivo ZIP que contém um arquivo WAR é implantado com êxito no AWS Elastic Beanstalk, mas o URL do aplicativo relata um erro 404 não encontrado ... 899
Os nomes de pasta de artefatos do pipeline parecem estar truncados ... 901
Adicione CodeBuild GitClone permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com ... 902
Adicionar CodeBuild GitClone permissões para ações CodeCommit de origem ... 904

Erro de pipeline: uma implantação com a CodeDeployToECS ação retorna uma mensagem de erro: “Exceção ao tentar ler o arquivo de artefato de definição de tarefa de: nome do artefato de < origem” > 905
GitHub (via aplicativo OAuth) ação de origem: a lista de repositórios mostra diferentes repositórios 905
GitHub (via GitHub aplicativo) ação de origem: não é possível concluir a conexão com um repositório 906
Erro do Amazon S3: o < ARN da CodePipeline função de serviço > está negando o acesso ao S3 para o bucket do S3 < BucketName > 906
Pipelines com Amazon S3, Amazon ECR CodeCommit ou fonte não são mais iniciados automaticamente 909
Erro de conexão ao se conectar a GitHub: “Ocorreu um problema, verifique se os cookies estão habilitados em seu navegador” ou “O proprietário de uma organização deve instalar o GitHub aplicativo” 910
Pipelines com o modo de execução alterado para o modo QUEUED ou PARALLEL falham quando o limite de execução é atingido 911
Ao alterar para os modos QUEUED ou SUPERSEDED, pipelines configurados no modo PARALLEL podem apresentar uma definição desatualizada. 911
Os pipelines alterados do modo PARALLEL exibirão um modo de execução anterior 912
Pipelines configurados com conexões que filtram gatilhos por caminhos de arquivos podem falhar ao iniciar durante a criação da ramificação. 912
Pipelines com conexões que usam filtragem de gatilho por caminhos de arquivo podem não iniciar quando o limite de arquivos é atingido 913
CodeCommit ou as revisões de origem do S3 no modo PARALELO podem não corresponder ao evento EventBridge 913
A ação de implantação do EC2 falha com uma mensagem de erro `Não existe tal arquivo` 914
A ação EKS Deploy falha com uma mensagem de erro de `cluster inacessível` 915
O pipeline não é acionado para todas as ramificações quando várias ações de origem fazem referência ao mesmo repositório 916
Precisa de ajuda com outro problema? 916
Segurança 917
Proteção de dados 918
Privacidade do tráfego entre redes 919
Criptografia em repouso 919
Criptografia em trânsito 920

Gerenciamento das chaves de criptografia 920
Configure a criptografia do lado do servidor para artefatos armazenados no Amazon S3 para CodePipeline 920
Use AWS Secrets Manager para rastrear senhas de bancos de dados ou chaves de API de terceiros 923
Gerenciamento de identidade e acesso 924
Público 925
Autenticação com identidades 925
Gerenciar o acesso usando políticas 926
Como AWS CodePipeline funciona com o IAM 928
Identity-based exemplos de políticas 933
Resource-based exemplos de políticas 962
Solução de problemas 963
referência de permissões 965
Gerenciar a função CodePipeline de serviço 976
Resposta a incidentes 983
Validação de conformidade 983
Resiliência 984
Segurança da infraestrutura 984
Práticas recomendadas de segurança 985
Referência da estrutura do pipeline 986
Declaração de pipeline 990
name 992
roleArn 993
Loja de artefatos OU lojas de artefatos 993
estágios 994
version 995
executionMode 995
Tipo de tubulação 995
variables 996
triggers 996
metadata 1000
Declaração de estágio 1001
name 1004
actions 1004
condições 1004

regras ... 1005
Declaração de ação ... 1005
name ... 1009
region ... 1009
roleArn ... 1010
namespace ... 1010
ação TypeId ... 1010
InputArtifacts ... 1012
Artefatos de saída ... 1012
configuração (por provedor de ação) ... 1013
Ordem de execução ... 1015
Provedores de ação válidos em CodePipeline ... 1016
Configurações válidas para o parâmetro PollForSourceChanges ... 1021
Artefatos de entrada e saída válidos para cada tipo de ação ... 1023
Parâmetros de configuração válidos para cada tipo de provedor ... 1025
Referência da estrutura da ação ... 1029
Referência de ação do Amazon EC2 ... 1030
Tipo de ação ... 1031
Parâmetros de configuração ... 1031
Input artifacts (Artefatos de entrada) ... 1034
Artefatos de saída ... 1035
Permissões de política do perfil de serviço para a ação de implantação do EC2 ... 1035
Implantar referência do arquivo de especificação ... 1038
Declaração de ação ... 1039
Declaração de ação com exemplo de especificação Implantar ... 1041
Consulte também ... 1042
Referência da ação de origem do Amazon ECR ... 1042
Tipo de ação ... 1043
Parâmetros de configuração ... 1043
Input artifacts (Artefatos de entrada) ... 1044
Artefatos de saída ... 1044
Variáveis de saída ... 1044
Permissões do perfil de serviço: ação do Amazon ECR ... 1045
Declaração de ação (exemplo do Amazon ECR) ... 1045
Consulte também ... 1047
Referência da ação de compilação ECRBuildAndPublish ... 1047

Tipo de ação 1048
Parâmetros de configuração 1048
Input artifacts (Artefatos de entrada) 1049
Artefatos de saída 1049
Variáveis de saída 1049
Permissões do perfil de serviço: ação ECRBuildAndPublish 1050
Declaração de ação 1052
Consulte também 1053
Referência de ação de implantação do Amazon ECS e CodeDeploy azul esverdeado 1053
Tipo de ação 1054
Parâmetros de configuração 1054
Input artifacts (Artefatos de entrada) 1056
Artefatos de saída 1057
Permissões do perfil de serviço: ação CodeDeployToECS 1057
Declaração de ação 1059
Consulte também 1061
Referência da ação de implantação do Amazon Elastic Container Service 1062
Tipo de ação 1063
Parâmetros de configuração 1063
Input artifacts (Artefatos de entrada) 1064
Artefatos de saída 1064
Permissões do perfil de serviço: ação padrão do Amazon ECS 1064
Declaração de ação 1066
Consulte também 1067
Referência da ação de implantação EKS do Amazon Elastic Container Service 1068
Tipo de ação 1069
Parâmetros de configuração 1069
Input artifacts (Artefatos de entrada) 1070
Artefatos de saída 1071
Variáveis de ambiente 1071
Variáveis de saída 1071
Permissões de política de perfil de serviço 1071
Declaração de ação 1074
Consulte também 1075
Referência da ação de implantação do AWS Lambda 1075
Tipo de ação 1076

Parâmetros de configuração 1076
Input artifacts (Artefatos de entrada) 1080
Artefatos de saída 1080
Variáveis de saída 1080
Permissões de política do perfil de serviço para a ação de implantação do Lambda 1080
Declaração de ação 1082
Consulte também 1083
Referência da ação de implantação do Amazon S3 1083
Tipo de ação 1084
Parâmetros de configuração 1084
Input artifacts (Artefatos de entrada) 1086
Artefatos de saída 1086
Permissões do perfil de serviço: ação de implantação do S3 1086
Exemplo de configuração da ação 1087
Consulte também 1090
Referência da ação de origem do Amazon S3 1090
Tipo de ação 1091
Parâmetros de configuração 1091
Input artifacts (Artefatos de entrada) 1093
Artefatos de saída 1093
Variáveis de saída 1094
Permissões do perfil de serviço: ação de origem do S3 1094
Declaração de ação 1095
Consulte também 1096
Referência da ação de implantação do AWS AppConfig 1097
Tipo de ação 1097
Parâmetros de configuração 1097
Input artifacts (Artefatos de entrada) 1098
Artefatos de saída 1098
Permissões do perfil de serviço: ação `AppConfig` 1098
Exemplo de configuração da ação 1099
Consulte também 1100
Referência da ação de implantação do CloudFormation 1100
Tipo de ação 1101
Parâmetros de configuração 1101
Input artifacts (Artefatos de entrada) 1106

Artefatos de saída 1107
Variáveis de saída 1107
Permissões da função de serviço: CloudFormation ação 1107
Declaração de ação 1109
Consulte também 1110
CloudFormation StackSets 1111
Como CloudFormation StackSets ações funcionam 1112
Como estruturar StackSets ações em um pipeline 1114
A ação `CloudFormationStackSet` 1115
A CloudFormationStackInstances ação 1129
Permissões do perfil de serviço: ação `CloudFormationStackSet` 1140
Permissões do perfil de serviço: ação `CloudFormationStackInstances` 1140
Modelos de permissões para operações de conjuntos de pilhas 1140
Tipos de dados de parâmetro do modelo 1141
Consulte também 1110
AWS CodeBuild referência de ação de construção e teste 1143
Tipo de ação 1144
Parâmetros de configuração 1144
Input artifacts (Artefatos de entrada) 1147
Artefatos de saída 1148
Variáveis de saída 1149
Permissões da função de serviço: CodeBuild ação 1149
Declaração de ação (CodeBuild exemplo) 1150
Consulte também 1151
AWS CodePipeline invocar referência de ação 1152
Tipo de ação 1152
Parâmetros de configuração 1153
Input artifacts (Artefatos de entrada) 1155
Artefatos de saída 1156
Permissões da política de função de serviço para a CodePipeline ação de invocação 1156
Declaração de ação 1156
Consulte também 1157
AWS CodeCommit referência de ação de origem 1157
Tipo de ação 1158
Parâmetros de configuração 1159
Input artifacts (Artefatos de entrada) 1160

Artefatos de saída 1160
Variáveis de saída 1161
Permissões da função de serviço: CodeCommit ação 1162
Exemplo de configuração da ação 1162
Consulte também 1165
Referência da ação de implantação do AWS CodeDeploy 1165
Tipo de ação 1166
Parâmetros de configuração 1166
Input artifacts (Artefatos de entrada) 1166
Artefatos de saída 1166
Permissões da função de serviço: AWS CodeDeploy ação 1167
Declaração de ação 1168
Consulte também 1169
CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com 1170
Tipo de ação 1173
Parâmetros de configuração 1174
Input artifacts (Artefatos de entrada) 1175
Artefatos de saída 1175
Variáveis de saída 1176
Permissões da função de serviço: AWS CodeConnections ação 1176
Declaração de ação 1177
Instalação do aplicativo de instalação e criação de uma conexão 1179
Consulte também 1179
Referência da ação Commands 1180
Considerações para a ação Commands 1181
Permissões de política de perfil de serviço 1182
Tipo de ação 1183
Parâmetros de configuração 1183
Input artifacts (Artefatos de entrada) 1188
Artefatos de saída 1188
Variáveis de ambiente 1188
Permissões do perfil de serviço: ação Comandos 1190
Declaração de ação (exemplo) 1190
Consulte também 1192
AWS Device Farm referência de ação de teste 1192

Tipo de ação ... 1192
Parâmetros de configuração ... 1193
Input artifacts (Artefatos de entrada) ... 1197
Artefatos de saída ... 1197
Permissões da função de serviço: AWS Device Farm ação ... 1197
Declaração de ação ... 1198
Consulte também ... 1199
Referência da ação de implantação do Elastic Beanstalk ... 1200
Tipo de ação ... 1200
Parâmetros de configuração ... 1200
Input artifacts (Artefatos de entrada) ... 1201
Artefatos de saída ... 1201
Permissões do perfil de serviço: ação de implantação do ElasticBeanstalk ... 1201
Declaração de ação ... 1202
Consulte também ... 1203
Referência da ação invocar InspectorScan do Amazon Inspector ... 1203
ID do tipo de ação ... 1204
Parâmetros de configuração ... 1205
Input artifacts (Artefatos de entrada) ... 1207
Artefatos de saída ... 1207
Variáveis de saída ... 1207
Permissões do perfil de serviço: ação InspectorScan ... 1207
Declaração de ação ... 1208
Consulte também ... 1209
AWS Lambda invocar referência de ação ... 1210
Tipo de ação ... 1210
Parâmetros de configuração ... 1210
Input artifacts (Artefatos de entrada) ... 1211
Artefatos de saída ... 1211
Variáveis de saída ... 1211
Exemplo de configuração da ação ... 1211
Exemplo de evento JSON ... 1212
Consulte também ... 1215
Referência da ação de implantação do AWS OpsWorks ... 1215
Tipo de ação ... 1215
Parâmetros de configuração ... 1216

Input artifacts (Artefatos de entrada) 1216
Artefatos de saída 1216
Permissões da função de serviço: AWS OpsWorks ação 1216
Exemplo de configuração da ação 1217
Consulte também 1218
Referência da ação de implantação do AWS Service Catalog 1218
Tipo de ação 1218
Parâmetros de configuração 1218
Input artifacts (Artefatos de entrada) 1219
Artefatos de saída 1219
Permissões do perfil de serviço: ação do Service Catalog 1219
Configurações de ação de exemplo por tipo do arquivo de configuração 1220
Consulte também 1221
AWS Step Functions 1222
Tipo de ação 1215
Parâmetros de configuração 1222
Input artifacts (Artefatos de entrada) 1224
Artefatos de saída 1224
Variáveis de saída 1224
Permissões do perfil de serviço: ação `StepFunctions` 1224
Exemplo de configuração da ação 1225
Comportamento 1228
Consulte também 1100
Referência da estrutura de regras 1231
CloudWatchAlarm 1231
Tipo de regra 1232
Parâmetros de configuração 1232
Exemplo de configuração de regra 1232
Consulte também 1233
Regra do CodeBuild 1234
Permissões de política de perfil de serviço 1234
Tipo de regra 1235
Parâmetros de configuração 1235
Exemplo de configuração de regra 1237
Consulte também 1238
Comandos 1238

Considerações sobre a regra Comands ... 1239
Permissões de política de perfil de serviço ... 1234
Tipo de regra ... 1235
Parâmetros de configuração ... 1235
Exemplo de configuração de regra ... 1237
Consulte também ... 1238
DeploymentWindow ... 1243
Tipo de regra ... 1244
Parâmetros de configuração ... 1244
Exemplo de configuração de regra ... 1246
Consulte também ... 1247
LambdaInvoke ... 1247
Tipo de regra ... 1232
Parâmetros de configuração ... 1248
Exemplo de configuração de regra ... 1248
Consulte também ... 1249
VariableCheck ... 1249
Tipo de regra ... 1250
Parâmetros de configuração ... 1250
Exemplo de configuração de regra ... 1253
Consulte também ... 1254
Referência do modelo de integração ... 1255
Como os tipos de ação de terceiros funcionam com o integrador ... 1255
Conceitos ... 1256
Modelos de integração compatíveis ... 1258
Modelo de integração do Lambda ... 1259
Atualize sua função do Lambda para lidar com a entrada de CodePipeline ... 1260
Retorne os resultados da sua função Lambda para CodePipeline ... 1264
Use tokens de continuação para aguardar os resultados de um processo assíncrono ... 1266
Forneça CodePipeline as permissões para invocar a função Lambda do integrador em tempo de execução ... 1267
Modelo de integração do operador de trabalho ... 1267
Escolher e configurar uma estratégia de gerenciamento de permissões para o operador de trabalho ... 1268
Referência de arquivo de definições de imagem ... 1270
Arquivo imagedefinitions.json para ações de implantação padrão do Amazon ECS ... 1270

Detail.json arquivo de imagem para ações de blue/green implantação do Amazon ECS 1273
Referência de variáveis 1278
Conceitos 1279
Variáveis 1279
Namespaces 1280
Casos de uso de variáveis 1281
Configurar variáveis 1282
Configurar variáveis em nível de pipeline 1282
Configuração de variáveis no nível da ação 1283
Resolução de variáveis 1285
Regras para variáveis 1286
Variáveis disponíveis para ações de pipeline 1287
Ações com chaves variáveis definidas 1287
Ações com chaves variáveis configuradas pelo usuário 1293
Trabalhar com padrões glob na sintaxe 1296
Atualizar pipelines de sondagem para o método de detecção de alterações recomendado 1298
Atualizar uma ação de origem GitHub (via aplicativo OAuth) para uma ação de origem GitHub (via GitHub aplicativo) 1299
Etapa 1: Substitua sua ação (por meio do aplicativo OAuth) GitHub 1300
Etapa 2: criar uma conexão com GitHub 1301
Etapa 3: Salve sua ação GitHub de origem 1302
Cotas 1304
Apêndice A: GitHub (por meio do aplicativo OAuth) ações de origem 1320
Adicionar uma ação de GitHub origem (por meio do aplicativo OAuth) 1321
GitHub (via aplicativo OAuth) referência de ação de origem 1322
Tipo de ação 1323
Parâmetros de configuração 1323
Input artifacts (Artefatos de entrada) 1325
Artefatos de saída 1325
Variáveis de saída 1326
Declaração de ação (GitHub exemplo) 1326
Conectando-se a GitHub (OAuth) 1328
Consulte também 1328
Histórico do documento 1329
Atualizações anteriores 1364
Referência de recursos do CodePipeline 1376

..... mccclxxx

O que é AWS CodePipeline?

AWS CodePipeline é um serviço de entrega contínua que você pode usar para modelar, visualizar e automatizar as etapas necessárias para lançar seu software. Você pode modelar e configurar rapidamente os diferentes estágios de um processo de lançamento de software. CodePipeline automatiza as etapas necessárias para liberar suas alterações de software continuamente. Para obter informações sobre preços para CodePipeline, consulte Preços.

Tópicos

- Integração e entrega contínuas
- O que eu posso fazer com CodePipeline?
- Uma rápida olhada em CodePipeline
- Como faço para começar CodePipeline?
- CodePipeline conceitos
- DevOps exemplo de pipeline
- Como funcionam as execuções de pipeline
- Artefatos de entrada e saída
- Como funcionam as condições do estágio?
- Tipos de pipeline
- Que tipo de pipeline é ideal para mim?

Integração e entrega contínuas

CodePipeline é um serviço de entrega contínua que automatiza a criação, o teste e a implantação do seu software na produção.

A entrega contínua é uma metodologia de desenvolvimento de software em que o processo de lançamento é automatizado. Cada alteração de software é automaticamente criada, testada e implantada na produção. Antes do envio final para a produção, uma pessoa, um teste automatizado ou uma regra de negócios decide quando o envio final deve ocorrer. Embora toda alteração bem-sucedida de software possa ser imediatamente liberada para produção com a entrega contínua, nem todas as alterações precisam ser liberadas imediatamente.

A integração contínua é uma prática de desenvolvimento de software na qual os membros de uma equipe usam um sistema de controle de versão e frequentemente integram o trabalho com o mesmo

local, como uma ramificação principal. Toda alteração é criada e verificada para detectar erros de integração o mais rápido possível. O foco da integração contínua é construir e testar o código automaticamente, em comparação com a entrega contínua, que automatiza o inteiro processo de lançamento de software até a produção.

Para obter mais informações, consulte Praticando integração contínua e entrega contínua em AWS: Acelerando a entrega de software com. DevOps

Você pode usar o CodePipeline console, o AWS Command Line Interface (AWS CLI), os AWS SDKs ou qualquer combinação deles para criar e gerenciar seus pipelines.

O que eu posso fazer com CodePipeline?

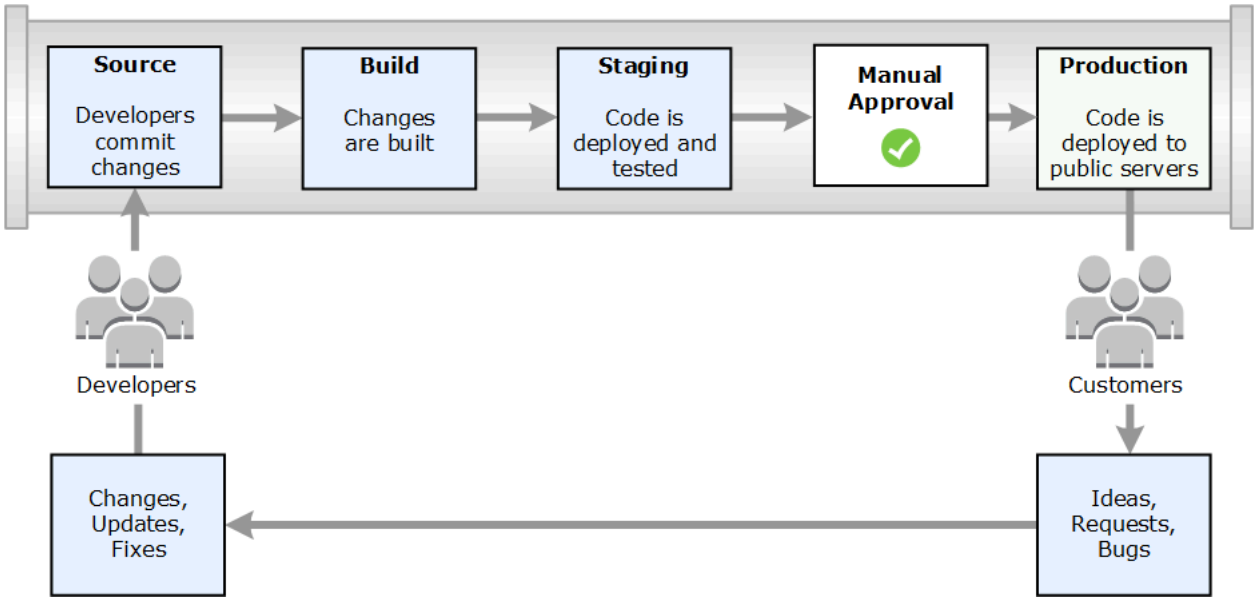
Você pode usar CodePipeline para ajudá-lo a criar, testar e implantar automaticamente seus aplicativos na nuvem. Especificamente, é possível:

- Automatize seus processos de lançamento: automatiza CodePipeline totalmente seu processo de lançamento de ponta a ponta, começando pelo repositório de origem por meio da criação, teste e implantação. É possível impedir que as alterações se desloquem em um pipeline ao incluir uma ação de aprovação manual em qualquer estágio, exceto no estágio Origem. Você poderá lançar quando desejar, da maneira que desejar, nos sistemas que escolher, em uma instância ou em várias instâncias.
- Estabeleça um processo de lançamento consistente: defina um conjunto consistente de etapas para cada alteração de código. CodePipeline executa cada estágio do seu lançamento de acordo com seus critérios.
- Acelerar a entrega e melhorar a qualidade: é possível automatizar o processo de lançamento para permitir que os desenvolvedores testem e lancem o código progressivamente e acelerar o lançamento de novos recursos para seus clientes.
- Usar as ferramentas favoritas: é possível incluir as ferramentas de origem, criação e implantação existentes no pipeline. Para obter uma lista completa das Serviços da AWS ferramentas de terceiros atualmente suportadas pelo CodePipeline, consulteIntegrações de produtos e serviços com CodePipeline.
- Consultar o progresso rapidamente: é possível consultar o status dos pipelines em tempo real, verificar os detalhes de todos os alertas, repetir as ações com falhas, ver detalhes das revisões de origem usadas na execução mais recente do pipeline em cada estágio e executar novamente e de maneira manual.

- Visualizar detalhes do histórico do pipeline: Você pode visualizar detalhes sobre execuções de um pipeline, incluindo horários de início e de término, duração da execução e IDs de execução.

Uma rápida olhada em CodePipeline

O diagrama a seguir mostra um exemplo de processo de lançamento usando o CodePipeline.



Nesse exemplo, quando os desenvolvedores confirmam alterações em um repositório de origem, o CodePipeline detecta automaticamente as alterações. Essas alterações são criadas e, se houver testes configurados, eles serão executados. Após a conclusão dos testes, o código criado será implantado nos servidores de preparação para a realização de testes. No servidor de teste, CodePipeline executa mais testes, como testes de integração ou de carga. Após a conclusão bem-sucedida desses testes e após a aprovação de uma ação de aprovação manual que foi adicionada ao pipeline, CodePipeline implanta o código testado e aprovado nas instâncias de produção.

CodePipeline pode implantar aplicativos em instâncias do EC2 usando CodeDeploy, AWS Elastic Beanstalk, ou AWS OpsWorks Stacks. CodePipeline também pode implantar aplicativos baseados em contêineres em serviços usando o Amazon ECS. Os desenvolvedores também podem usar os pontos de integração fornecidos CodePipeline para conectar outras ferramentas ou serviços, incluindo serviços de compilação, provedores de teste ou outros alvos ou sistemas de implantação.

Um pipeline pode ser tão simples ou tão complexo quanto à necessidade do processo de lançamento.

Como faço para começar CodePipeline?

Para começar com CodePipeline:

1. Saiba como CodePipeline funciona lendo a CodePipeline conceitos seção.
2. Prepare-se para usar CodePipeline seguindo as etapas emComeçando com CodePipeline.
3. Experimente CodePipeline seguindo as etapas nos CodePipeline tutoriais tutoriais.
4. Use CodePipeline para seus projetos novos ou existentes seguindo as etapas emCriar um pipeline, estágios e ações.

CodePipeline conceitos

Modelar e configurar seu processo de lançamento automatizado é mais fácil se você entender os conceitos e termos usados em AWS CodePipeline. Veja alguns conceitos que você deve saber ao usar o CodePipeline.

Para ver um exemplo de DevOps pipeline, consulteDevOps exemplo de pipeline.

Os seguintes termos são usados em CodePipeline:

Tópicos

- Pipelines
- Execuções de pipeline
- Operadores de estágio
- Execuções de ação
- Tipos de execução
- Tipos de ação
- Artefatos
- Revisões de origem
- Acionadores
- Variáveis
- Condições
- Regras

Pipelines

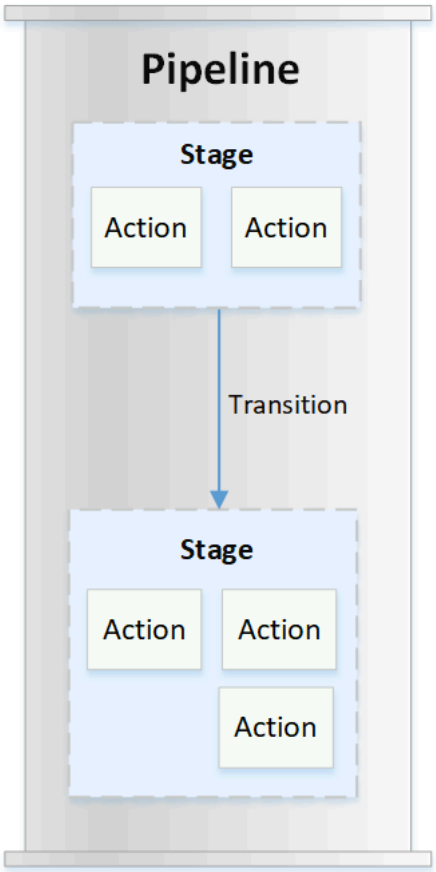
Um pipeline é uma estrutura de fluxo de trabalho que descreve como as alterações de software passam pelo processo de lançamento. Cada pipeline é composto por uma série de estágios.

Estágios

Um estágio é uma unidade lógica que você pode usar para isolar um ambiente e limitar o número de alterações simultâneas nesse ambiente. Cada estágio contém ações que são executadas nos artefatosdo aplicativo. Seu código-fonte é um exemplo de artefato. Um estágio pode ser um estágio de compilação, em que o código-fonte é compilado e os testes são executados. Também pode ser um estágio de implantação, em que o código é implantado em ambientes de tempo de execução. Cada estágio é composto por uma série de ações seriais ou paralelas.

Transições

Uma transição é o ponto em que uma execução de pipeline passa para o próximo estágio no pipeline. Você pode desativar a transição de entrada de um estágio para impedir que execuções entrem nesse estágio e habilitar a transição para permitir que as execuções continuem. Quando mais de uma execução chega a uma transição desabilitada, somente a execução mais recente continua com o próximo estágio quando a transição está habilitada. Isso significa que execuções mais recentes continuam substituindo execuções em espera enquanto a transição está desabilitada e, após a transição ser habilitada, a execução que continua é a execução de substituição.



Ações

Uma ação é um conjunto de operações executadas no código do aplicativo e configuradas para que as ações sejam executadas no pipeline em um ponto especificado. Isso pode incluir itens, como uma ação de origem de uma alteração no código, uma ação para implantar o aplicativo em instâncias e assim por diante. Por exemplo, um estágio de implantação pode conter uma ação de implantação que implanta código em um serviço de computação como o Amazon EC2 ou AWS Lambda.

Os tipos de CodePipeline ação válidos são `source` `build` `testdeploy`,`approval`,, `invoke` e. Para obter uma lista de provedores de ação, consulte Provedores de ação válidos em CodePipeline.

As ações podem ser executadas em série ou em paralelo. Para obter informações sobre ações seriais e paralelas em um estágio, consulte as informações `runOrder` nos requisitos de estrutura de ação.

Execuções de pipeline

Uma execução é um conjunto de alterações liberadas por um pipeline. Cada execução de pipeline é única e tem seu próprio ID. Uma execução corresponde a um conjunto de alterações, como uma

confirmação mesclada ou uma liberação manual da última confirmação. Duas execuções podem liberar o mesmo conjunto de alterações em momentos diferentes.

Embora um pipeline possa processar várias execuções ao mesmo tempo, um estágio do pipeline processa apenas uma execução por vez. Para fazer isso, um estágio é bloqueado enquanto processa uma execução. Duas execuções de pipeline não podem ocupar o mesmo estágio ao mesmo tempo. A execução à espera de entrar no estágio ocupado é uma execução de entrada. Uma execução de entrada ainda pode falhar, ser substituída ou ser interrompida manualmente. Para obter mais informações sobre como as execuções de entrada funcionam, consulte Como funcionam as execuções de entrada.

As execuções de pipeline atravessam estágios de pipeline em ordem. Os status válidos para pipelines são `InProgress`, `Stopping`, `Stopped`, `Succeeded`, `Superseded` e `Failed`.

Para obter mais informações, consulte PipelineExecution.

Execuções interrompidas

A execução do pipeline pode ser interrompida manualmente para que a execução do pipeline em andamento não continue pelo pipeline. Se for interrompida manualmente, uma execução de pipeline mostrará um status `Stopping` até que ela seja completamente interrompida. Depois, ela mostrará um status `Stopped`. Uma execução de pipeline `Stopped` pode ser repetida.

Há duas maneiras de interromper a execução de um pipeline:

- Interromper e aguardar
- Interromper e abandonar

Para obter informações sobre casos de uso para interromper uma execução e detalhes de sequência para essas opções, consulte Como as execuções de pipeline são interrompidas.

Execuções com falha

Se uma execução falhar, ela será parada e não atravessará o pipeline completamente. Seu status muda para `FAILED` e CodePipeline desbloqueia o palco. Uma execução mais recente pode recuperar o atraso e entrar no estágio desbloqueado e bloqueá-lo. Você pode tentar uma execução com falha novamente, a menos que a execução com falha tenha sido substituída ou não possa ser repetida. Você pode reverter um estágio com falha para uma execução anterior bem-sucedida.

Modos de execução

Para fornecer o conjunto mais recente de alterações por meio de um pipeline, execuções mais recentes passam e substituem execuções menos recentes que já estão sendo executadas por meio do pipeline. Quando isso ocorre, a execução mais antiga é substituída pela execução mais recente. Uma execução pode ser substituída por uma execução mais recente em um determinado ponto, que é o ponto entre estágios. SUPERSEDED é o modo de execução padrão.

No modo SUPERSEDED, se uma execução estiver aguardando para entrar em um estágio bloqueado, uma execução mais recente poderá alcançá-la e substituí-la. A execução mais recente agora aguarda o desbloqueio do estágio, e a execução substituída é interrompida com um status `SUPERSEDED`. Quando uma execução de pipeline é substituída, a execução é interrompida e não atravessa completamente o pipeline. Não é mais possível tentar a execução substituída novamente depois que ela tiver sido substituída nesse estágio. Outros modos de execução disponíveis são os modos PARALLEL ou QUEUED.

Para ter mais informações sobre os modos de execução e estágios bloqueados, consulte Como as execuções são processadas no modo SUPERSEDED.

Operadores de estágio

Quando uma execução de pipeline passa por um estágio, o estágio está no processo de conclusão de todas as ações dentro dele. Para ter informações sobre como funcionam as operações de estágio e informações sobre os estágios bloqueados, consulte Como as execuções são processadas no modo SUPERSEDED.

Os status válidos dos estágios são `InProgress`, `Stopping`, `Stopped`, `Succeeded` e `Failed`. Você pode repetir um estágio com falha, a menos que o estágio com falha não possa ser repetido. Para obter mais informações, consulte StageExecution. É possível reverter um estágio para uma execução anterior bem-sucedida definida. Um estágio pode ser configurado para reverter automaticamente em caso de falha, conforme detalhado em Configuração da reversão de estágio. Para obter mais informações, consulte RollbackStage.

Execuções de ação

Uma execução de ação é o processo de conclusão de uma ação configurada que opera em artefatos designados. Esses podem ser artefatos de entrada, artefatos de saída ou ambos. Por exemplo, uma ação de compilação pode executar comandos de compilação em um artefato de entrada, como compilar o código-fonte do aplicativo. Os detalhes da execução da ação incluem um ID de execução

da ação, o gatilho de origem de execução do pipeline relacionado e os artefatos de entrada e saída da ação.

Os status válidos das ações são `InProgress`, `Abandoned`, `Succeeded` e `Failed`. Para obter mais informações, consulte ActionExecution.

Tipos de execução

Uma execução de pipeline ou estágio pode ser uma execução padrão ou revertida.

Para tipos padrão, a execução tem um ID exclusivo e é uma execução de pipeline completa. Uma reversão de pipeline tem um estágio a ser revertido e uma execução bem-sucedida para o estágio como a execução de destino para a qual será revertido. A execução do pipeline de destino é usada para recuperar revisões e variáveis de origem para o estágio a ser executado novamente.

Tipos de ação

Os tipos de ação são ações pré-configuradas que estão disponíveis para seleção em CodePipeline. O tipo de ação é definido por seu proprietário, provedor, versão e categorial O tipo de ação fornece parâmetros personalizados que são usados para concluir as tarefas de ação em um pipeline.

Para obter informações sobre produtos Serviços da AWS e serviços de terceiros que você pode integrar ao seu funil com base no tipo de ação, consulteIntegrações com tipos de CodePipeline ação.

Para obter informações sobre os modelos de integração compatíveis com os tipos de ação em CodePipeline, consulteReferência do modelo de integração.

Para obter informações sobre como provedores terceirizados podem configurar e gerenciar tipos de ação em CodePipeline, consulteComo trabalhar com tipos de ação.

Artefatos

Os artefatos se referem à coleta de dados, como código-fonte de aplicativo, aplicativos criados, dependências, arquivos de definições, modelos e assim por diante, que são trabalhados por ações de pipeline. Os artefatos são produzidos por algumas ações e consumidos por outras. Em um pipeline, os artefatos podem ser o conjunto de arquivos trabalhados por uma ação (artefatos de entrada) ou a saída atualizada de uma ação concluída (artefatos de saída).

As ações passam a saída para outra ação para processamento adicional usando o bucket de artefatos do pipeline. CodePipeline copia artefatos para a loja de artefatos, onde a ação os coleta. Para obter mais informações sobre artefatos, consulte Artefatos de entrada e saída.

Revisões de origem

Quando você faz uma alteração no código-fonte, uma versão é criada. Uma revisão de origem é a versão de uma alteração de origem que aciona uma execução de pipeline. Uma execução processa as revisões de origem. Para repositórios GitHub e CodeCommit repositórios, esse é o commit. Para buckets ou ações do S3, essa é a versão do objeto.

É possível iniciar a execução de um pipeline com uma revisão de origem, como uma confirmação, especificada por você. A execução processará a revisão especificada e substituirá o que teria sido a revisão usada para a execução. Para obter mais informações, consulte Iniciar um pipeline com uma substituição da revisão de origem.

Acionadores

Os gatilhos são eventos que acionam seu pipeline. Alguns gatilhos, como o acionamento de um pipeline manualmente, estão disponíveis para todos os provedores de ação de origem em um pipeline. Determinados gatilhos dependem do provedor de origem de um pipeline. Por exemplo, CloudWatch os eventos devem ser configurados com recursos de eventos da Amazon CloudWatch que tenham o ARN do pipeline adicionado como destino na regra do evento. O Amazon CloudWatch Events é o gatilho recomendado para a detecção automática de alterações em pipelines com uma ação de origem CodeCommit ou S3. Os webhooks são um tipo de gatilho configurado para eventos de repositórios de terceiros. Por exemplo, o WebhookV2 é um tipo de gatilho que permite que as tags do Git sejam usadas para iniciar pipelines com provedores de origem terceirizados GitHub.com, como GitHub Enterprise Server GitLab.com, GitLab autogerenciado ou Bitbucket Cloud. Na configuração do pipeline, você pode especificar um filtro para gatilhos, como solicitação push ou pull. Você pode filtrar eventos de envio de código em etiquetas, ramificações ou caminhos de arquivo Git. Você pode filtrar eventos de pull request por eventos (abertos, atualizados, fechados), ramificações ou caminhos de arquivo.

Para ter mais informações sobre gatilhos, consulte Inicie um funil em CodePipeline. Para assistir a um tutorial que explica como usar as tags Git como gatilhos no pipeline, consulte Tutorial: Usar tags do Git para iniciar o pipeline.

> **Important**
>
> Os pipelines inativos por mais de 30 dias terão a sondagem desabilitada para o pipeline. Para obter mais informações, consulte sondagem DisabledAt na referência da estrutura

do pipeline. Consulte as etapas de migração do pipeline da sondagem para a detecção de alterações baseada em eventos em Métodos de detecção da alteração.

Variáveis

Uma variável é um valor que pode ser usado para configurar ações dinamicamente em seu pipeline. As variáveis podem ser declaradas no nível do pipeline ou emitidas por ações no pipeline. Os valores de variáveis são resolvidos no momento da execução do pipeline e podem ser visualizados no histórico da execução. Para variáveis declaradas no nível do pipeline, você pode definir valores padrão na configuração do pipeline ou substituí-los para uma determinada execução. Para variáveis emitidas por uma ação, o valor fica disponível após a conclusão bem-sucedida de uma ação. Para obter mais informações, consulte Referência de variáveis.

Condições

Uma condição contém um conjunto de regras que são avaliadas. Se todas as regras em uma condição forem bem-sucedidas, a condição será atendida. Você pode configurar condições para que, quando os critérios não forem atendidos, o resultado especificado, como falha no estágio, seja ativado. As condições também são chamadas de portas porque permitem especificar quando uma execução entrará e passará por um estágio ou sairá do estágio após passar por ele. Isso é análogo a permitir que uma fila de tráfego em uma rodovia se reúna em um portão fechado e especificar a abertura do portão para permitir o fluxo de tráfego para dentro de uma área. Os resultados para os tipos de condição incluem falha no estágio ou reversão do estágio. As condições ajudam você a especificar quando essas ações acontecem em um estágio do pipeline. Você pode substituir as condições no runtime.

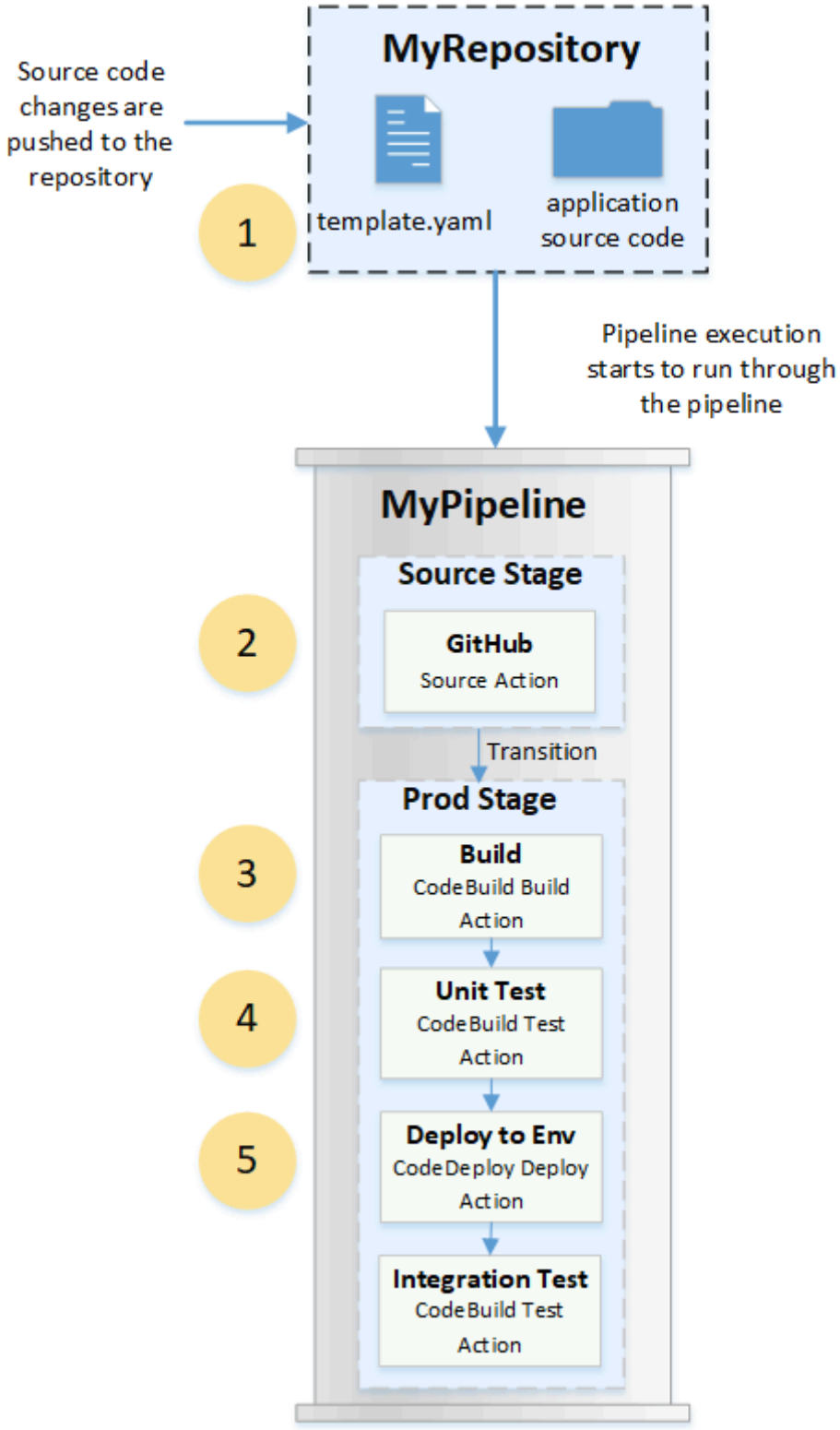
Existem três tipos de condições. As condições de entrada respondem à pergunta “Se as regras da condição forem atendidas, entre no estágio”. O estágio é bloqueado quando a execução entra nele, depois as regras são executadas. Para condições “em caso de falha”, a regra é acionada quando um estágio falha, resultando na reversão de um estágio com falha. Para condições “em caso de sucesso”, a regra é ativada quando um estágio é bem-sucedido, como verificar se há alarmes em uma execução bem-sucedida antes de prosseguir. Por exemplo, uma condição On Success resultaria na reversão de um estágio bem-sucedido se a CloudWatchAlarm regra descobrisse que há alarmes no ambiente de implantação. Para obter mais informações, consulte Como funcionam as condições do estágio?.

Regras

As condições usam uma ou mais regras pré-configuradas que executam e realizam verificações, ativando o resultado configurado quando a condição não for atendida. Por exemplo, atender a todas as regras para uma regra de condição de entrada que verifica o status do alarme e os tempos de janela de implantação implantará um estágio bem-sucedido após todas as verificações serem aprovadas. Para obter mais informações, consulte Como funcionam as condições do estágio?.

DevOps exemplo de pipeline

Como exemplo de DevOps pipeline, um pipeline de dois estágios pode ter um estágio de origem chamado Source e um segundo estágio chamado Prod. Neste exemplo, o pipeline está atualizando o aplicativo com as alterações mais recentes e implantando continuamente o resultado mais recente. Antes de implantar o aplicativo mais recente, o pipeline compila e testa o aplicativo web. Neste exemplo, um grupo de desenvolvedores configurou um modelo de infraestrutura e o código-fonte de um aplicativo web em um GitHub repositório chamado MyRepository.



Por exemplo, um desenvolvedor envia uma correção para a página de índice do aplicativo web e ocorre o seguinte:

1. O código-fonte do aplicativo é mantido em um repositório configurado como uma ação de GitHub origem no pipeline. Quando os desenvolvedores enviam os commits para o repositório, CodePipeline detecta a alteração enviada e a execução do pipeline começa no estágio de origem.
2. A ação de GitHub origem é concluída com êxito (ou seja, as alterações mais recentes foram baixadas e armazenadas no bucket de artefatos exclusivo dessa execução). Os artefatos de saída produzidos pela ação de GitHub origem, que são os arquivos do aplicativo do repositório, são então usados como artefatos de entrada a serem trabalhados pelas ações na próxima etapa.
3. A execução do pipeline faz a transição do Estágio de origem para o Estágio de produção. A primeira ação no Prod Stage executa um projeto de construção criado CodeBuild e configurado como uma ação de construção no pipeline. A tarefa de compilação extrai uma imagem do ambiente de compilação e compila o aplicativo web em um contêiner virtual.
4. A próxima ação no Prod Stage é um projeto de teste de unidade criado CodeBuild e configurado como uma ação de teste no pipeline.
5. O código testado da unidade depois é trabalhado por uma ação de implantação no Estágio de produção que implanta o aplicativo em um ambiente de produção. Depois que a ação de implantação for concluída com êxito, a ação final na etapa é um projeto de teste de integração criado CodeBuild e configurado como uma ação de teste no pipeline. A ação de teste chama scripts de shell que instalam e executam uma ferramenta de teste, como um verificador de links, no aplicativo web. Após a conclusão bem-sucedida, a saída é um aplicativo web compilado e um conjunto de resultados de teste.

Os desenvolvedores podem adicionar ações ao pipeline que implantam ou fazem testes adicionais no aplicativo depois que ele é compilado e testado para cada alteração.

Para obter mais informações, consulte Como funcionam as execuções de pipeline.

Como funcionam as execuções de pipeline

Esta seção fornece uma visão geral da forma como CodePipeline processa um conjunto de mudanças. CodePipelinerastreia cada execução de pipeline que começa quando um pipeline é iniciado manualmente ou quando uma alteração é feita no código-fonte. CodePipeline usa os seguintes modos de execução para lidar com a forma como cada execução progride no pipeline. Para obter mais informações, consulte Definir ou alterar o modo de execução do pipeline.

- Modo SUPERSEDED: uma execução mais recente pode ultrapassar uma mais antiga. Esse é o padrão.

- Modo QUEUED: as execuções são processadas uma por uma na ordem em que são enfileiradas. Isso requer o tipo de pipeline V2.
- Modo PARALLEL: no modo PARALLEL, as execuções ocorrem simultaneamente e independentemente umas das outras. As execuções não esperam que outras operações sejam concluídas antes de começar ou terminar. Isso requer o tipo de pipeline V2.

> **Important**
>
> Para pipelines em modo PARALLEL, a reversão do estágio não está disponível. Da mesma forma, condições de falha com um tipo de resultado de reversão não podem ser adicionadas a um pipeline de modo PARALLEL.

Como as execuções de pipeline são iniciadas

Você pode iniciar uma execução ao alterar seu código-fonte ou iniciar o pipeline manualmente. Você também pode acionar uma execução por meio de uma regra do Amazon CloudWatch Events que você agenda. Por exemplo, quando uma alteração no código-fonte é enviada para um repositório configurado como a ação de origem do pipeline, o pipeline detecta a alteração e inicia uma execução.

> **Note**
>
> Se um pipeline tiver várias ações de origem, todas elas serão executadas novamente, mesmo que uma alteração seja detectada para apenas uma ação de origem.

Como as revisões de origem são processadas em execuções de pipeline

Para cada execução do pipeline que começa com alterações no código-fonte (revisões da origem), as revisões da origem são determinadas da seguinte forma.

- Para pipelines com uma CodeCommit fonte, o HEAD é clonado CodePipeline no momento em que o commit é enviado. Por exemplo, uma confirmação é enviada, o que inicia o pipeline para a execução 1. No momento em que uma segunda confirmação é enviada, inicia-se o pipeline para a execução 2.

> **Note**
>
> Para pipelines no modo PARALLEL com uma CodeCommit origem, independentemente da confirmação que acionou a execução do pipeline, a ação de origem sempre clonará o HEAD no momento em que for iniciada. Para obter mais informações, consulte CodeCommit ou as revisões de origem do S3 no modo PARALELO podem não corresponder ao evento EventBridge.

- Para pipelines com uma origem do S3, o EventBridge evento para a atualização do bucket do S3 é usado. Por exemplo, o evento é gerado quando um arquivo é atualizado no bucket de origem, que inicia o pipeline para execução 1. No momento em que o evento para uma segunda atualização do bucket é feito, inicia-se o pipeline para a execução 2.

  > **Note**
  >
  > Para pipelines no modo PARALLEL com uma origem do S3, independentemente da etiqueta de imagem que acionou a execução, a ação de origem sempre começará com a etiqueta de imagem mais recente. Para obter mais informações, consulte CodeCommit ou as revisões de origem do S3 no modo PARALELO podem não corresponder ao evento EventBridge.

- Para pipelines com uma fonte de conexões, como o Bitbucket, o HEAD é clonado CodePipeline no momento em que o commit é enviado. Por exemplo, em um pipeline no modo PARALLEL, o primeiro commit inicia a execução 1, enquanto o segundo será utilizado para a segunda execução do pipeline.

Como as substituições de fonte funcionam com o transformador de entrada EventBridge

É possível usar substituições para iniciar um pipeline com um ID de revisão de origem específico fornecido para a execução do pipeline. Por exemplo, se você quiser iniciar um pipeline que processará um ID de confirmação específico de sua CodeCommit origem, você pode adicionar o ID de confirmação como uma substituição ao iniciar seu funil.

Há quatro tipos de revisão de origem para `revisionType`:

- `COMMIT_ID`

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

> **Note**
>
> Para os tipos de revisões de origem `COMMIT_ID` e `IMAGE_DIGEST`, o ID de revisão de origem se aplica a todo o conteúdo no repositório, em todas as ramificações.

> **Note**
>
> Para os `S3_OBJECT_KEY` tipos `S3_OBJECT_VERSION_ID` e de revisões de fonte, qualquer um dos tipos pode ser usado de forma independente ou pode ser usado em conjunto para substituir a fonte por um ID de versão específico ObjectKey . Para `S3_OBJECT_KEY`, o parâmetro de configuração `AllowOverrideForS3ObjectKey` precisa ser definido como `true`. Para ter mais informações sobre os parâmetros de configuração de origem do S3, consulte Parâmetros de configuração.

Você pode especificar substituições de origem usando o transformador de entrada in. EventBridge Use o transformador de entrada para passar os dados como uma das seguintes opções:

- Você pode usar o transformador de entrada para passar os dados como parâmetros JSON.
- Você pode usar o transformador de entrada para passar variáveis de pipeline.

Para ver exemplos de como passar os dados como parâmetros JSON, consulteAções e recursos de origem do Amazon ECR EventBridge, Conectando-se às ações de origem do Amazon S3 que usam EventBridge e AWS CloudTrail para S3 e CodeCommit ações de origem e EventBridge para. CodeCommit

Como as execuções de pipeline são interrompidas

Para usar o console a fim de interromper a execução de um pipeline, você pode selecionar Stop execution (Interromper execução) na página de visualização do pipeline, na página do histórico da execução ou na página do histórico detalhado. Para usar a CLI a fim de interromper a execução de

um pipeline, use o comando `stop-pipeline-execution`. Para obter mais informações, consulte Interrompa a execução de um pipeline em CodePipeline.

Há duas maneiras de interromper a execução de um pipeline:

- Interromper e aguardar: todas as execuções de ação em andamento podem ser concluídas e as ações subsequentes não são iniciadas. A execução do pipeline não continua para estágios subsequentes. Não é possível usar essa opção em uma execução que já está em um estado `Stopping`.
- Interromper e abandonar: todas as execuções de ação em andamento são abandonadas e não são concluídas, e as ações subsequentes não são iniciadas. A execução do pipeline não continua para estágios subsequentes. É possível usar essa opção em uma execução que já está em um estado `Stopping`.

> **Note**
>
> Essa opção pode levar a tarefas com falha ou a tarefas fora de sequência.

Cada opção resulta em uma sequência diferente de fases de execução de pipeline e ação, como se segue.

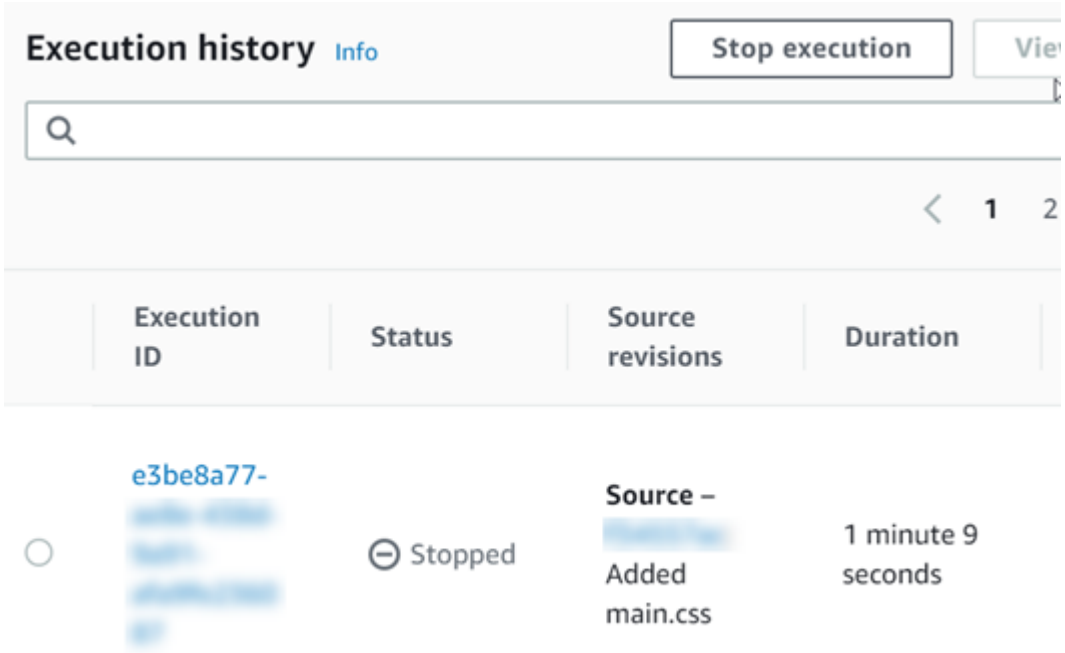
Opção 1: interromper e aguardar

Quando você optar por interromper e aguardar, a execução selecionada continuará até que as ações em andamento sejam concluídas. Por exemplo, a execução do pipeline a seguir foi interrompida enquanto a ação de compilação estava em andamento.

1. Na visualização do pipeline, o banner da mensagem de êxito é exibido e a ação de compilação continua até que seja concluída. O status de execução do pipeline é Stopping (Interrompendo).

   Na visualização do histórico, o status das ações em andamento, como a ação de compilação, é In progress (Em andamento) até que a ação de compilação seja concluída. Enquanto as ações estão em andamento, o status de execução do pipeline é Stopping (Interrompendo).
2. A execução é interrompida quando o processo de interrupção é concluído. Se a ação de compilação for concluída com êxito, seu status será Succeeded (Bem-sucedido), e a execução do pipeline mostrará um status Stopped (Interrompido). As ações subsequentes não são iniciadas. O botão Retry (Repetir) está habilitado.

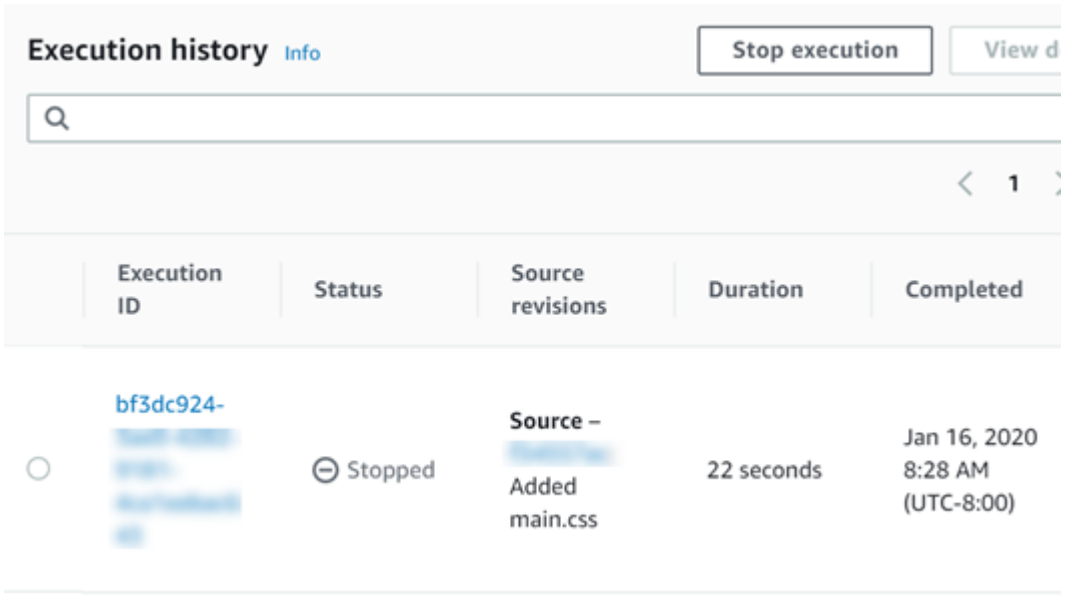
Na visualização do histórico, o status da execução é Stopped (Interrompido) após a conclusão da ação em andamento.



Opção 2: interromper e abandonar

Quando você escolhe interromper e abandonar, a execução selecionada não aguarda a conclusão das ações em andamento. As ações são abandonadas. Por exemplo, a execução do pipeline a seguir foi interrompida e abandonada enquanto a ação de compilação estava em andamento.

1. Na visualização de pipeline, a mensagem de banner de sucesso é exibida, a ação de compilação mostra um status de In progress (Em andamento) e a execução do pipeline mostra um status Stopping (Interrompendo).
2. Depois que a execução do pipeline for interrompida, a ação de compilação mostrará um status Abandoned (Abandonado), e a execução do pipeline mostra um status de Stopped (Interrompido). As ações subsequentes não são iniciadas. O botão Retry (Repetir) está habilitado.
3. Na visualização do histórico, o status da execução é Stopped (Interrompido).



Casos de uso para interromper a execução de um pipeline

Recomendamos que você use a opção de interromper e aguardar para interromper a execução de um pipeline. Essa opção é mais segura porque evita possíveis tarefas com falha ou fora de sequência no pipeline. Quando uma ação é abandonada em CodePipeline, o provedor da ação continua com todas as tarefas relacionadas à ação. No caso de uma CloudFormation ação, a ação de implantação no pipeline é abandonada, mas a atualização da pilha pode continuar e resultar em uma falha na atualização.

Como exemplo de ações abandonadas que podem resultar em tarefas fora de sequência, se você estiver implantando um arquivo grande (1 GB) por meio de uma ação de implantação do S3 e optar por interromper e abandonar a ação enquanto a implantação já estiver em andamento, a ação será abandonada no Amazon S3 CodePipeline, mas continuará no Amazon S3. O Amazon S3 não encontra nenhuma instrução para cancelar o upload. Depois, se você iniciar uma nova execução de pipeline com um arquivo muito pequeno, agora haverá duas implantações em andamento. Como o tamanho do arquivo da nova execução é pequeno, a nova implantação é concluída enquanto ainda está sendo feito upload da implantação antiga. Quando a implantação antiga for concluída, o novo arquivo será substituído pelo arquivo antigo.

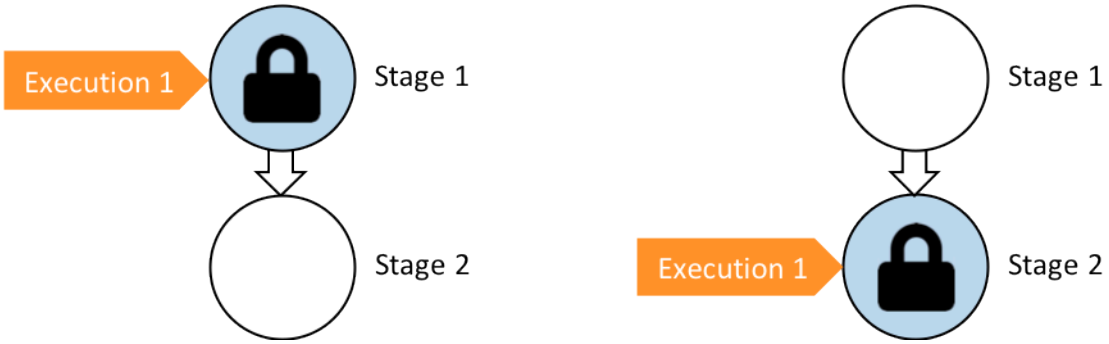
Você pode querer usar a opção interromper e abandonar no caso em que você tem uma ação personalizada. Por exemplo, é possível abandonar uma ação personalizada com um trabalho que não precise terminar antes de iniciar uma nova execução para uma correção de erro.

Como as execuções são processadas no modo SUPERSEDED

O modo padrão para processamento de execuções é o modo SUPERSEDED. Uma execução consiste em um conjunto de alterações capturadas e processadas pela execução. Os pipelines podem processar várias execuções ao mesmo tempo. Cada execução é executada por meio do pipeline separadamente. O pipeline processa cada execução na ordem e pode substituir uma execução anterior por uma posterior. As regras a seguir são usadas para processar execuções em um pipeline para o modo SUPERSEDED.

Regra 1: os estágios são bloqueados quando uma execução está sendo processada

Como cada estágio pode processar apenas uma execução por vez, o estágio é bloqueado enquanto está em andamento. Quando a execução conclui um estágio, uma transição é feita para o próximo estágio no pipeline.

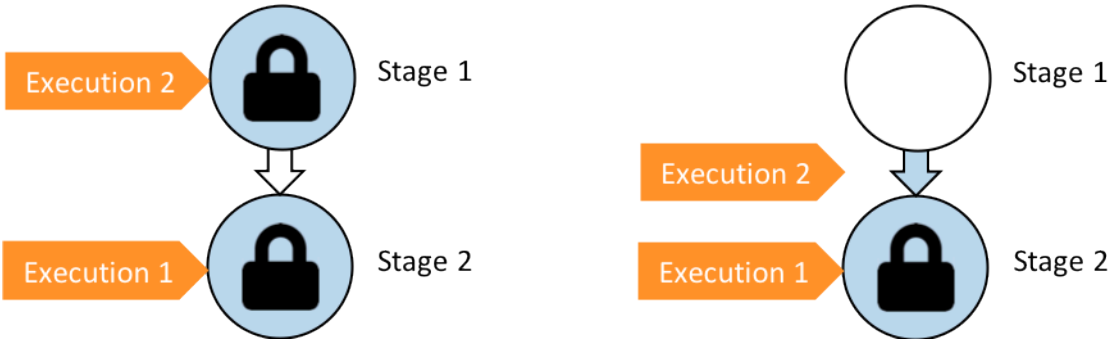


Antes: Stage 1 is locked as Execution 1 enters. Depois: Stage 2 is locked as Execution 1 enters.

Regra 2: as execuções subsequentes aguardam o desbloqueio do estágio

Enquanto um estágio está bloqueado, as execuções em espera são mantidas na frente do estágio bloqueado. Antes de um estágio ser considerado com concluído, todas as ações configuradas para um estágio devem ser concluídas com êxito. Uma falha libera o bloqueio no estágio. Quando uma execução é interrompida, a execução não continua em um estágio e o estágio é desbloqueado.

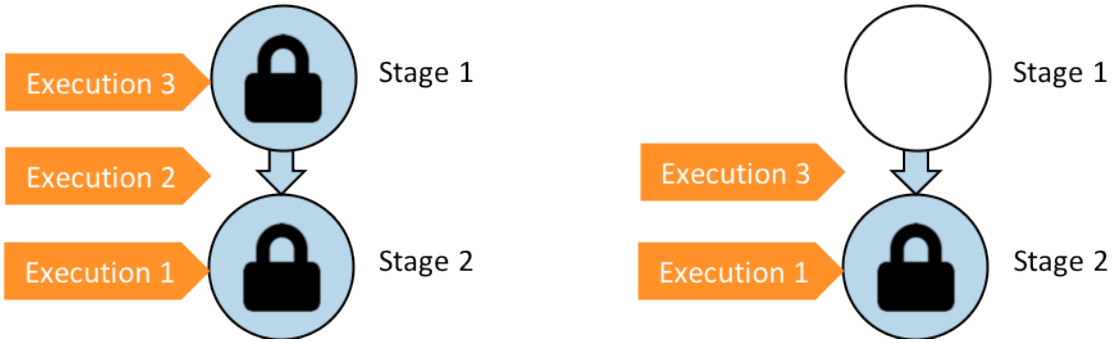
> **Note**
>
> Antes de interromper uma execução, recomendamos que você desative a transição na frente do estágio. Dessa forma, quando o estágio é desbloqueado devido à execução interrompida, o estágio não aceita uma execução de pipeline subsequente.



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

Regra 3: as execuções em espera são substituídas por execuções mais recentes

As execuções só são substituídas entre estágios. Um estágio bloqueado mantém uma execução na frente do estágio em espera pela conclusão do estágio. Uma execução mais recente ultrapassa uma execução em espera e continua para o próximo estágio assim que o estágio é desbloqueado. A execução substituída não continua. Neste exemplo, a Execução 2 foi substituída pela Execução 3 ao aguardar o estágio bloqueado. A Execução 3 entra no estágio seguinte.



Antes: a execução 2 espera entre os estágios enquanto a execução 3 entra no estágio 1. Após: a execução 3 sai do estágio 1. A execução 2 é substituída pela execução 3.

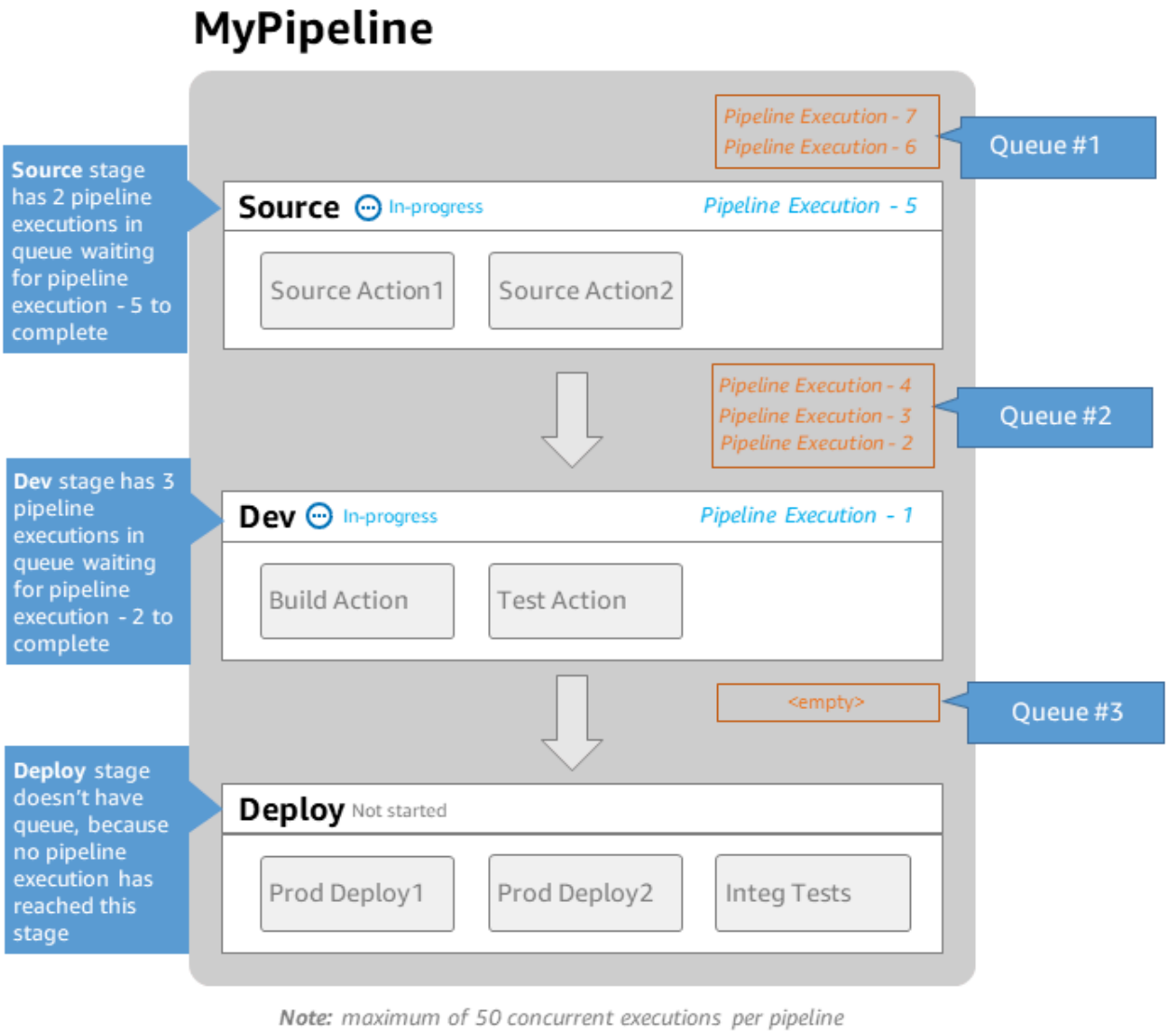
Para ter mais informações sobre considerações para visualizar e alternar entre modos de execução, consulte Definir ou alterar o modo de execução do pipeline. Para ter mais informações sobre cotas com modos de execução, consulte Cotas em AWS CodePipeline.

Como as execuções são processadas no modo QUEUED

Para pipelines no modo QUEUED, os estágios são bloqueados quando uma execução está sendo processada; no entanto, as execuções em espera não ultrapassam as que já foram iniciadas.

As execuções em espera se reúnem nos pontos de entrada dos estágios bloqueados na ordem em que chegam ao estágio, formando uma fila de execuções em espera. Com o modo QUEUED, você pode ter várias filas no mesmo pipeline. Quando uma execução em fila entra em um estágio, o estágio é bloqueado e nenhuma outra execução pode entrar. Esse comportamento permanece o mesmo do modo SUPERSEDED. Quando a execução termina o estágio, o estágio fica desbloqueado e pronto para a próxima execução.

O diagrama a seguir mostra como os estágios de um pipeline no modo QUEUED processam as execuções. Por exemplo, enquanto o estágio de Origem processa a execução 5, as execuções de 6 e 7 formam a fila 1 e aguardam no ponto de entrada do estágio. A próxima execução na fila será processada após o desbloqueio do estágio.



Para ter mais informações sobre considerações para visualizar e alternar entre modos de execução, consulte Definir ou alterar o modo de execução do pipeline. Para ter mais informações sobre cotas com modos de execução, consulte Cotas em AWS CodePipeline.

Como as execuções são processadas no modo PARALLEL

Para pipelines no modo PARALLEL, as execuções são independentes umas das outras e não esperam que outras execuções sejam concluídas antes de começar. Não há filas. Para visualizar as execuções paralelas no pipeline, use a visualização do histórico de execução.

Use o modo PARALLEL em ambientes de desenvolvimento em que cada recurso tem sua própria ramificação de recursos e é implantado em destinos que não são compartilhados por outros usuários.

Para ter mais informações sobre considerações para visualizar e alternar entre modos de execução, consulte Definir ou alterar o modo de execução do pipeline. Para ter mais informações sobre cotas com modos de execução, consulte Cotas em AWS CodePipeline.

Gerenciar o fluxo do pipeline

O fluxo de execuções do pipeline pode ser controlado por:

- Uma transição que controla o fluxo de execuções no estágio. As transições podem ser habilitadas ou desabilitadas. Quando uma transição é desabilitada, as execuções do pipeline não podem entrar no estágio. A execução do pipeline que espera para entrar em um estágio em que a transição está desativada é chamada de execução de entrada. Depois que você habilitar a transição, uma execução de entrada passará para o estágio e o bloqueará.

  De forma semelhante às execuções que aguardam um estágio bloqueado, quando uma transição é desabilitada, a execução que está aguardando para entrar no estágio ainda pode ser substituída por uma nova execução. Quando uma transição desabilitada é habilitada novamente, a execução mais recente, incluindo as que substituíram execuções mais antigas enquanto a transição estava desabilitada, entra no estágio.
- Uma ação de aprovação que impede que um pipeline faça a transição para a próxima ação até que uma permissão seja concedida (por exemplo, por meio de uma aprovação manual de uma identidade autorizada). Você pode usar uma ação de aprovação quando quiser controlar o momento em que um pipeline faz a transição para um estágio final de Production (Produção), por exemplo.

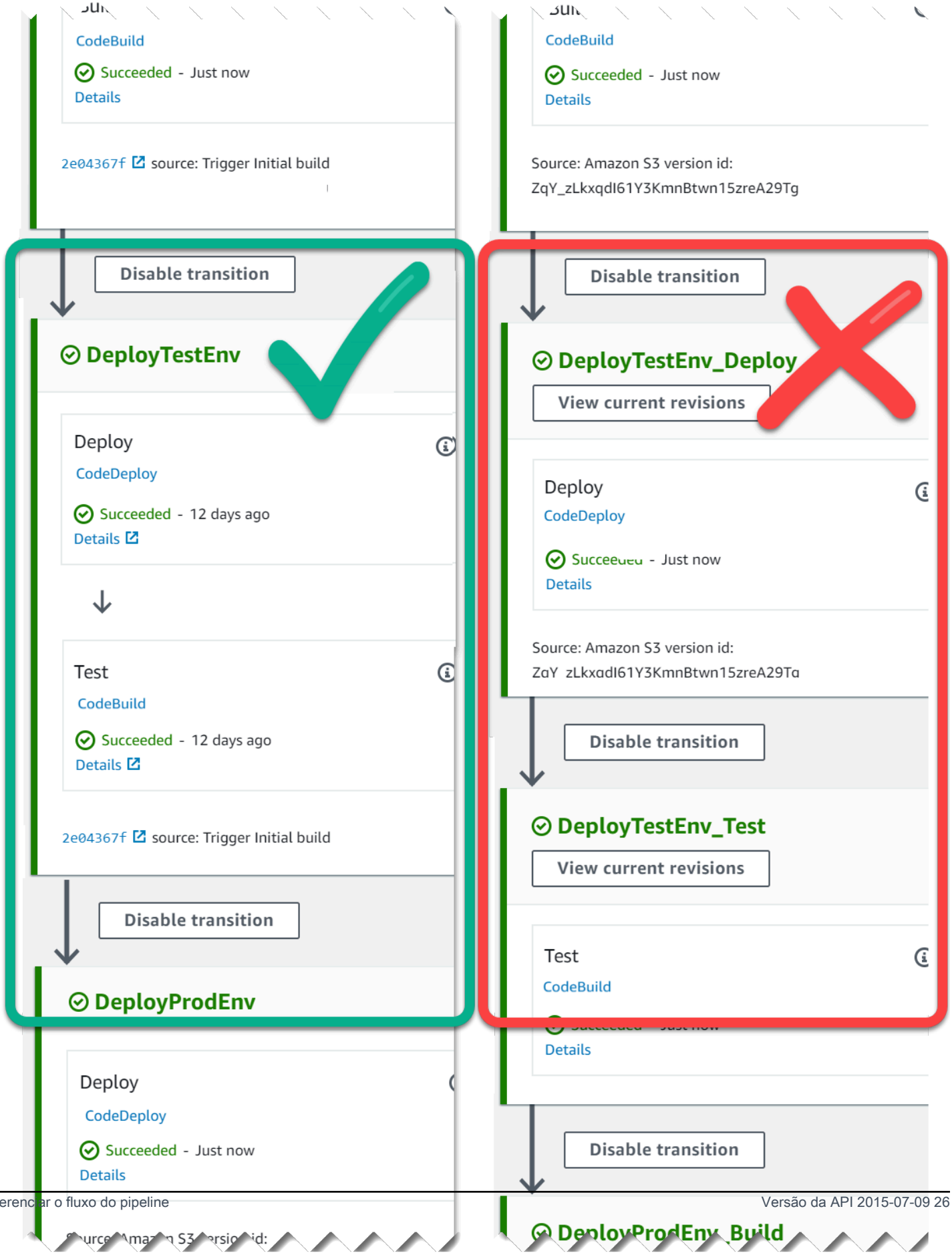
> **Note**
>
> Um estágio com uma ação de aprovação é bloqueado até que a ação de aprovação seja aprovada ou rejeitada ou tenha expirado. Uma ação de aprovação expirada é processada da mesma maneira que uma ação com falha.

- Uma falha, quando uma ação em um estágio não é concluída com êxito. A revisão não faz a transição para a próxima ação no estágio ou para o próximo estágio no pipeline. O seguinte pode ocorrer:
  - Você retome manualmente o estágio que contém as ações com falha. Isso retoma a execução (ele repete as ações que falharam e, se forem bem-sucedidas, continua no stage/pipeline).
  - Outra execução entra no estágio com falha e substitui a execução com falha. Nesse ponto, a execução com falha não pode ser repetida.

Estrutura recomendada do pipeline

Ao decidir como uma alteração no código deve fluir pelo pipeline, é melhor agrupar as ações relacionadas dentro de um estágio para que, quando o estágio estiver bloqueado, todas as ações processem a mesma execução. Você pode criar um estágio para cada ambiente de aplicativo ou zona de disponibilidade e assim por diante. Região da AWS Um pipeline com muitos estágios (ou seja, muito granular) pode permitir muitas alterações simultâneas, enquanto um pipeline com muitas ações em um estágio grande (muito grande) podem demorar muito para liberar uma alteração.

Como um exemplo, uma ação de teste após uma ação de implantação no mesmo estágio certamente testará a mesma alteração que foi implantada. Neste exemplo, uma alteração é implantada em um ambiente de teste, depois, testada, e, então, a alteração mais recente no ambiente de teste é implantada em um ambiente de produção. No exemplo recomendado, o ambiente de teste e o ambiente de produção são estágios separados.


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

À esquerda: ações relacionadas de teste, implantação e aprovação agrupadas (recomendado). À direita: ações relacionadas em estágios separados (não recomendado).

Como funcionam as execuções de entrada

Execução de entrada é uma execução que aguarda a disponibilização de um estágio, uma transição ou uma ação indisponível antes de avançar. O próximo estágio, transição ou ação pode não estar disponível porque:

- Outra execução já entrou no próximo estágio e a bloqueou.
- A transição para entrar no próximo estágio está desabilitada.

Será possível desabilitar uma transição para manter uma execução de entrada se quiser determinar se uma execução atual tem tempo para ser concluída nos estágios subsequentes ou interromper todas as ações em um ponto específico. Para determinar se você tem uma execução de entrada, é possível visualizar o pipeline no console ou ver a saída do comando get-pipeline-state.

As execuções de entrada funcionam com as seguintes considerações:

- Assim que a ação, a transição ou o estágio bloqueado se tornam disponíveis, a execução de entrada em andamento entra no estágio e prossegue pelo pipeline.
- Enquanto a execução de entrada está aguardando, ela pode ser interrompida manualmente. Uma execução de entrada pode ter o status `InProgress`, `Stopped` ou `Failed`.
- Quando uma execução de entrada é interrompida ou falha, não é possível repeti-la porque não há nenhuma ação com falha para repetir. Quando uma execução de entrada é interrompida e a transição está habilitada, a execução de entrada interrompida não prossegue pelo estágio.

É possível visualizar ou interromper uma execução de entrada.

Artefatos de entrada e saída

CodePipeline se integra às ferramentas de desenvolvimento para verificar alterações no código e, em seguida, criar e implantar em todos os estágios do processo de entrega contínua. Artefatos são os arquivos que são trabalhados por ações no pipeline, como arquivos ou pastas com código de aplicação, arquivos de página de índice, scripts, etc. Por exemplo, o artefato da ação de origem do Amazon S3 é um nome de arquivo (ou caminho de arquivo) em que os arquivos de código-fonte da aplicação são fornecidos para a ação de origem do pipeline. Os arquivos são fornecidos como

um arquivo ZIP, como o seguinte nome do artefato de exemplo: `SampleApp_Windows.zip`. O artefato de saída para a ação de origem, os arquivos de código-fonte da aplicação, são o artefato de saída da ação e o artefato de entrada para a próxima ação, como uma ação de compilação. Como outro exemplo, uma ação de compilação pode executar comandos que compilem o código-fonte da aplicação para um artefato de entrada, que são os arquivos de código-fonte da aplicação. Consulte a página de referência de configuração da ação para uma ação específica para obter detalhes sobre os parâmetros do artefato, como AWS CodeBuild referência de ação de construção e teste para a CodeBuild ação.

As ações usam artefatos de entrada e saída armazenados no bucket de artefatos do Amazon S3 que você escolheu ao criar o pipeline. CodePipeline compacta e transfere os arquivos para artefatos de entrada ou saída conforme apropriado para o tipo de ação no palco.

> **Note**
>
> O bucket de artefatos não é o mesmo usado como local do arquivo de origem de um pipeline em que a ação de origem escolhida é o S3.

Por exemplo:

1. CodePipeline aciona a execução do pipeline quando há uma confirmação no repositório de origem, fornecendo o artefato de saída (quaisquer arquivos a serem criados) do estágio de origem.
2. O artefato de saída (todos os arquivos a serem compilados) do estágio anterior é processado como um artefato de entrada para o estágio de Compilação. Um artefato de saída (o aplicativo compilado) do estágio de Compilação pode ser um aplicativo atualizado ou uma imagem do Docker atualizada criada para um contêiner.
3. O artefato de saída do estágio anterior (o aplicativo de compilação) é processado como um artefato de entrada para o estágio de implantação, como ambientes de preparação ou de produção na Nuvem AWS. Você pode implantar aplicativos para uma frota de implantação ou pode implantar aplicativos baseados em contêiner para tarefas em execução nos clusters do ECS.

Ao criar ou editar uma ação, você designa o artefato ou os artefatos de entrada e de saída para a ação. Neste exemplo de um pipeline de dois estágios, com um estágio de Origem e de Implantação, em Editar ação, escolha o nome do artefato da ação de origem para o artefato de entrada da ação de implantação.

- Quando você usa o console para criar seu primeiro pipeline, CodePipeline cria um bucket do Amazon S3 no mesmo Conta da AWS e armazena itens Região da AWS para todos os pipelines. Toda vez que você usa o console para criar outro pipeline nessa região, CodePipeline cria uma pasta para esse pipeline no bucket. Ele usará essa pasta para armazenar os artefatos do pipeline conforme a execução do processo de lançamento automatizado. Esse bucket é chamado de codepipeline- *region* -*12345EXAMPLE-1a2b-3c4d-67890EXAMPLE*, onde *region* está a AWS região na qual você criou o pipeline e *12345EXAMPLE-1a2b-3c4d-67890EXAMPLE* é um valor aleatório baseado em um identificador universalmente exclusivo (UUID) que garante que o nome do bucket seja exclusivo.

  > **Note**
  >
  > Se você já tem um bucket começando com codepipeline- region - na região em que você está criando o pipeline, use-o CodePipeline como o bucket padrão. Também segue a ordem lexicográfica; por exemplo, codepipeline-region-abcexample é escolhido antes de codepipeline-region-defexample.

  CodePipeline trunca os nomes dos artefatos, o que pode fazer com que alguns nomes de buckets pareçam semelhantes. Mesmo que o nome do artefato pareça estar truncado, CodePipeline mapeia para o repositório de artefatos de uma forma que não seja afetada por artefatos com nomes truncados. O pipeline pode funcionar normalmente. Isso não é um problema com a pasta ou os artefatos. Há um limite de 100 caracteres para nomes de pipelines. Embora o nome da pasta do artefato possa parecer reduzido, ele ainda é exclusivo para o pipeline.

  Ao criar ou editar um pipeline, você deve ter um bucket de artefatos no pipeline Conta da AWS e Região da AWS, além disso, deve ter um bucket de artefatos por região em que planeja executar uma ação. Se você usar o console para criar um pipeline ou ações entre regiões, buckets de artefatos padrão são configurados pelo CodePipeline nas regiões em que se encontram as ações.

  Se você usar o AWS CLI para criar um pipeline, poderá armazenar os artefatos desse pipeline em qualquer bucket do Amazon S3, desde que esse bucket esteja no Região da AWS mesmo Conta da AWS e no pipeline. É possível fazer isso caso se preocupe em não exceder os limites dos buckets do Amazon S3 permitidos para a conta. Se você usar o AWS CLI para criar ou editar um funil e adicionar uma ação entre regiões (uma ação com um AWS provedor em uma região diferente do seu funil), deverá fornecer um bucket de artefatos para cada região adicional em que planeja executar uma ação.

- Cada ação tem um tipo. Dependendo do tipo, a ação poderá ter um destes itens ou os dois:
  - Um artefato de entrada, que é o que ele utiliza ou no qual trabalha durante o curso de execução da ação.
  - Um artefato de saída, que é o resultado da ação.

  Todos os artefatos de saída no pipeline devem ter um nome exclusivo. Cada artefato de entrada de uma ação deve corresponder ao artefato de saída de uma ação anterior no pipeline, não importa se essa ação for imediatamente anterior à ação em um estágio ou for executada em um estágio que ocorreu vários estágios antes.

  Mais de uma ação pode trabalhar em um artefato.

Como funcionam as condições do estágio?

Para cada condição que especifica uma regra, a regra é executada. Se a condição falhar, o estágio executará o resultado especificado. Opcionalmente, como parte da regra, você também especifica quais recursos CodePipeline devem ser usados em determinados casos. Por exemplo, a `CloudWatchAlarm` regra usará um recurso de CloudWatch alarme para executar verificações da condição.

Uma condição pode corresponder a várias regras, e cada regra pode especificar um dos três provedores.

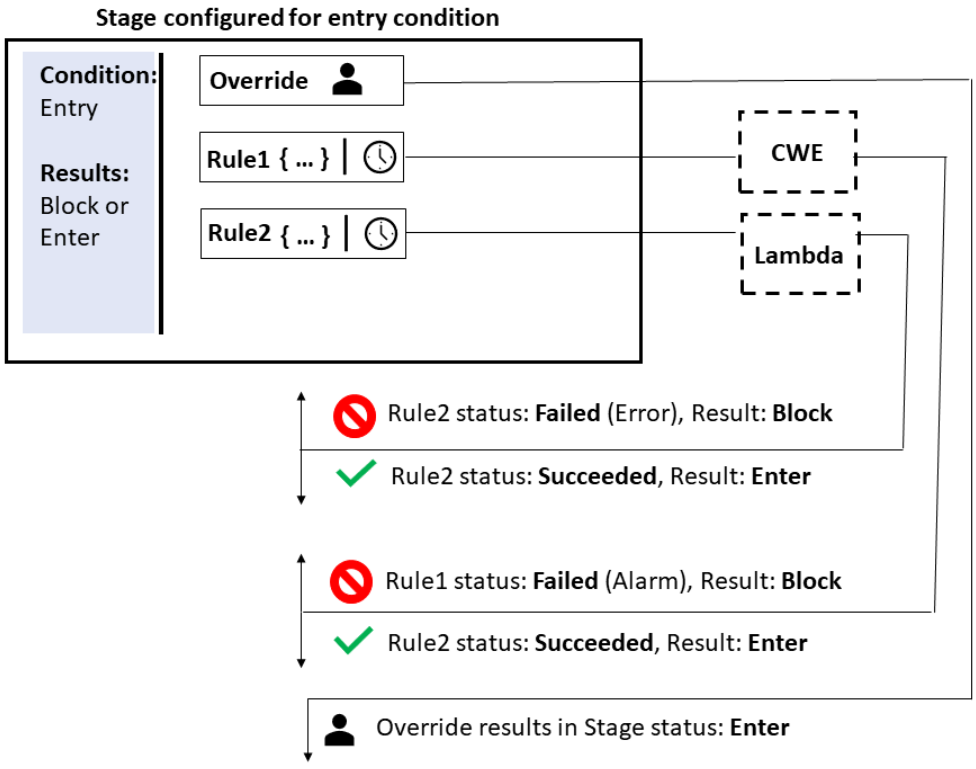
O fluxo de alto nível para criar condições é o seguinte:

1. Escolha o tipo de condição dentre os tipos de condição disponíveis em CodePipeline. Por exemplo, use um tipo de condição “em caso de sucesso” para configurar um estágio para que, após o sucesso do estágio, um conjunto de regras possa ser usado para executar verificações antes de prosseguir.
2. Escolha a regra . Por exemplo, a regra `CloudWatchAlarm` verificará os alarmes e usará o EB para verificar um limite de alarme pré-configurado. Se a verificação for bem-sucedida e o alarme estiver abaixo do limite, o estágio poderá prosseguir.
3. Configure o resultado, como uma reversão que seria usada se a regra falhasse.

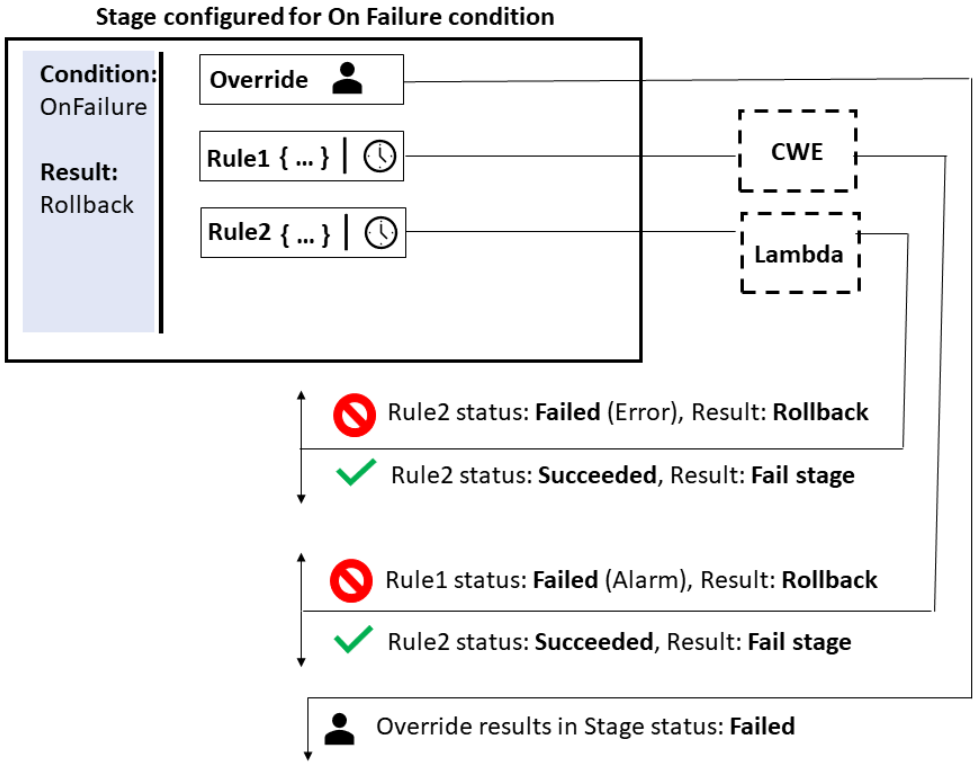
As condições são usadas para tipos específicos de expressões e cada uma tem opções específicas para resultados disponíveis da seguinte forma:

- Entrada: as condições para fazer verificações que, se atendidas, permitem a entrada em um estágio. As regras são usadas com as seguintes opções de resultado: Falhar ou Ignorar
- Em caso de falha: as condições para fazer verificações para o estágio em que ela falha. As regras estão envolvidas com a seguinte opção de resultado: Reversão
- Bem-sucedido: as condições para fazer verificações para o estágio em que for bem-sucedida. As regras são envolvidas com as seguintes opções de resultado: Reversão ou Falha

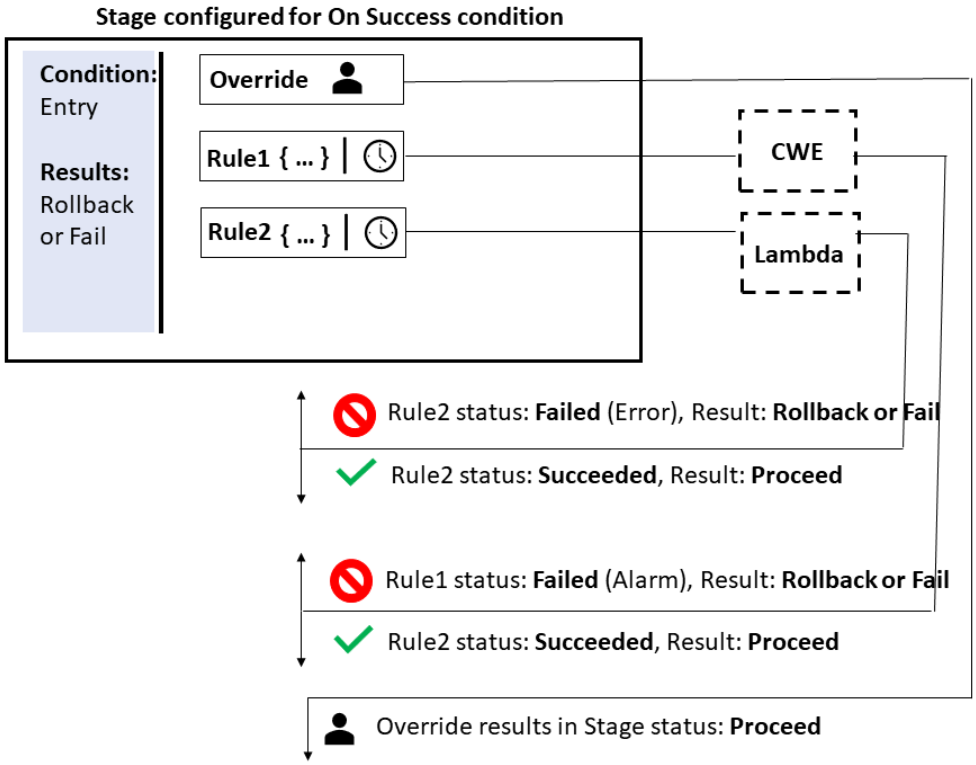
O diagrama a seguir mostra um exemplo de fluxo para o tipo de condição de entrada CodePipeline. As condições respondem à pergunta: O que deverá acontecer se a condição não for atendida, ou seja, se alguma regra falhar? No fluxo a seguir, uma condição de entrada é configurada com uma LambdaInvoke regra e uma `CloudWatchAlarm` regra. Se a regra falhar, o resultado configurado como Falha será acionado.



O diagrama a seguir mostra um exemplo de fluxo para a condição On Failure, digite in CodePipeline. As condições respondem à pergunta: O que deverá acontecer se a condição for atendida, ou seja, se todas as regras forem bem-sucedidas nas verificações? No fluxo a seguir, uma condição On Failure é configurada com uma LambdaInvoke regra e uma `CloudWatchAlarm` regra. Se a regra falhar, o resultado configurado como Falha será acionado.



O diagrama a seguir mostra um exemplo de fluxo para a condição On Success, digite in CodePipeline. As condições respondem à pergunta: O que deverá acontecer se a condição for atendida, ou seja, se todas as regras forem bem-sucedidas nas verificações? No fluxo a seguir, uma condição “em caso de sucesso” é configurada com uma regra `LambdaInvoke` e uma regra `CloudWatchAlarm`. Se a regra falhar, o resultado configurado como Falha será acionado.



Regras para condições de estágio

Ao configurar as condições do estágio, você seleciona entre as regras predefinidas e especifica os resultados da regra. Um estado de condição será Falha se alguma das regras da condição falhar, e Bem-sucedido se todas as regras forem bem-sucedidas. A forma como os critérios são atendidos para as condições Em caso de falha e Bem-sucedido depende do tipo de regra.

A seguir estão algumas regras gerenciadas que você pode adicionar às condições do estágio.

- As condições podem usar a regra Commands para especificar comandos a fim de atender aos critérios de regra para condições. Para ter mais informações sobre essa regra, consulte Comandos.
- As condições podem usar a AWS DeploymentWindow regra para especificar horários de implantação aprovados para permitir uma implantação. Os critérios da regra serão medidos com uma expressão cron fornecida para uma janela de implantação. A regra é bem-sucedida quando a data e a hora na janela de implantação atendem aos critérios na expressão cron da regra. Para ter mais informações sobre essa regra, consulte DeploymentWindow.
- As condições podem usar a regra AWS Lambda para verificar os estados de erro gerados de funções do Lambda configuradas. A regra é atendida quando a verificação recebe o resultado da

função do Lambda. Um erro da função do Lambda atende aos critérios das condições “em caso de falha”. Para ter mais informações sobre essa regra, consulte LambdaInvoke.

- As condições podem usar a AWS CloudWatchAlarm regra para verificar se há alarmes configurados a partir de CloudWatch eventos. A regra é cumprida quando a verificação retorna um estado de alarme de OK, ALARM ou INSUFFICIENT_DATA. Para condições “em caso de sucesso”, OK e INSUFFICIENT_DATA atendem aos critérios. ALARM atende aos critérios das condições “em caso de falha”. Para ter mais informações sobre essa regra, consulte CloudWatchAlarm.
- As condições podem usar a VariableCheck regra para criar uma condição em que a variável de saída seja verificada em relação a uma expressão fornecida. A regra passa na verificação quando o valor da variável atende aos critérios da regra, como o valor ser igual ou maior que uma variável de saída especificada. Para ter mais informações sobre essa regra, consulte VariableCheck.

Tipos de pipeline

CodePipeline fornece os seguintes tipos de pipeline, que diferem em características e preço, para que você possa adaptar os recursos e o custo do pipeline às necessidades de seus aplicativos.

- Os pipelines do tipo V1 têm uma estrutura JSON que contém parâmetros padrão no nível do pipeline, do estágio e da ação.
- Os pipelines do tipo V2 têm a mesma estrutura do tipo V1, junto com parâmetros adicionais para segurança de lançamento e configuração de gatilhos.

Para obter informações sobre preços para CodePipeline, consulte Preços.

Consulte a página CodePipeline referência de estrutura de tubulação para obter detalhes sobre os parâmetros em cada tipo de pipeline. Para obter informações sobre qual tipo de pipeline selecionar, consulte Que tipo de pipeline é ideal para mim?.

Que tipo de pipeline é ideal para mim?

O tipo de pipeline é determinado pelo conjunto de características e recursos compatíveis com cada versão do pipeline.

Veja um resumo dos casos de uso e características disponíveis para cada tipo de pipeline.

	Tipo V1	Tipo V2
Características		
Casos de uso	• Implantações padrão	• Implantações com configuração a partir da transmissão de variáveis em nível de pipeline em runtime • Implantações em que os pipelines são configurados para iniciar com tags do Git
Variáveis do Action-level	Compatível	Compatível
Modo de execução EM PARALLEL	Não compatível	Compatível
Pipeline-level variables	Não compatível	Compatível
Modo de execução QUEUED	Não compatível	Compatível
Reversão para estágios do pipeline	Não compatível	Compatível
Substituições da revisão de origem	Não compatível	Compatível
Condições de estágio	Não compatível	Compatível
Gatilhos e filtragem de etiquetas Git, solicitações pull, ramificações ou caminhos de arquivo	Não compatível	Compatível
A ação Commands	Não compatível	Compatível
Criar condições de entrada com resultado Ignorar	Não compatível	Compatível

	Tipo V1	Tipo V2
Características		
Configurar a repetição automática em caso de falha para um estágio	Não compatível	Compatível

Para obter informações sobre preços para CodePipeline, consulte Preços.

Você pode criar e executar um script Python para ajudar a analisar o custo potencial de mover um pipeline do tipo V1 para um pipeline do tipo V2.

> **Note**
>
> O exemplo de script abaixo é destinado apenas para fins de demonstração e avaliação. Não é uma ferramenta de cotação e não garante o custo do uso real de um pipeline do tipo V2, além de não incluir impostos que possam ser aplicados. Para obter informações sobre preços para CodePipeline, consulte Preços.

Como criar e executar um script para ajudar você a avaliar o custo de migrar um pipeline do tipo V1 para um pipeline do tipo V2

1. Baixe e instale o Python.
2. Abra uma janela do terminal. Execute o comando a seguir para criar um novo script python chamado PipelineCostAnalyzer.py.

```
vi PipelineCostAnalyzer.py
```

3. Copie e cole o código a seguir no 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. No terminal ou no prompt de comando, altere os diretórios para onde você criou o script do analisador.

   Nesse diretório, execute o comando a seguir, onde region é Região da AWS onde você criou os pipelines V1 que deseja analisar. Ou você também pode avaliar um pipeline específico fornecendo o nome:

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

   Por exemplo, execute o comando a seguir para executar o script python chamado PipelineCostAnalyzer.py. Neste exemplo, a região é `us-west-2`.

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

   > **Note**
   >
   > Esse script analisará todos os pipelines V1 na Região da AWS especificada, a menos que você especifique um nome de pipeline específico.

5. No exemplo de saída do script a seguir, podemos ver a lista de execuções de ações, a lista de execuções de ações que eram qualificadas para cobrança, o runtime total dessas execuções de ações e o custo estimado dessas ações conforme executadas em um pipeline 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
```

Neste exemplo, o valor negativo na última linha representa o valor estimado que pode ser economizado ao migrar para pipelines do tipo V2.

> **Note**
>
> A saída do script e os exemplos relacionados que mostram custos e outras informações são apenas estimativas. São apenas para fins de demonstração e avaliação e não garantem nenhuma economia real. Para obter informações sobre preços para CodePipeline, consulte Preços.

Começando com CodePipeline

Se você é CodePipeline novato, pode seguir os tutoriais deste guia depois de seguir as etapas deste capítulo para configurar.

O CodePipeline console inclui informações úteis em um painel dobrável que você pode abrir a partir do ícone de informações ou de qualquer link de informações na página.

().

Você pode fechar esse painel a qualquer momento.

O CodePipeline console também fornece uma maneira de pesquisar rapidamente seus recursos, como repositórios, criar projetos, aplicativos de implantação e pipelines. Selecione Go to resource (Acessar recurso) ou pressione a tecla / e digite o nome do recurso. Qualquer correspondência aparecerá na lista. As pesquisas não diferenciam letras maiúsculas de minúsculas. Só é possível ver recursos para os quais você tem permissão de visualizar. Para obter mais informações, consulte Visualizar recursos no console.

Antes de poder usar AWS CodePipeline pela primeira vez, você deve criar seu Conta da AWS e criar seu primeiro usuário administrativo.

Tópicos

- Inscreva-se para um Conta da AWS
- Aplique uma política gerenciada para acesso administrativo ao CodePipeline
- Instale o AWS CLI
- Abra o console para CodePipeline
- Próximas etapas

Inscreva-se para um Conta da AWS

Para começar AWS, você precisa de um Conta da AWS. Há duas maneiras de se inscrever em AWS. Se você quiser ter controle total sobre a configuração da conta e onde seu conteúdo é armazenado, ou se tiver cargas de trabalho regulamentadas, como HIPAA ou FedRAMP, use Inscrever-se em (avançado). AWS Se você for um novo usuário e quiser usar um login que já possui para entrar AWS e criar um AWS ambiente com padrões pré-configurados em select Regiões da AWS, use Inscrever-se em. AWS

Aplique uma política gerenciada para acesso administrativo ao CodePipeline

Você deve conceder permissões para interagir com CodePipeline. A maneira mais rápida de fazer isso é aplicando a política gerenciada `AWSCodePipeline_FullAccess` ao usuário administrativo.

> **Note**
>
> A `AWSCodePipeline_FullAccess` política inclui permissões que permitem que o usuário do console transmita uma função do IAM para CodePipeline outra pessoa Serviços da AWS. Isso permite que o serviço assuma a função e execute ações em seu nome. As permissões `iam:PassRole` são aplicadas quando você anexa a política a um usuário, função ou grupo. Certifique-se de que a política só é aplicada a usuários confiáveis. Quando os usuários com essas permissões usam o console para criar ou editar um pipeline, as seguintes opções ficam disponíveis:
>
> - Crie uma função CodePipeline de serviço ou escolha uma existente e passe a função para CodePipeline
> - Pode escolher criar uma regra de CloudWatch Eventos para detecção de alterações e passar a função de serviço de CloudWatch CloudWatch Eventos para Eventos
>
> Para obter mais informações, consulte Conceder permissões a um usuário para passar uma função para um AWS service (Serviço da AWS).

> **Note**
>
> A `AWSCodePipeline_FullAccess` política fornece acesso a todas as CodePipeline ações e recursos aos quais o usuário do IAM tem acesso, bem como a todas as ações possíveis ao criar estágios em um pipeline, como criar estágios que incluem CodeDeploy o Elastic Beanstalk ou o Amazon S3. Como uma das melhores práticas, você deve conceder a indivíduos apenas as permissões de que precisam para executar suas tarefas. Para obter mais informações sobre como restringir os usuários do IAM a um conjunto limitado de CodePipeline ações e recursos, consulteRemover permissões da função CodePipeline de serviço.

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em Centro de Identidade do AWS IAM:

  Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do Centro de Identidade do AWS IAM .

- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criando um perfil para um provedor de identidades de terceiros (federação) no Guia do Usuário do IAM.

- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.
  - (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

Instale o AWS CLI

Para chamar CodePipeline comandos da AWS CLI em uma máquina de desenvolvimento local, você deve instalar a AWS CLI. Essa etapa é opcional se você pretende começar usando somente as etapas deste guia para o CodePipeline console.

Para instalar e configurar o AWS CLI

1. Em sua máquina local, baixe e instale AWS CLI o. Isso permitirá que você interaja com a linha CodePipeline de comando. Para obter mais informações, consulte Como configurar a interface de linha de AWS comando.

   > **Note**
   >
   > CodePipeline funciona somente com AWS CLI as versões 1.7.38 e posteriores. Para determinar qual versão do AWS CLI que você pode ter instalado, execute o comandoaws --version. Para atualizar uma versão mais antiga do AWS CLI para a versão mais recente, siga as instruções em Desinstalando o e AWS CLI, em seguida, siga as instruções em Instalando o. AWS Command Line Interface

2. Configure o AWS CLI com o configure comando, da seguinte forma:

```
aws configure
```

Quando solicitado, especifique a chave de AWS acesso e a chave de acesso AWS secreta do usuário do IAM com CodePipeline o qual você usará. Quando solicitado pelo nome da região padrão, especifique a região em que você criará o pipeline, como`us-east-2`. Quando solicitado pelo formato de saída padrão, especifique `json`. Por exemplo:

```
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 obter mais informações sobre o IAM, chaves de acesso e chaves secretas, consulte Gerenciamento de chaves de acesso para usuários do IAM e Como obter as credenciais?
>
> Para obter mais informações sobre as regiões e os endpoints disponíveis CodePipeline, consulte AWS CodePipeline endpoints e cotas.

Abra o console para CodePipeline

- Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

Próximas etapas

Você concluiu os pré-requisitos. Você pode começar a usar CodePipeline. Para começar a trabalhar com o CodePipeline, consulte CodePipeline tutoriais.

Integrações de produtos e serviços com CodePipeline

Por padrão, AWS CodePipeline é integrado a vários produtos Serviços da AWS e serviços de parceiros. Use as informações nas seções a seguir para ajudá-lo a configurar CodePipeline a integração com os produtos e serviços que você usa.

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com este serviço.

Tópicos

- Integrações com tipos de CodePipeline ação
- Integrações gerais com CodePipeline
- Exemplos da comunidade

Integrações com tipos de CodePipeline ação

As informações de integrações neste tópico são organizadas por tipo de CodePipeline ação.

Tópicos

- Integrações de ações de origem
- Integrações de ações de compilação
- Integrações de ações de teste
- Implantar integrações de ações
- Integração da ação de aprovação ao Amazon Simple Notification Service
- Integrações de ações de invocação

Integrações de ações de origem

As informações a seguir são organizadas por tipo de CodePipeline ação e podem ajudá-lo a configurar CodePipeline a integração com os seguintes provedores de ação de origem.

Tópicos

- Ações de origem do Amazon ECR
- Ações de origem do Amazon S3

- Conexões com o Bitbucket Cloud GitHub (via GitHub aplicativo), GitHub Enterprise Server e GitLab.com GitLab autogerenciado
- CodeCommit ações de origem
- GitHub (via aplicativo OAuth) ações de origem

Ações de origem do Amazon ECR

O Amazon ECR é um serviço de repositório de imagens AWS do Docker. Os comandos enviar e receber docker são utilizados para carregar imagens de docker em seu repositório. O URI e a imagem do repositório do Amazon ECR são usados nas definições de tarefa do Amazon ECS para fazer referência a informações da imagem de origem.

Saiba mais:

- Para ver os parâmetros de configuração e um JSON/YAML trecho de exemplo, consulte Referência da ação de origem do Amazon ECR
- Criar um pipeline, estágios e ações
- Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR

Ações de origem do Amazon S3

O Amazon S3 é o armazenamento para a Internet. Você pode utilizar o Amazon S3 para armazenar e recuperar qualquer volume de dados, a qualquer momento, de qualquer lugar na web. Você pode configurar CodePipeline para usar um bucket Amazon S3 versionado como a ação de origem do seu código.

> **Note**
>
> O Amazon S3 também pode ser incluído em um pipeline como uma ação de implantação.

Saiba mais:

- Para ver os parâmetros de configuração e um JSON/YAML trecho de exemplo, consulte Referência da ação de origem do Amazon S3
- Etapa 1: criar um bucket de origem do S3 para a aplicação
- Criar um pipeline (CLI)

- CodePipeline usa a Amazon EventBridge (anteriormente Amazon CloudWatch Events) para detectar alterações em seu bucket de origem do Amazon S3. Consulte Integrações gerais com CodePipeline.

Conexões com o Bitbucket Cloud GitHub (via GitHub aplicativo), GitHub Enterprise Server e GitLab.com GitLab autogerenciado

As conexões (`CodeStarSourceConnection`ações) são usadas para acessar seu Bitbucket Cloud GitHub, GitHub Enterprise Server ou repositório GitLab autogerenciado de terceiros. GitLab.com

> **Note**
>
> Esse recurso não está disponível nas regiões Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Melbourne), Ásia-Pacífico (Osaka), África (Cidade do Cabo), Oriente Médio (Bahrein), Oriente Médio (EAU), Europa (Espanha), Europa (Zurique), Israel (Tel Aviv) ou (). AWS GovCloud US-West Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline. Para considerações sobre essa ação na região Europa (Milão), consulte a nota em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

Bitbucket Cloud	Você pode configurar CodePipeline para usar um repositório do Bitbucket Cloud como fonte do seu código. É necessário que você já tenha criado uma conta do Bitbucket e pelo menos um repositório do Bitbucket Cloud. Você pode adicionar uma ação de origem para o repositório do Bitbucket Cloud criando um pipeline ou editando um já existente. **Note** Você pode criar conexões para um repositório do Bitbucket Cloud. Não há suporte a tipos de provedores instalados do Bitbucket, como o Bitbucket Server. Você pode configurar recursos chamados conexões para permitir que seus pipelines acessem repositórios de código de terceiros. Ao configurar uma

	conexão, instale a aplicação de conector no repositório de código de terceiros e vincule-a à conexão criada. Para o Bitbucket Cloud, use a opção Bitbucket no console ou a ação `CodestarSourceConnection` na CLI. Consulte Conexões do Bitbucket Cloud. Você pode usar a opção Clone completo dessa ação para referenciar os metadados do Git do repositório a fim de que as ações downstream possam executar comandos do Git diretamente. Essa opção só pode ser usada por ações CodeBuild posteriores. Saiba mais: • Para ver os parâmetros de configuração e um JSON/YAML trecho de exemplo, consulte. CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com • Para visualizar um tutorial de conceitos básicos que cria um pipeline com uma origem do Bitbucket Cloud, consulte Conceitos básicos sobre conexões.
GitHub ou nuvem GitHub corporativa	Você pode configurar CodePipeline para usar um GitHub repositório como fonte do seu código. Você deve ter criado anteriormente uma GitHub conta e pelo menos um GitHub repositório. Você pode adicionar uma ação de origem ao seu GitHub repositório criando um pipeline ou editando um existente. Você pode configurar recursos chamados conexões para permitir que seus pipelines acessem repositórios de código de terceiros. Ao configurar uma conexão, instale a aplicação de conector no repositório de código de terceiros e vincule-a à conexão criada. Use a opção de provedor GitHub (via GitHub aplicativo) no console ou a `CodestarSourceConnection` ação na CLI. Consulte GitHub conexões. Você pode usar a opção Clone completo dessa ação para referenciar os metadados do Git do repositório a fim de que as ações downstream possam

	executar comandos do Git diretamente. Essa opção só pode ser usada por ações CodeBuild posteriores. Saiba mais: • Para ver os parâmetros de configuração e um JSON/YAML trecho de exemplo, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com • Para ver um tutorial que mostra como se conectar a um GitHub repositór io e usar a opção Clonagem completa, consulte. Tutorial: use o clone completo com uma fonte de GitHub pipeline • A ação atual GitHub (via GitHub aplicativo) é a ação de origem da versão 2 para GitHub. A ação GitHub (via aplicativo OAuth) é a ação da versão 1 GitHub gerenciada com a autenticação de token OAuth. Embora não seja recomendável usar a ação GitHub (via aplicativo OAuth), os pipelines existentes com a ação GitHub (via aplicativo OAuth) continuarão funcionan do sem nenhum impacto. Agora você pode usar uma ação CodeStarS ourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com de origem em seu pipeline que gerencia sua ação de GitHub origem com GitHub aplicativos. Se você tiver um pipeline que usa a ação GitHub (via aplicativo OAuth), veja as etapas para atualizá-lo para usar uma ação GitHub (via GitHub aplicativo) em. Atualizar uma ação de origem GitHub (via aplicativo OAuth) para uma ação de origem GitHub (via GitHub aplicativo)
GitHub Servidor corporativo	Você pode configurar CodePipeline para usar um repositório do GitHub Enterprise Server como fonte do seu código. Você deve ter criado anteriorm ente uma GitHub conta e pelo menos um GitHub repositório. Você pode adicionar uma ação de origem ao seu repositório do GitHub Enterprise Server criando um pipeline ou editando um existente. Você pode configurar recursos chamados conexões para permitir que seus pipelines acessem repositórios de código de terceiros. Ao configurar uma conexão, instale a aplicação de conector no repositório de código de terceiros e vincule-a à conexão criada.

	Use a opção GitHub Enterprise Server provider no console ou a `CodestarSourceConnection` ação na CLI. Consulte GitHub Conexões do Enterprise Server. Você pode usar a opção Clone completo dessa ação para referenciar os metadados do Git do repositório a fim de que as ações downstream possam executar comandos do Git diretamente. Essa opção só pode ser usada por ações CodeBuild posteriores. Saiba mais: • Para ver os parâmetros de configuração e um JSON/YAML trecho de exemplo, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com • Para ver um tutorial que mostra como se conectar a um GitHub repositório e usar a opção Clonagem completa, consulte. Tutorial: use o clone completo com uma fonte de GitHub pipeline
GitLab.com	Você pode configurar CodePipeline para usar um GitLab.com repositório como fonte do seu código. Você deve ter criado anteriormente uma GitLab.com conta e pelo menos um GitLab.com repositório. Você pode adicionar uma ação de origem ao seu GitLab.com repositório criando um pipeline ou editando um existente. Use a opção de GitLabprovedor no console ou a `CodestarSourceConnection` ação com o `GitLab` provedor na CLI. Consulte GitLab.com conexões. Saiba mais: • Para ver os parâmetros de configuração e um JSON/YAML trecho de exemplo, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com

GitLab autogerenciado	Você pode configurar CodePipeline para usar uma instalação GitLab autogerenciada como fonte do seu código. Você deve ter criado uma GitLab conta anteriormente e ter uma assinatura autogerenciada GitLab (Enterprise Edition ou Community Edition). Você pode adicionar uma ação de origem ao seu repositório GitLab autogerenciado criando um pipeline ou editando um existente. Você pode configurar recursos chamados conexões para permitir que seus pipelines acessem repositórios de código de terceiros. Ao configurar uma conexão, instale a aplicação de conector no repositório de código de terceiros e vincule-a à conexão criada. Use a opção de provedor GitLab autogerenciado no console ou a `CodestarSourceConnection` ação na CLI. Consulte Conexões para GitLab autogerenciamento. Você pode usar a opção Clone completo dessa ação para referenciar os metadados do Git do repositório a fim de que as ações downstream possam executar comandos do Git diretamente. Essa opção só pode ser usada por ações CodeBuild posteriores. Saiba mais: • Para ver os parâmetros de configuração e um JSON/YAML trecho de exemplo, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com • Para conhecer as etapas para criar uma conexão com esse tipo de provedor, consulte Conexões para GitLab autogerenciamento.

CodeCommit ações de origem

CodeCommit é um serviço de controle de versão que você pode usar para armazenar e gerenciar ativos de maneira privada (como documentos, código-fonte e arquivos binários) em nuvem. Você pode configurar CodePipeline para usar uma ramificação em um CodeCommit repositório como fonte do seu código. Crie o repositório e associe-o a um diretório de trabalho em sua máquina local. Em seguida, você pode criar um pipeline que use a ramificação como parte de uma ação de origem em

um estágio. Você pode se conectar ao CodeCommit repositório criando um pipeline ou editando um existente.

Você pode usar a opção Clone completo dessa ação para referenciar os metadados do Git do repositório a fim de que as ações downstream possam executar comandos do Git diretamente. Essa opção só pode ser usada por ações CodeBuild posteriores.

Saiba mais:

- Para ver os parâmetros de configuração e um JSON/YAML trecho de exemplo, consulte. CodeCommit referência de ação de origem
- Tutorial: criar um pipeline simples (CodeCommit repositório)
- CodePipeline usa o Amazon CloudWatch Events para detectar alterações nos CodeCommit repositórios usados como fonte para um pipeline. Toda ação de origem tem uma regra de evento correspondente. Essa regra de evento inicia o pipeline quando ocorre uma alteração no repositório. Consulte Integrações gerais com CodePipeline.

GitHub (via aplicativo OAuth) ações de origem

A ação GitHub (via aplicativo OAuth) é a ação da versão 1 GitHub gerenciada com aplicativos OAuth. Nas regiões disponíveis, você também pode usar uma ação de CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com origem em seu pipeline que gerencia sua ação de GitHub origem com GitHub aplicativos. Se você tiver um pipeline que usa a ação GitHub (via aplicativo OAuth), veja as etapas para atualizá-lo para usar uma ação GitHub (via GitHub aplicativo) em. Atualizar uma ação de origem GitHub (via aplicativo OAuth) para uma ação de origem GitHub (via GitHub aplicativo)

> **Note**
>
> Embora não seja recomendável usar a ação GitHub (via aplicativo OAuth), os pipelines existentes com a ação GitHub (via aplicativo OAuth) continuarão funcionando sem nenhum impacto.

Saiba mais:

- Para obter mais informações sobre o acesso OAuth-based GitHub (via aplicativo OAuth) em contraste com o acesso baseado em aplicativo GitHub , consulte. https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps
- Para ver um apêndice que contém os detalhes da ação GitHub (via aplicativo OAuth), consulte. Apêndice A: GitHub (por meio do aplicativo OAuth) ações de origem

Integrações de ações de compilação

As informações a seguir são organizadas por tipo de CodePipeline ação e podem ajudá-lo a configurar CodePipeline a integração com os seguintes provedores de ações de compilação.

Tópicos

- CodeBuild criar ações
- CloudBees criar ações
- Ações de compilação e publicação do Amazon ECR
- Ações de compilações Jenkins
- TeamCity criar ações

CodeBuild criar ações

O CodeBuild é um serviço de compilação totalmente gerenciado que compila o código-fonte, executa testes em unidades e produz artefatos prontos para implantação.

Você pode adicionar CodeBuild como uma ação de construção ao estágio de construção de um pipeline. Para obter mais informações, consulte a Referência de configuração de CodePipeline ação paraAWS CodeBuild referência de ação de construção e teste.

> **Note**
>
> CodeBuild também pode ser incluído em um pipeline como uma ação de teste, com ou sem uma saída de compilação.

Saiba mais:

- Para ver os parâmetros de configuração e um JSON/YAML trecho de exemplo, consulte. AWS CodeBuild referência de ação de construção e teste

- O que é CodeBuild?
- CodeBuild— Serviço de construção totalmente gerenciado

CloudBees criar ações

Você pode configurar CodePipeline para usar CloudBeespara criar ou testar seu código em uma ou mais ações em um pipeline.

Saiba mais:

- re:Invent 2017: Cloud First com AWS

Ações de compilação e publicação do Amazon ECR

O Amazon ECR é um serviço de repositório de imagens AWS do Docker. Os comandos enviar e receber docker são utilizados para carregar imagens de docker em seu repositório.

Você pode adicionar a ação `ECRBuildAndPublish` ao pipeline para automatizar a compilação e o envio de uma imagem. Para obter mais informações, consulte a referência de configuração da CodePipeline ação paraReferência da ação de compilação `ECRBuildAndPublish`.

Ações de compilações Jenkins

Você pode configurar CodePipeline para usar o Jenkins CI para criar ou testar seu código em uma ou mais ações em um pipeline. Você deve ter criado previamente um projeto Jenkins e instalado e configurado o CodePipeline plug-in para Jenkins para esse projeto. Você pode conectar-se ao projeto do Jenkins criando um novo pipeline ou editando um existente.

O acesso para Jenkins é configurado por projeto. Você deve instalar o CodePipeline plug-in para Jenkins em todas as instâncias do Jenkins com as quais deseja usar. CodePipeline Você também deve configurar o CodePipeline acesso ao projeto Jenkins. Proteja seu projeto Jenkins configurando-o para aceitar somente HTTPS/SSL conexões. Se seu projeto Jenkins estiver instalado em uma instância do Amazon EC2, considere fornecer AWS suas credenciais AWS CLI instalando-as em cada instância. Em seguida, configure um AWS perfil nessas instâncias com as credenciais que você deseja usar para conexões. Essa é uma alternativa para adicioná-las e armazená-las na interface web do Jenkins.

Saiba mais:

- Acesso ao Jenkins
- Tutorial: Criar um pipeline de quatro estágios

TeamCity criar ações

Você pode configurar CodePipeline para usar TeamCitypara criar e testar seu código em uma ou mais ações em um pipeline.

Saiba mais:

- TeamCity Plugin para CodePipeline

Integrações de ações de teste

As informações a seguir são organizadas por tipo de CodePipeline ação e podem ajudá-lo a configurar CodePipeline a integração com os seguintes provedores de ações de teste.

Tópicos

- CodeBuild ações de teste
- AWS Device Farm ações de teste
- Ações de teste do Ghost Inspector
- OpenText LoadRunner Ações de teste na nuvem
- Automação de teste do Reflect

CodeBuild ações de teste

CodeBuildé um serviço de criação totalmente gerenciado na nuvem. CodeBuild compila seu código-fonte, executa testes de unidade e produz artefatos prontos para serem implantados.

Você pode adicionar CodeBuild a um pipeline como uma ação de teste. Para obter mais informações, consulte a Referência de configuração do CodePipeline para AWS CodeBuild referência de ação de construção e teste.

> **Note**
>
> CodeBuild também pode ser incluído em um pipeline como uma ação de construção, com um artefato de saída de construção obrigatório.

Saiba mais:

- Para ver os parâmetros de configuração e um JSON/YAML trecho de exemplo, consulte. AWS CodeBuild referência de ação de construção e teste
- O que é CodeBuild?

AWS Device Farm ações de teste

O AWS Device Farm é um serviço de teste de aplicativos que pode ser usado para testar e interagir com aplicativos Android, iOS e web em telefones e tablets reais físicos. Você pode configurar CodePipeline para usar AWS Device Farm para testar seu código em uma ou mais ações em um pipeline. AWS Device Farm permite que você faça upload de seus próprios testes ou use testes de compatibilidade integrados e sem scripts. Como o teste é executado em paralelo, vários dispositivos começam a ser testados em questão de minutos. Um relatório de teste que contém resultados de alto nível, registros de baixo nível, capturas de tela de pixel a pixel e dados de desempenho é atualizado à medida que os testes são concluídos. AWS Device Farm suporta testes de aplicativos nativos e híbridos para Android, iOS e Fire OS, incluindo aqueles criados com Titanium PhoneGap, Xamarin, Unity e outras estruturas. É compatível com acesso remoto de aplicativos Android, o que permite interagir diretamente com dispositivos de teste.

Saiba mais:

- Para ver os parâmetros de configuração e um JSON/YAML trecho de exemplo, consulte. AWS Device Farm referência de ação de teste
- O que é AWS Device Farm?
- Usando AWS Device Farm em um estágio CodePipeline de teste

Ações de teste do Ghost Inspector

Você pode configurar CodePipeline para usar o Ghost Inspector para testar seu código em uma ou mais ações em um pipeline.

Saiba mais:

- Documentação do Ghost Inspector para integração de serviços com CodePipeline

OpenText LoadRunner Ações de teste na nuvem

Você pode configurar CodePipeline para usar o OpenText LoadRunner Cloud em uma ou mais ações em um pipeline.

Saiba mais:

- LoadRunner Documentação na nuvem para integração com CodePipeline

Automação de teste do Reflect

O Reflect é a solução de automação de AI-powered testes que permite que você simplifique os testes e supere os desafios dos processos manuais. Com a automação de testes no-code, o Reflect simplifica a criação, a execução e a manutenção de testes, permitindo que você crie testes robustos e repetíveis sem exigir nenhum conhecimento técnico. Ao eliminar a complexidade e garantir o mínimo de interrupção nos fluxos de trabalho, ele ajuda a agilizar os testes e a disponibilizar sempre aplicativos de alta qualidade com confiança.

Saiba mais:

- AWS CodePipeline teste a integração com o Reflect

Implantar integrações de ações

As informações a seguir são organizadas por tipo de CodePipeline ação e podem ajudá-lo a configurar CodePipeline a integração com os seguintes provedores de ações de implantação.

Tópicos

- Ações de implantação do Amazon EC2
- Amazon Elastic Kubernetes Service EKS implementa ações
- Ação de implantação do Amazon S3
- AWS AppConfig implantar ações
- CloudFormation implantar ações

- CloudFormation StackSets implantar ações
- Ação de implantação do Amazon ECS
- Ações de implantação do Elastic Beanstalk
- OpsWorks implantar ações
- Ações de implantação do Service Catalog
- Ação de implantação do Amazon Alexa
- CodeDeploy implantar ações
- XebiaLabs implantar ações

Ações de implantação do Amazon EC2

O Amazon EC2 permite que você crie e gerencie a computação na nuvem. Você pode adicionar uma ação a um pipeline que use o Amazon EC2 como um provedor de implantação que implanta o aplicativo às instâncias.

Saiba mais:

- Consulte a página de referência da ação em Referência de ação do Amazon EC2.
- Para ver um tutorial, consulte Tutorial: Implante em instâncias do Amazon EC2 com CodePipeline.

Amazon Elastic Kubernetes Service EKS implementa ações

O Amazon EKS permite que você crie e gerencie clusters Kubernetes. Você pode adicionar uma ação a um pipeline que use o Amazon EKS como um provedor de implantação que implanta a imagem no cluster. Você pode usar modelos do Helm ou arquivos de manifesto do Kubernetes.

Saiba mais:

- Consulte a página de referência da ação em Referência de ação de implantação do Amazon Elastic Kubernetes Service EKS.
- Para ver um tutorial, consulte Tutorial: Implante no Amazon EKS com CodePipeline.

Ação de implantação do Amazon S3

O Amazon S3 é o armazenamento para a Internet. Você pode utilizar o Amazon S3 para armazenar e recuperar qualquer volume de dados, a qualquer momento, de qualquer lugar na web. Você pode adicionar uma ação a um pipeline que usa o Amazon S3 como provedor de implantação.

> **Note**
>
> O Amazon S3 também pode ser incluído em um pipeline como uma ação de origem.

Saiba mais:

- Criar um pipeline, estágios e ações
- Tutorial: Criar um pipeline que usa o Amazon S3 como um provedor de implantação

AWS AppConfig implantar ações

AWS AppConfig é a capacidade de AWS Systems Manager criar, gerenciar e implantar rapidamente configurações de aplicativos. Você pode usar AppConfig com aplicativos hospedados em instâncias EC2 AWS Lambda, contêineres, aplicativos móveis ou dispositivos de IoT.

Saiba mais:

- CodePipeline Referência de configuração de ação para Referência da ação de implantação do AWS AppConfig
- Tutorial: Crie um pipeline que use AWS AppConfig como provedor de implantação

CloudFormation implantar ações

CloudFormationoferece aos desenvolvedores e administradores de sistemas uma maneira fácil de criar e gerenciar uma coleção de AWS recursos relacionados, usando modelos para provisionar e atualizar esses recursos. Você pode usar os modelos de exemplo do serviço ou criar outros modelos. Os modelos descrevem os AWS recursos e quaisquer dependências ou parâmetros de tempo de execução necessários para executar seu aplicativo.

O AWS Serverless Application Model (AWS SAM) se estende CloudFormation para fornecer uma maneira simplificada de definir e implantar aplicativos sem servidor. AWS O SAM é compatível com APIs do Amazon API Gateway, funções AWS Lambda e tabelas do Amazon DynamoDB. Você pode

usar CodePipeline com CloudFormation e o AWS SAM para fornecer continuamente seus aplicativos sem servidor.

Você pode adicionar uma ação a um pipeline usado AWS CloudFormation como provedor de implantação. Ao usar AWS CloudFormation como provedor de implantação, você pode agir em CloudFormation pilhas e conjuntos de alterações como parte da execução de um pipeline. CloudFormation pode criar, atualizar, substituir e excluir pilhas e conjuntos de alterações quando um pipeline é executado. Como resultado, AWS recursos personalizados podem ser criados, provisionados, atualizados ou encerrados durante a execução de um pipeline, de acordo com as especificações fornecidas nos CloudFormation modelos e nas definições de parâmetros.

Saiba mais:

- CodePipeline Referência de configuração de ação para Referência da ação de implantação do CloudFormation
- Entrega contínua com CodePipeline — Aprenda a usar CodePipeline para criar um fluxo de trabalho de entrega contínua para CloudFormation.
- Automatizando a implantação de Lambda-based aplicativos — Aprenda a usar o modelo de aplicativo AWS sem servidor e a criar um fluxo de trabalho de entrega contínua CloudFormation para seu aplicativo. Lambda-based

CloudFormation StackSets implantar ações

CloudFormationoferece uma maneira de implantar recursos em várias contas e AWS regiões.

Você pode usar CodePipeline with CloudFormation para atualizar sua definição de conjunto de pilhas e implantar atualizações em suas instâncias.

Você pode adicionar as seguintes ações a um pipeline para usar AWS CloudFormation StackSets como provedor de implantação.

- CloudFormationStackSet
- CloudFormationStackInstances

Saiba mais:

- CodePipeline Referência de configuração de ação para CloudFormation Referência da ação de implantação do StackSets

- Tutorial: criar um pipeline com AWS CloudFormation StackSets ações de implantação

Ação de implantação do Amazon ECS

O Amazon ECS é um serviço de gerenciamento de contêineres de alta escalabilidade e alto desempenho que permite a você executar aplicações baseadas em contêiner na Nuvem AWS. Ao criar um pipeline, você pode selecionar o Amazon ECS como um provedor de implantação. Uma alteração de código no repositório de controle de origem aciona o pipeline para que ele crie uma nova imagem do Docker, a envie por push ao seu registro de contêiner e, em seguida, implante a imagem atualizada no Amazon ECS. Você também pode usar a ação do provedor ECS (Blue/Green) CodePipeline para rotear e implantar tráfego para o Amazon ECS com. CodeDeploy

Saiba mais:

- O que é o Amazon ECS?
- Tutorial: Implantação contínua com CodePipeline
- Criar um pipeline, estágios e ações
- Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR

Ações de implantação do Elastic Beanstalk

O Elastic Beanstalk é um serviço para implantar e escalar aplicativos e serviços web desenvolvidos com Java, .NET, PHP, Python, Ruby Node.js, Go e Docker em servidores conhecidos, como Apache, Nginx, Passenger e IIS. Você pode configurar CodePipeline para usar o Elastic Beanstalk para implantar seu código. Você pode criar a aplicação e o ambiente do Elastic Beanstalk para serem usados em uma ação de implantação em um estágio antes da criação do pipeline ou durante o uso do assistente Criar pipeline.

> **Note**
>
> Esse atributo não está disponível na região Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Melbourne), Oriente Médio (EAU), Europa (Espanha) ou Europa (Zurique). Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline.

Saiba mais:

- Conceitos básicos do Elastic Beanstalk
- Criar um pipeline, estágios e ações

OpsWorks implantar ações

OpsWorks é um serviço de gerenciamento de configuração que ajuda você a configurar e operar aplicativos de todas as formas e tamanhos usando o Chef. Usando AWS OpsWorks Stacks, você pode definir a arquitetura do aplicativo e a especificação de cada componente, incluindo instalação de pacotes, configuração de software e recursos como armazenamento. Você pode configurar CodePipeline para usar para AWS OpsWorks Stacks implantar seu código em conjunto com livros de receitas e aplicativos personalizados do Chef em. OpsWorks

- Custom Chef Cookbooks — OpsWorks usa Chef Cookbooks para lidar com tarefas como instalar e configurar pacotes e implantar aplicativos.
- Aplicativos — Um OpsWorks aplicativo consiste em código que você deseja executar em um servidor de aplicativos. O código da aplicação é armazenado em um repositório, por exemplo, um bucket do Amazon S3.

Antes de criar o pipeline, você cria a OpsWorks pilha e a camada. Você pode criar o OpsWorks aplicativo para usar em uma ação de implantação em um estágio antes de criar o pipeline ou ao usar o assistente Create Pipeline.

CodePipeline Atualmente, o suporte para OpsWorks está disponível somente na região Leste dos EUA (Norte da Virgínia) (us-east-1).

Saiba mais:

- Usando CodePipeline com AWS OpsWorks Stacks
- Livros de receitas e receitas
- AWS OpsWorks Apps

Ações de implantação do Service Catalog

O Service Catalog permite que as organizações criem e gerenciem catálogos de produtos aprovados para uso em AWS.

Você pode configurar CodePipeline para implantar atualizações e versões dos seus modelos de produto no Service Catalog. Você pode criar o produto do Service Catalog para ser usado em uma ação de implantação e, em seguida, usar o assistente Criar pipeline para criar o pipeline.

Saiba mais:

- Tutorial: Criar um pipeline que realiza a implantação no Service Catalog
- Criar um pipeline, estágios e ações

Ação de implantação do Amazon Alexa

O Amazon Alexa Skills Kit permite que você crie e distribua habilidades baseadas na nuvem para usuários de dispositivos. Alexa-enabled

> **Note**
>
> Este atributo não está disponível na região Ásia-Pacífico (Hong Kong) ou Europa (Milão). Para usar outras ações de implantação disponíveis nessa região, consulte Implantar integrações de ações.

Você pode adicionar uma ação a um pipeline que usa o Alexa Skills Kit como provedor de implantação. Alterações na origem são detectadas pelo pipeline e, depois, ele implanta atualizações na skill da Alexa no serviço da Alexa.

Saiba mais:

- Tutorial: Criar um pipeline que implanta uma skill do Amazon Alexa

CodeDeploy implantar ações

CodeDeploycoordena implantações de aplicativos em EC2/on-premises instâncias da Amazon, plataformas de computação Amazon Elastic Container Service e plataformas de computação sem servidor AWS Lambda . Você pode configurar CodePipeline para usar CodeDeploy para implantar seu código. Você pode criar o aplicativo do CodeDeploy , a implantação e o grupo de implantação para usar em uma ação de implantação em um estágio antes de criar o pipeline ou quando usar o assistente Create Pipeline (Criar pipeline).

Saiba mais:

- Etapa 3: criar um aplicativo no CodeDeploy
- Tutorial: criar um pipeline simples (CodeCommit repositório)

XebiaLabs implantar ações

Você pode configurar CodePipeline para usar XebiaLabspara implantar seu código em uma ou mais ações em um pipeline.

Saiba mais:

- Usando o XL Deploy com CodePipeline

Integração da ação de aprovação ao Amazon Simple Notification Service

O Amazon SNS é um serviço de notificações por push rápido, flexível e totalmente gerenciado que permite enviar mensagens individuais ou distribuir mensagens para um grande número de destinatários. Com o Amazon SNS, é simples e econômico enviar notificações por push para usuários de dispositivos móveis, destinatários de e-mail ou até mesmo enviar mensagens a outros serviços distribuídos.

Ao criar uma solicitação de aprovação manual no CodePipeline, você pode, opcionalmente, publicar em um tópico no Amazon SNS para que todos os usuários do IAM inscritos nela sejam notificados de que a ação de aprovação está pronta para ser revisada.

Saiba mais:

- O que é o Amazon SNS?
- Conceder permissões ao Amazon SNS para uma função CodePipeline de serviço

Integrações de ações de invocação

As informações a seguir são organizadas por tipo de CodePipeline ação e podem ajudá-lo a configurar CodePipeline a integração com os seguintes provedores de ação de invocação.

Tópicos

- Ações invocar do Amazon Inspector

- Ações de invocação do Lambda
- Ações de invocação do Step Functions

Ações invocar do Amazon Inspector

O Amazon Inspector é um serviço de gerenciamento de vulnerabilidades que descobre workloads automaticamente e as verifica continuamente em busca de vulnerabilidades de software e exposições não intencionais da rede. O Amazon Inspector oferece suporte a vários formatos de arquivamento, inclusive tar e war, e o Amazon Inspector oferece suporte a binários, inclusive binários Rust e Go.

Você pode configurar a CodePipeline `InspectorScan` ação para automatizar a digitalização do código-fonte ou do repositório de imagens do Amazon ECR em busca de vulnerabilidades.

Saiba mais:

- CodePipeline referência de configuração de ação para Referência da ação invocar `InspectorScan` do Amazon Inspector

Ações de invocação do Lambda

O Lambda permite executar códigos sem provisionar ou gerenciar servidores. Você pode configurar CodePipeline para usar as funções do Lambda para adicionar flexibilidade e funcionalidade aos seus pipelines. Você pode criar a função do Lambda para ser adicionada como uma ação em um estágio antes da criação do pipeline ou durante o uso do assistente Criar pipeline.

Saiba mais:

- CodePipeline Referência de configuração de ação para AWS Lambda invocar referência de ação
- Invocar um AWS Lambda função em um pipeline em CodePipeline

Ações de invocação do Step Functions

O Step Functions permite criar e configurar máquinas de estado. Você pode configurar CodePipeline para usar ações de invocação do Step Functions para acionar execuções de máquinas de estado.

Saiba mais:

- CodePipeline Referência de configuração de ação para AWS Step Functions invocar referência de ação
- Tutorial: use uma ação de AWS Step Functions invocação em um pipeline

Integrações gerais com CodePipeline

As AWS service (Serviço da AWS) integrações a seguir não se baseiam em tipos de CodePipeline ação.

Amazon CloudWatch	A Amazon CloudWatch monitora seus AWS recursos. Saiba mais: • O que é a Amazon CloudWatch?
Amazon EventBridge	EventBridgeA Amazon é um serviço web que detecta alterações em você Serviços da AWS com base nas regras que você define e invoca uma ação em uma ou mais especificadas Serviços da AWS quando ocorre uma alteração. • Inicie a execução de um pipeline automaticamente quando algo mudar — Você pode configurar CodePipeline como alvo nas regras configuradas na Amazon EventBridge. Essa ação faz o pipeline iniciar automaticamente quando outro serviço muda. Saiba mais: • O que é a Amazon EventBridge? • Inicie um funil em CodePipeline. • CodeCommit ações de origem e EventBridge • Receba notificações quando o estado de um pipeline mudar — Você pode configurar EventBridge regras para detectar e reagir às mudanças no estado de execução de um pipeline, estágio ou ação. Saiba mais: • Monitoramento de CodePipeline eventos

	• Tutorial: configurar uma regra de CloudWatch eventos para receber notificações por e-mail sobre mudanças no estado do pipeline
AWS Cloud9	AWS Cloud9 é um IDE on-line, que você acessa por meio do seu navegador da web. O IDE oferece uma experiência de edição de código completa com suporte para várias linguagens de programação e depuradores de tempo de execução, bem como um terminal integrado . Em segundo plano, uma instância do Amazon EC2 hospeda um ambiente de AWS Cloud9 desenvolvimento. Para obter mais informaçõ es, consulte o Guia do usuário do AWS Cloud9. Saiba mais: • Configurar o AWS Cloud9
AWS CloudTrail	CloudTrailcaptura chamadas de AWS API e eventos relacionados feitos por ou em nome de uma AWS conta e entrega arquivos de log para um bucket do Amazon S3 que você especificar. Você pode configura r CloudTrail para capturar chamadas de API do CodePipeline console, CodePipeline comandos da AWS CLI e da CodePipeline API. Saiba mais: • Registrando chamadas de CodePipeline API com AWS CloudTrail
AWS CodeStar Notificações	Você pode configurar notificações para que os usuários saibam sobre alterações importantes, como quando um pipeline inicia a execução. Para obter mais informações, consulte Criar uma regra de notificação.

AWS Key Management Service	O AWS KMS é um serviço gerenciado que facilita a criação e o controle de chaves de criptografia usadas para criptografar seus dados. Por padrão, CodePipeline usa AWS KMS para criptografar artefatos para pipelines armazenados em buckets do Amazon S3. Saiba mais: • Para criar um pipeline que usa um bucket de origem, um bucket de artefatos e uma função de serviço de uma AWS conta e CodeDeploy recursos de uma AWS conta diferente, você deve criar uma chave KMS gerenciada pelo cliente, adicionar a chave ao pipeline e configurar políticas e funções da conta para permitir o acesso entre contas. Para obter mais informações, consulte Crie um pipeline CodePipeline que use recursos de outro AWS account. • Para criar um pipeline a partir de uma AWS conta que implanta uma CloudFormation pilha em outra AWS conta, você deve criar uma chave KMS gerenciada pelo cliente, adicionar a chave ao pipeline e configurar políticas e funções da conta para implantar a pilha em outra conta. AWS Para obter mais informações, consulte Como eu uso CodePipeline para implantar uma AWS CloudFormation pilha em uma conta diferente? • Para configurar a criptografia do lado do servidor para o bucket de artefatos S3 do seu pipeline, você pode usar a chave KMS AWS gerenciada padrão ou criar uma chave KMS gerenciada pelo cliente e configurar a política do bucket para usar a chave de criptografia. Para obter mais informações, consulte Configure a criptografia do lado do servidor para artefatos armazenados no Amazon S3 para CodePipeline. Para uma chave do AWS KMS key, você pode usar o ID da chave, o ARN da chave ou o ARN do alias. **Note** Os aliases são reconhecidos apenas na conta que criou a chave do KMS. Para ações entre contas, você só pode usar o

	ID da chave ou o ARN da chave para identificar a chave. Cross-account as ações envolvem o uso da função da outra conta (AccountB), portanto, especificar o ID da chave usará a chave da outra conta (AccountB).

Exemplos da comunidade

As seguintes seções fornecem links para publicações no blog, artigos e exemplos fornecidos pela comunidade.

> **Note**
>
> Esses links são fornecidos apenas para fins informativos e não devem ser considerados uma lista abrangente ou um endosso do conteúdo dos exemplos. AWS não é responsável pelo conteúdo ou pela precisão do conteúdo externo.

Tópicos

- Exemplos de integração: publicações no blog

Exemplos de integração: publicações no blog

- Rastreando o status da AWS CodePipeline compilação a partir do repositório Git de terceiros

  Aprenda a configurar recursos que exibirão seu pipeline e criarão o status da ação no repositório de terceiros, tornando mais fácil para o desenvolvedor monitorar o status sem mudar de contexto.

  Publicado em março de 2021

- Completo CI/CD com AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, e AWS CodePipeline

  Saiba como configurar um pipeline que usa os CodeDeploy serviços CodeCommit,, CodePipeline CodeBuild, e para compilar, criar e instalar um aplicativo Java com controle de versão em um conjunto de instâncias Linux do Amazon EC2.

  Publicado em setembro de 2020

- Testando e criando CI/CD pipelines para AWS Step Functions

  Saiba como configurar recursos que coordenarão a máquina de estado do Step Functions e o pipeline.

  Publicado em março de 2020

- Implementando DevSecOps usando CodePipeline

  Saiba como usar um CI/CD pipeline para CodePipeline automatizar os controles de segurança preventivos e de detetive. Esta postagem aborda como usar um pipeline para criar um grupo de segurança simples e realizar verificações de segurança durante os estágios de origem, teste e produção para melhorar a postura de segurança de suas AWS contas.

  Publicado em março de 2017

- Implantação contínua no Amazon ECS usando CodePipeline, CodeBuild, Amazon ECR e CloudFormation

  Saiba como criar um pipeline de implantação contínua no Amazon Elastic Container Service (Amazon ECS). Os aplicativos são entregues como contêineres Docker usando CodePipeline CodeBuild,, Amazon ECR e. CloudFormation

  - Baixe um CloudFormation modelo de amostra e instruções para usá-lo para criar seu próprio pipeline de implantação contínua no repositório ECS Reference Architecture: Continuous Deployment. GitHub

  Publicado em janeiro de 2017

- Implantação contínua para aplicações sem servidor

  Saiba como usar uma coleção de Serviços da AWS para criar um pipeline de implantação contínua para seus aplicativos sem servidor. Você usará o Serverless Application Model (SAM) para definir o aplicativo e seus recursos e CodePipeline para orquestrar a implantação do aplicativo.

  - Visualize um exemplo de aplicação gravado na estrutura Go with the Gin e um shim de proxy do API Gateway.

  Publicado em dezembro de 2016

- Dimensionando DevOps implantações com CodePipeline o Dynatrace

Saiba como usar as soluções de monitoramento da Dynatrace para escalar pipelines CodePipeline, analisar automaticamente as execuções de testes antes que o código seja confirmado e manter os prazos de entrega ideais.

Publicado em novembro de 2016

- Crie um pipeline para AWS Elastic Beanstalk CodePipeline usar AWS CloudFormation e CodeCommit

  Saiba como implementar a entrega contínua em um CodePipeline pipeline para um aplicativo em AWS Elastic Beanstalk. Todos os AWS recursos são provisionados automaticamente por meio do uso de um CloudFormation modelo. Este passo a passo também incorpora CodeCommit e AWS Identity and Access Management (IAM).

  Publicado em maio de 2016

- Automatize CodeCommit e CodePipeline entre CloudFormation

  Use CloudFormation para automatizar o provisionamento de AWS recursos para um pipeline de entrega contínua que usa CodeCommit, CodePipeline, e. CodeDeploy AWS Identity and Access Management

  Publicado em abril de 2016

- Crie um Cross-Account pipeline em AWS CodePipeline

  Saiba como automatizar o provisionamento do acesso a pipelines entre contas no AWS CodePipeline usando o AWS Identity and Access Management. Inclui exemplos em um CloudFormation modelo.

  Publicado em março de 2016

- Explorando a parte ASP.NET principal 2: entrega contínua

  Saiba como criar um sistema completo de entrega contínua para um aplicativo ASP.NET principal usando CodeDeploy AWS CodePipeline e.

  Publicado em março de 2016

- Crie um pipeline usando o AWS CodePipeline console

  Saiba como usar o AWS CodePipeline console para criar um pipeline de dois estágios em um passo a passo baseado no. AWS CodePipeline Tutorial: Criar um pipeline de quatro estágios

Publicado em março de 2016

- Simulando AWS CodePipeline tubulações com AWS Lambda

  Saiba como invocar uma função Lambda que permite visualizar as ações e os estágios de CodePipeline um processo de entrega de software à medida que você o projeta, antes que o pipeline esteja operacional. Enquanto cria a estrutura do pipeline, você pode usar a função do Lambda para testar se o seu pipeline será concluído com êxito.

  Publicado em fevereiro de 2016

- Executando AWS Lambda funções no CodePipeline uso AWS CloudFormation

  Saiba como criar uma AWS CloudFormation pilha que provisione todos os AWS recursos usados na tarefa Invocar um AWS Lambda função em um pipeline em CodePipeline do guia do usuário.

  Publicado em fevereiro de 2016

- Provisionando ações personalizadas CodePipeline em AWS CloudFormation

  Saiba como usar AWS CloudFormation para provisionar ações personalizadas em CodePipeline.

  Publicado em janeiro de 2016

- Provisionamento com CodePipeline AWS CloudFormation

  Saiba como provisionar um pipeline básico de entrega contínua CodePipeline usando AWS CloudFormation.

  Publicado em dezembro de 2015

- Implantando de CodePipeline para OpsWorks Usando uma ação personalizada e AWS Lambda

  Saiba como configurar seu pipeline e a AWS Lambda função a ser implantada AWS OpsWorks usando CodePipeline.

  Publicado em julho de 2015

CodePipeline tutoriais

Depois de concluir as etapasComeçando com CodePipeline, você pode experimentar um dos AWS CodePipeline tutoriais neste guia do usuário.

Tópicos

- Tutorial: Implante em instâncias do Amazon EC2 com CodePipeline
- Tutorial: crie e envie uma imagem do Docker para o Amazon ECR com CodePipeline (tipo V2)
- Tutorial: Implante no Amazon EKS com CodePipeline
- Tutorial: Criar um pipeline que executa comandos com computação (tipo V2)
- Tutorial: Usar tags do Git para iniciar o pipeline
- Tutorial: Filtrar nomes de ramificação para solicitações pull a fim de iniciar o pipeline (tipo V2)
- Tutorial: Usar variáveis no nível do pipeline
- Tutorial: Criar um pipeline simples (bucket do S3)
- Tutorial: criar um pipeline simples (CodeCommit repositório)
- Tutorial: Criar um pipeline de quatro estágios
- Tutorial: configurar uma regra de CloudWatch eventos para receber notificações por e-mail sobre mudanças no estado do pipeline
- Tutorial: crie um pipeline que crie e teste seu aplicativo Android com AWS Device Farm
- Tutorial: crie um pipeline que teste seu aplicativo iOS com AWS Device Farm
- Tutorial: Criar um pipeline que realiza a implantação no Service Catalog
- Tutorial: criar um pipeline com AWS CloudFormation
- Tutorial: criar um pipeline que usa variáveis de AWS CloudFormation ações de implantação
- Tutorial: Implantação padrão do Amazon ECS com CodePipeline
- Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR
- Tutorial: Criar um pipeline que implanta uma skill do Amazon Alexa
- Tutorial: Criar um pipeline que usa o Amazon S3 como um provedor de implantação
- Tutorial: Crie um pipeline que publique seu aplicativo sem servidor no AWS Serverless Application Repository
- Tutorial: implantações da função Lambda com CodePipeline
- Tutorial: Usar variáveis com ações de invocação do Lambda

- Tutorial: use uma ação de AWS Step Functions invocação em um pipeline
- Tutorial: Crie um pipeline que use AWS AppConfig como provedor de implantação
- Tutorial: use o clone completo com uma fonte de GitHub pipeline
- Tutorial: use o clone completo com uma fonte de CodeCommit pipeline
- Tutorial: criar um pipeline com AWS CloudFormation StackSets ações de implantação
- Tutorial: criar uma regra de verificação de variável para um pipeline como uma condição de entrada

Tutorial: Implante em instâncias do Amazon EC2 com CodePipeline

Este tutorial ajuda você a criar uma ação de implantação na CodePipeline qual implanta seu código em instâncias que você configurou no Amazon EC2.

> **Note**
>
> Como parte da criação de um pipeline no console, um bucket de artefatos do S3 será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

> **Note**
>
> A ação de implantação EC2 está disponível somente para pipelines do tipo V2.

Pré-requisitos

Para você usar este tutorial para criar seu pipeline de CD, alguns recursos precisam estar em operação. Veja aqui estão os itens que você precisa para começar:

> **Note**
>
> Todos esses recursos devem ser criados na mesma AWS região.

- Um repositório de controle de origem (usado neste tutorial GitHub) onde você adicionará um `script.sh` arquivo de amostra.
- Você deve usar uma função CodePipeline de serviço existente que tenha sido atualizada com as permissões para essa ação. Para atualizar o perfil de serviço, consulte Permissões de política do perfil de serviço para a ação de implantação do EC2.

Assim que você atender a esses pré-requisitos, poderá continuar com o tutorial e criar seu pipeline de CD.

Etapa 1: Criar instâncias do Amazon EC2 Linux

Nesta etapa, você vai criar as instâncias do Amazon EC2 nas quais implantará uma aplicação de amostra. Como parte desse processo, crie uma função de instância no IAM, caso ainda não tenha criado uma função de instância na região onde deseja criar recursos.

Como criar uma função de instância

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel do console, escolha Roles (Funções).
3. Selecione Criar perfil.
4. Em Selecionar tipo de entidade confiável, selecione AWS service (Serviço da AWS). Em Escolha um caso de uso, selecione EC2. Em Select your use case (Selecione seu caso de uso), escolha EC2. Escolha Próximo.
5. Procure e selecione a política chamada **`AmazonSSMManagedEC2InstanceDefaultPolicy`**.
6. Procure e selecione a política chamada **`AmazonSSMManagedInstanceCore`**. Escolha Próximo: tags.
7. Selecione Próximo: revisar. Forneça um nome para a função (por exemplo, **`EC2InstanceRole`**).

> **Note**
>
> Anote o nome da função para a próxima etapa. Escolha essa função ao criar a instância.

> **Note**
>
> Você adicionará permissões a essa função para permitir o acesso ao bucket de artefato do S3 para o pipeline depois da criação do pipeline.

Selecione Criar perfil.

Para executar instâncias

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Na navegação lateral, escolha Instâncias e selecione Executar instâncias na parte superior da página.
3. Em Nome, insira **`MyInstances`**. Isso atribui à instância uma tag Chave de **`Name`** e uma tag Valor de **`MyInstances`**.
4. Em Imagens de aplicativos e sistemas operacionais (Amazon Machine Image), localize a opção Amazon Linux AMI com o AWS logotipo e verifique se ela está selecionada. (Essa AMI é descrita como AMI do Amazon Linux 2 (HVM) e é identificada como “Elegível para o nível gratuito”.)
5. Em Tipo de instância, selecione o tipo `t2.micro` elegível para o nível gratuito como configuração de hardware para a instância.
6. Na seção Par de chaves (login), selecione um par de chaves ou crie um.
7. Em Configurações de rede, verifique se o status é Habilitar.
8. Expanda Advanced details (Detalhes avançados). Em Perfil de instância do IAM, selecione o perfil do IAM criado no procedimento anterior (por exemplo, **`EC2InstanceRole`**).

> **Note**
>
> Não deixe a função da instância em branco, pois isso cria uma função padrão e não seleciona a função criada por você.

9. Em Resumo, em Número de instâncias, insira 2.
10. Escolha Iniciar instância.
11. É possível visualizar o status da ativação na página Instâncias. Ao executar uma instância, seu estado inicial é `pending`. Após o início da instância, seu estado muda para `running` e ela recebe um nome DNS público. (Se a coluna DNS público não for exibida, escolha o Show/Hide ícone e selecione DNS público.)

Etapa 2: adicionar permissões de bucket do artefato à função de instância do EC2

Atualize a função de instância do EC2 criada para a instância a fim de permitir que ela acesse o bucket de artefato do pipeline.

> **Note**
>
> Ao criar a instância, você cria ou usa uma função de instância do EC2 existente. Para evitar `Access Denied` erros, você deve adicionar permissões de bucket do S3 à função de instância para dar à instância permissões para o bucket de CodePipeline artefatos. Crie uma função padrão ou atualize a função existente com a permissão `s3:GetObject` limitada ao bucket de artefato da região do pipeline.

1. Navegue até seu funil no CodePipeline console. Escolha Configurações. Exiba o nome e o local do armazenamento de artefato de um pipeline existente. Anote o nome do recurso da Amazon (ARN) do bucket de artefato e o copie.
2. Navegue até o console do IAM e escolha Roles (Funções). Selecione a função de instância que você criou na etapa 1 deste tutorial.
3. Na guia Permissions (Permissões), escolha Add inline policy (Adicionar política em linha).
4. Adicione o JSON a seguir ao documento da política, substituindo o valor no campo `Resource` pelo ARN do bucket.

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

5. Selecione Atualizar.

Etapa 3: Adicionar um arquivo de script ao repositório

Cole esse texto de amostra a fim de criar o arquivo `script.sh` para a etapa pós-script da implantação.

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

Para adicionar um **`script.sh`** arquivo para seu repositório de origem

1. Abra um editor de texto e, em seguida, copie e cole o arquivo acima em um novo arquivo.
2. Confirme e envie o arquivo `script.sh` para o repositório de origem.

   a. Adicione o arquivo.

   ```
   git add .
   ```

   b. Confirme a alteração.

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

   c. Envie a confirmação.

   ```
   git push
   ```

   Anote o caminho no repositório.

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

Etapa 4: Criação do pipeline

Use o CodePipeline assistente para criar seus estágios de pipeline e conectar seu repositório de origem.

Para criar o pipeline

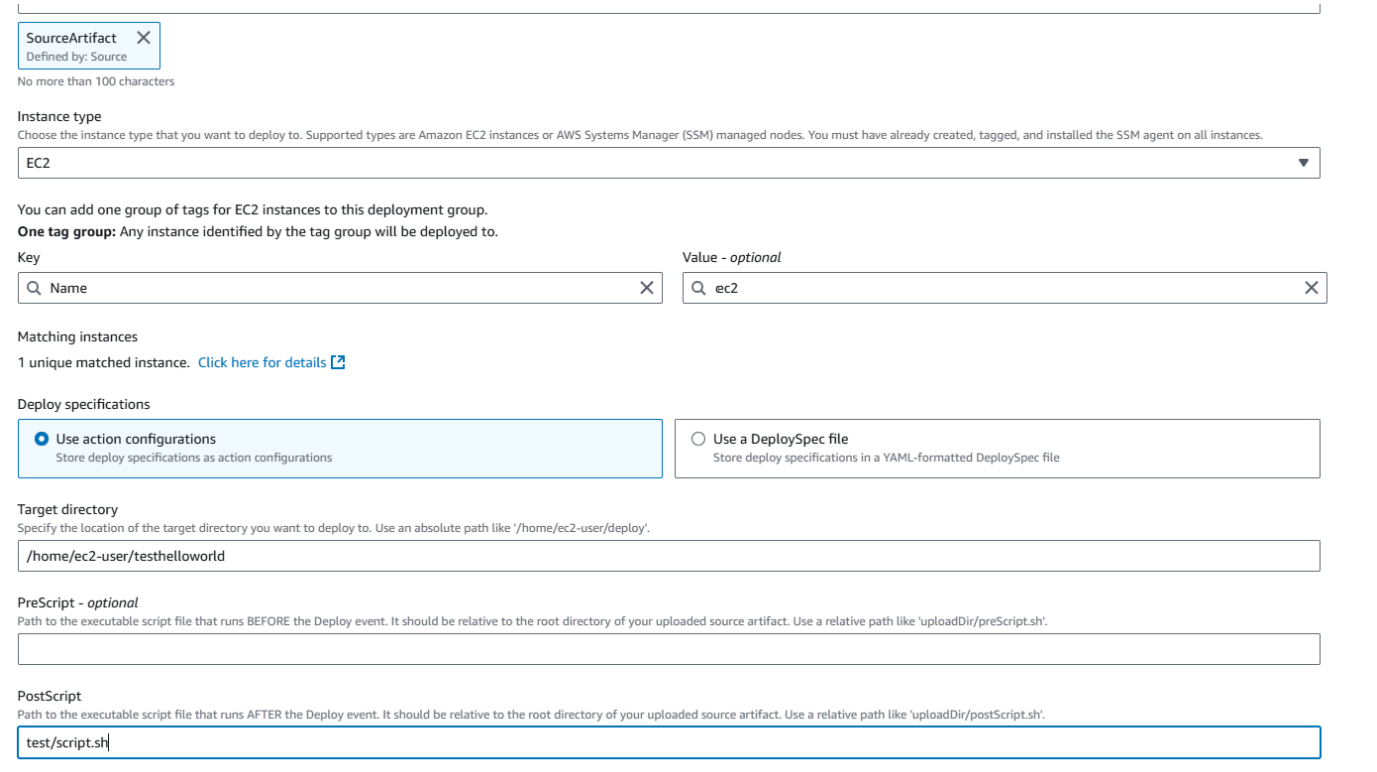
1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyPipeline`**.
5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.
6. Em Função de serviço, escolha Usar função de serviço existente e, em seguida, escolha a função de CodePipeline serviço que foi atualizada com as permissões necessárias para essa ação. Para configurar sua função CodePipeline de serviço para essa ação, consulte Permissões de política do perfil de serviço para a ação de implantação do EC2.
7. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
8. Na página Etapa 3: Adicionar etapa de origem, adicione uma etapa de origem:

   a. Em Provedor de origem, escolha GitHub (via GitHub aplicativo).

   b. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulte GitHub conexões.

   c. Em Repository name (Nome do repositório), selecione o nome do seu repositório do GitHub .

   Escolha Próximo.
9. Na página Etapa 4: Adicionar etapa de compilação, escolha Ignorar.
10. Na página Etapa 5: Adicionar etapa de implantação, escolha EC2.



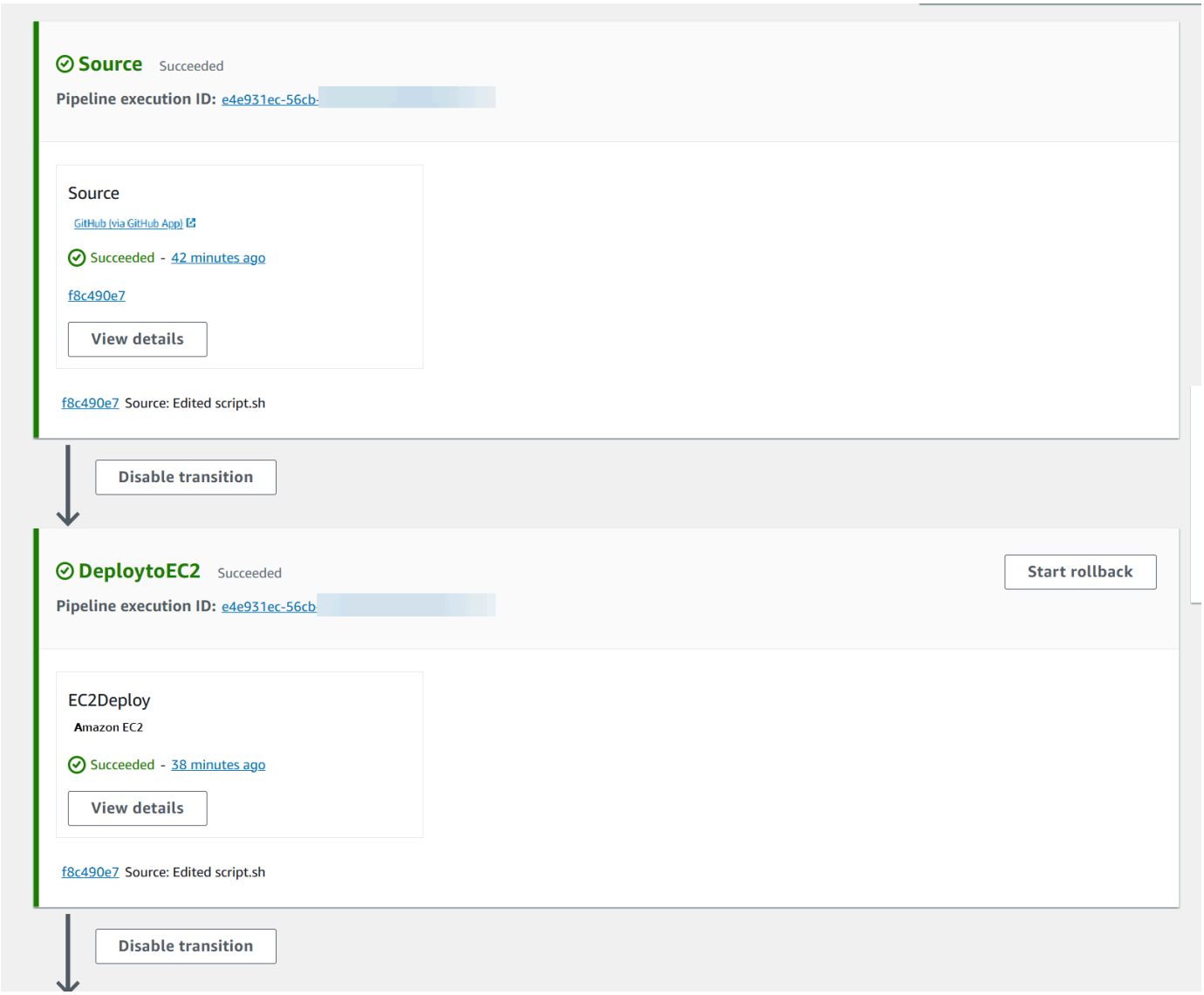
a. Em Diretório de destino, insira o diretório na instância na qual você deseja implantar, como `/home/ec2-user/testhelloworld`.

> **Note**
>
> Especifique o diretório de implantação que você deseja que a ação use na instância. A ação automatizará a criação do diretório especificado na instância como parte da implantação.

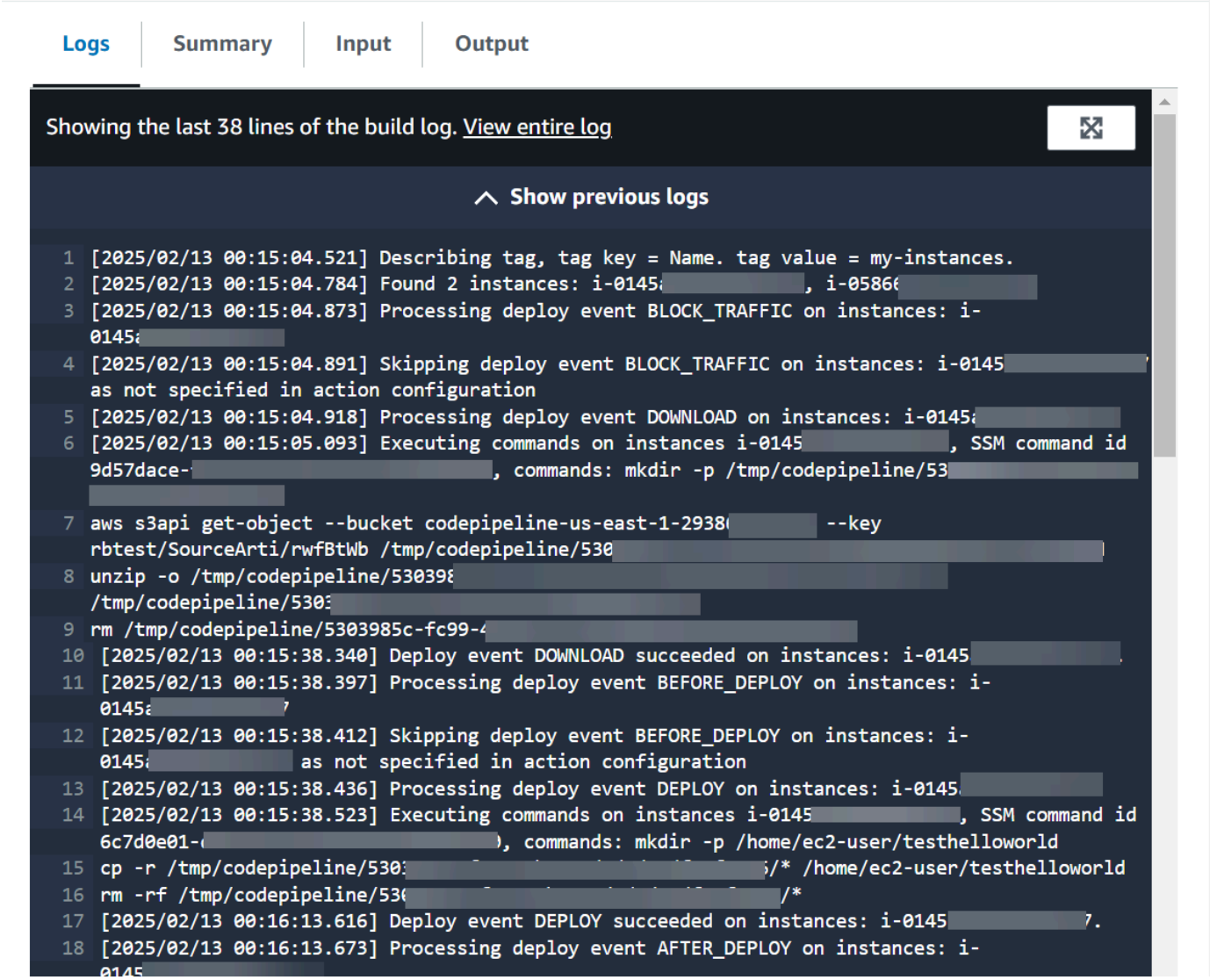
b. Para PostScript, insira o caminho e o nome do arquivo para seu script, como`test/script.sh`.

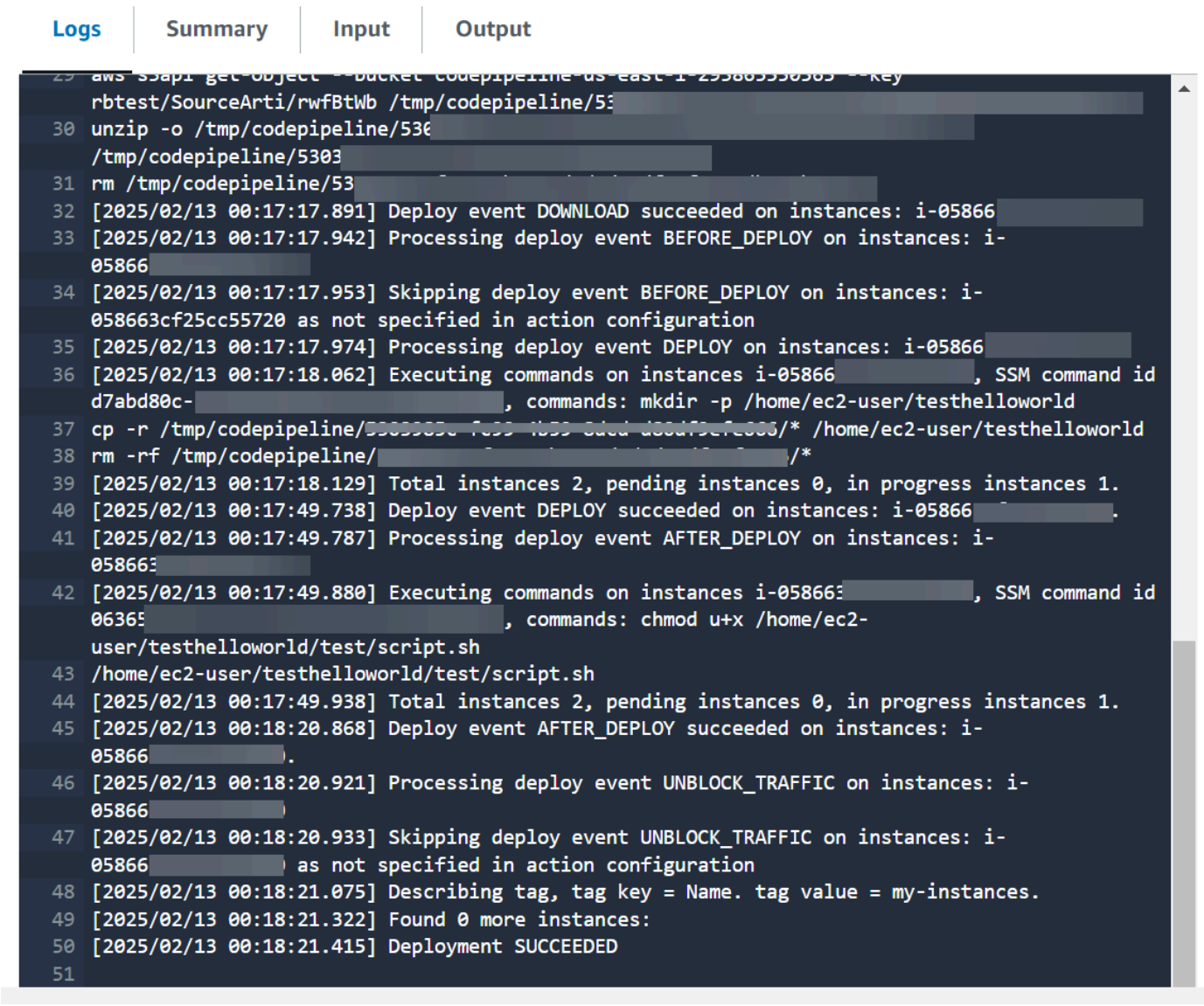
c. Escolha Próximo.

11. Na página Step 6: Review, revise a configuração do pipeline e escolha Create pipeline para criá-lo.



12. Depois que o pipeline for executado com êxito, escolha Visualizar detalhes para exibir os logs na ação a fim de exibir a saída da ação da computação gerenciada.


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



Etapa 5: teste seu pipeline

Seu pipeline deve ter tudo para executar uma implantação AWS contínua nativa de ponta a ponta. Agora, teste a funcionalidade enviando uma alteração de código ao repositório de origem.

Para testar o pipeline

1. Faça uma alteração no código no repositório de origem configurado, confirme e envie a alteração.
2. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
3. Escolha o pipeline na lista.

4. Observe a evolução do pipeline pelos respectivos estágios. O pipeline deve ser concluído, e a ação implanta o script nas instâncias.
5. Para obter mais informações sobre solução de problemas, consulte A ação de implantação do EC2 falha com uma mensagem de erro `Não existe tal arquivo`.

Tutorial: crie e envie uma imagem do Docker para o Amazon ECR com CodePipeline (tipo V2)

Este tutorial ajuda você a criar uma ação de construção CodePipeline que executa e envia sua imagem do Docker para o Amazon ECR após uma alteração no seu código-fonte. Este tutorial também mostra como adicionar uma ação de implantação do Amazon ECS que implanta a imagem enviada por push.

Important

Como parte da criação de um pipeline no console, um bucket de artefatos do S3 será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Note

Este tutorial é sobre a ação de ECRBuildAndPublish criação de um CodePipeline pipeline com um repositório de GitHub origem e uma ação padrão do Amazon ECS para implantação em um cluster do Amazon ECS. Para ver um tutorial que usa um pipeline com um repositório de imagens do ECR como fonte da ação de CodeDeploy blue/green implantação de um Amazon ECS, consulte. CodePipeline Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR

> **Important**
>
> Essa ação usa CodeBuild computação CodePipeline gerenciada para executar comandos em um ambiente de compilação. A execução da ação Comandos incorrerá em cobranças separadas no AWS CodeBuild.

Pré-requisitos

Para você usar este tutorial para criar seu pipeline de CD, alguns recursos precisam estar em operação. Veja aqui estão os itens que você precisa para começar:

> **Note**
>
> Todos esses recursos devem ser criados na mesma AWS região.

- Um repositório de controle de código-fonte (este tutorial usa GitHub) onde você adicionará o seguinte para este tutorial:
  - Na Etapa 1, você adicionará uma amostra do Dockerfile ao seu repositório de origem como artefato de entrada para a ECRBuildAndPublish ação de construção em. CodePipeline
  - Na Etapa 2, você adicionará um exemplo de arquivo imagedefinitions.json ao seu repositório de origem como requisito para a ação de implantação padrão do Amazon ECS em. CodePipeline
- Um repositório de imagens do Amazon ECR que contém uma imagem compilada por você a partir do Dockerfile. Para obter mais informações, consulte Criação de um repositório e Envio de uma imagem por push no Guia do usuário do Amazon Elastic Container Registry.
- Um cluster e um serviço do Amazon ECS criados na mesma região do repositório de imagens. Para obter mais informações, consulte Criação de um cluster e Criação de um serviço no Guia do desenvolvedor do Amazon Elastic Container Service.

Assim que você atender a esses pré-requisitos, poderá continuar com o tutorial e criar seu pipeline de CD.

Etapa 1: Adicionar um Dockerfile ao repositório de origem

Este tutorial usa a ECRBuildAndPublish ação para criar sua imagem do Docker e enviá-la para o Amazon ECR. A ação de computação gerenciada em é CodePipeline usada CodeBuild para

executar os comandos de login do ECR e envio de imagem. Você não precisa adicionar um `buildspec.yml` arquivo ao seu repositório de código-fonte para saber CodeBuild como fazer isso. Você só fornece o Dockerfile no repositório da mesma forma que faz para este exemplo.

Cole esse texto de amostra para criar o arquivo `Dockerfile`. Este Dockerfile de amostra é o mesmo da amostra usada nas instruções de imagem do ECR nos pré-requisitos.

```
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 adicionar um **`Dockerfile`** arquivo para seu repositório de origem

1. Abra um editor de texto e, em seguida, copie e cole o Dockerfile acima em um novo arquivo.
2. Confirme e envie o arquivo `Dockerfile` para o repositório de origem.

   a. Adicione o arquivo.

   ```
   git add .
   ```

   b. Confirme a alteração.

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

   c. Envie a confirmação.

```
git push
```

Não se esqueça de colocar o arquivo no nível raiz do repositório.

```
/ Dockerfile
```

Etapa 2: Adicionar um arquivo imagedefinitions.json ao repositório de origem

Este tutorial usa a ação de implantação padrão do Amazon ECS para CodePipeline implantar seu contêiner em seu cluster do Amazon ECS. A ação de implantação padrão do Amazon ECS exige um arquivo imagedefinitions.json contendo o nome e o URI da imagem. Para obter mais informações sobre o arquivo imagedefinitions.json, consulte Arquivo imagedefinitions.json para ações de implantação padrão do Amazon ECS.

Cole esse texto de amostra para criar o arquivo `imagedefinitions.json`. Use o nome no Dockerfile, como `hello-world`, e use o URI do repositório do Amazon ECR onde a imagem está armazenada.

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

Para adicionar um **`definições de imagem.json`** arquivo para seu repositório de origem

1. Abra um editor de texto e, em seguida, copie e cole o exemplo acima em um novo arquivo.
2. Confirme e envie o arquivo `imagedefinitions.json` para o repositório de origem.

   a. Adicione o arquivo.

   ```
   git add .
   ```

   b. Confirme a alteração.

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

c. Envie a confirmação.

```
git push
```

Não se esqueça de colocar o arquivo no nível raiz do repositório.

```
/ imagedefinitions.json
```

Etapa 3: Criação do pipeline

Use o CodePipeline assistente para criar seus estágios de pipeline e conectar seu repositório de origem.

Para criar o pipeline

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **MyPipeline**.
5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.
6. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.
7. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
8. Na página Etapa 3: Adicionar etapa de origem, adicione uma etapa de origem:

   a. Em Provedor de origem, escolha GitHub (via GitHub aplicativo).

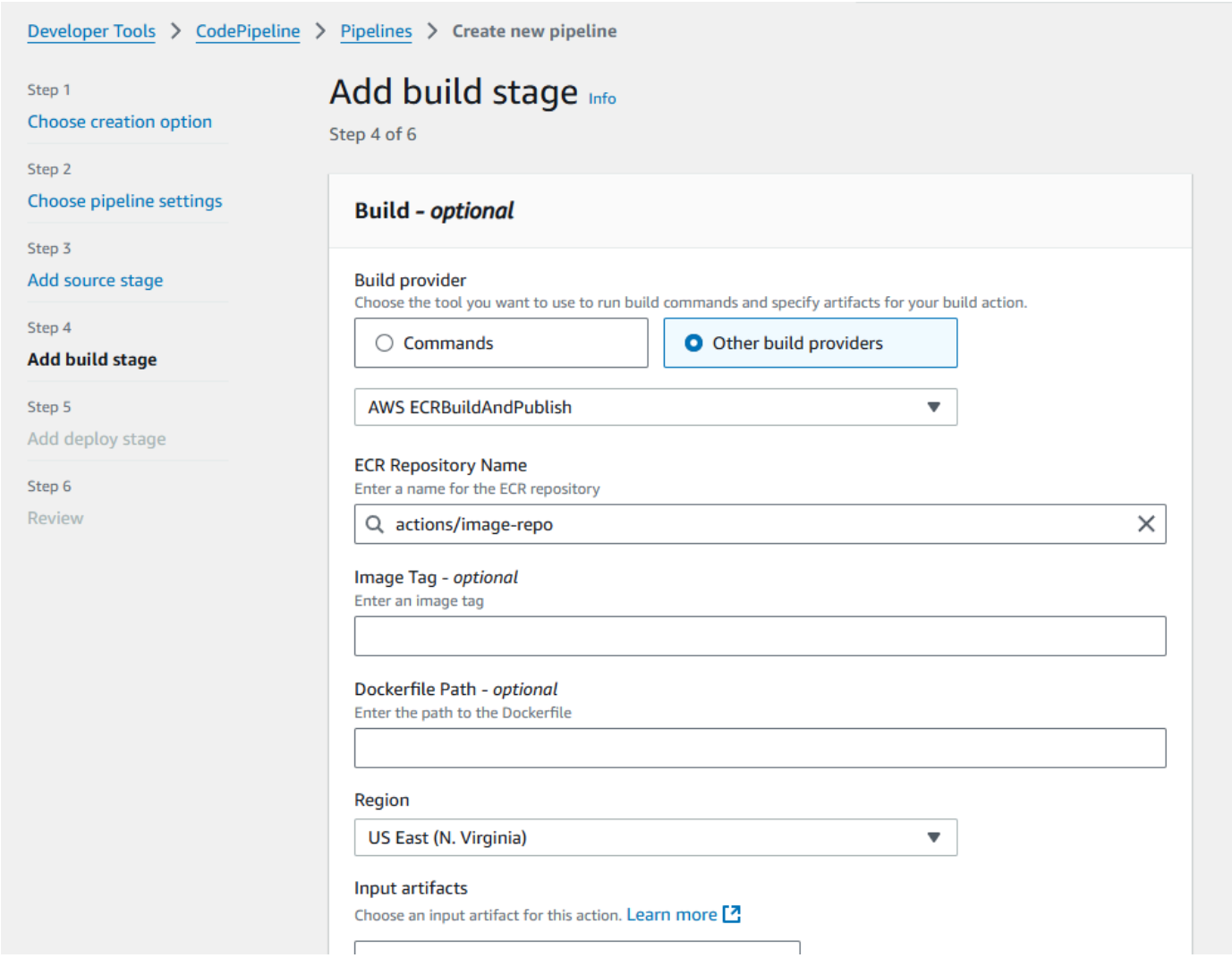
b. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulteGitHub conexões.

c. Em Repository name (Nome do repositório), selecione o nome do seu repositório do GitHub .

d. Em Ramificação padrão, selecione a ramificação que você deseja especificar quando o pipeline é iniciado manualmente ou com um evento de origem que não seja uma tag do Git. Se a origem da alteração não for o gatilho ou se a execução de um pipeline tiver sido iniciada manualmente, a alteração usada será a confirmação HEAD da ramificação padrão.

Escolha Próximo.

9. Na página Etapa 4: Adicionar estágio de construção, escolha Outros provedores de compilação e, em seguida, escolha ECRBuildAndPublish.



a. Para Nome do repositório do ECR, escolha o repositório de imagens.

b. Escolha Próximo.

10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

11. Na página Etapa 6: Adicionar etapa de implantação, escolha Ignorar etapa de implantação. Você adicionará a ação do ECS na etapa a seguir.

12. Na página Etapa 7: Revisão, revise a configuração do pipeline e escolha Criar pipeline para criar o pipeline.

13. Edite o pipeline para adicionar a ação de implantação do Amazon ECS ao pipeline:

    a. No canto superior direito, escolha Edit (Editar).

    b. Na parte inferior do diagrama, escolha + Add stage (+ Adicionar estágio). Em Nome do estágio, insira um nome, como **`Deploy`**.

    c. Escolha + Add action group (Adicionar grupo de ação).

    d. Em Nome da ação, insira um nome.

    e. Em Provedor de ação, escolha Amazon ECR. Permita que Region (Região) seja definida para a região do pipeline.

    f. Em Artefatos de entrada, escolha o artefato do estágio de origem, como `SourceArtifact`.

    g. Em Nome do cluster, escolha o cluster do Amazon ECS no qual o serviço está em execução.

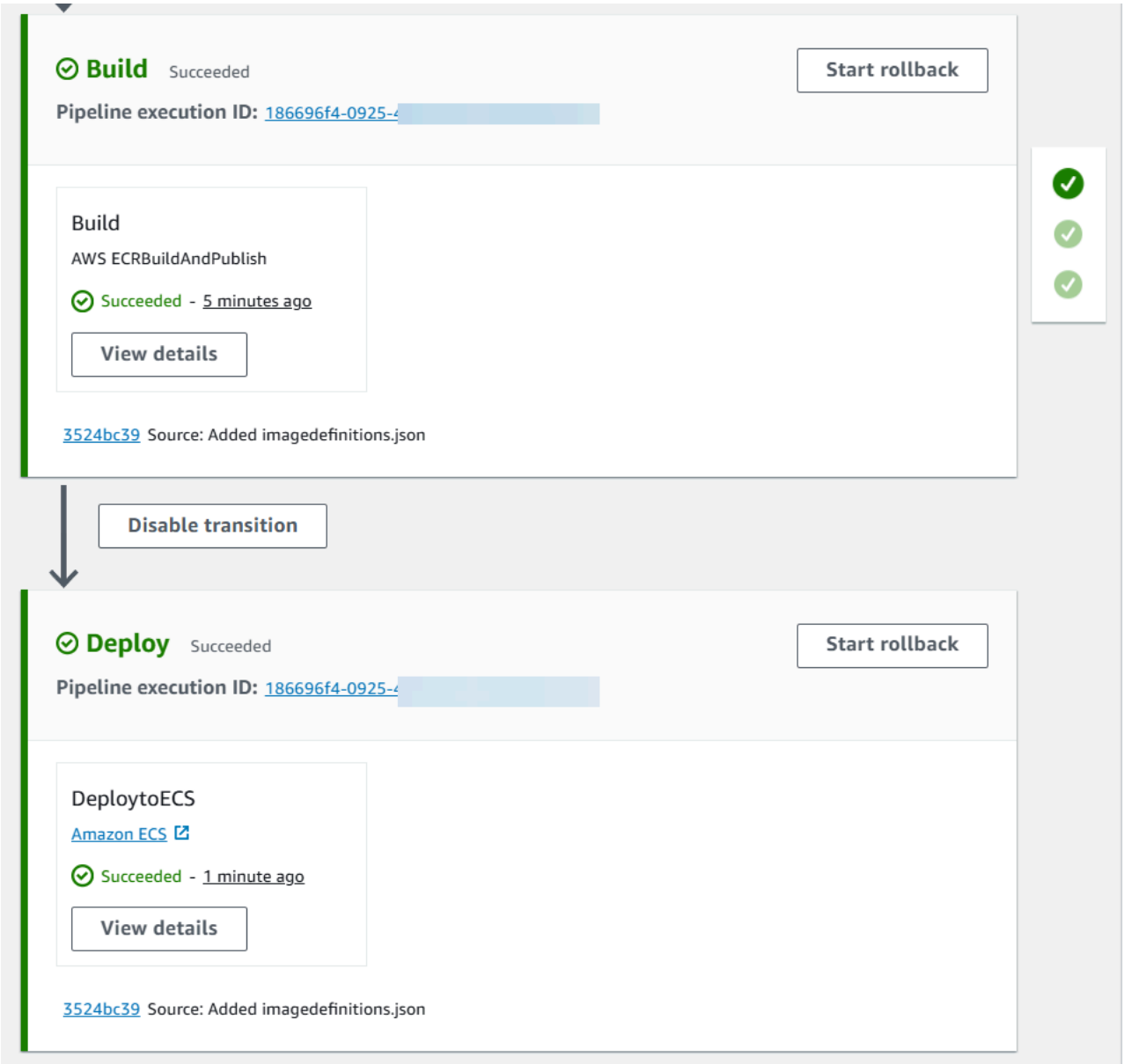
    h. Para Nome do serviço, escolha o serviço a ser atualizado.

    i. Escolha Salvar.

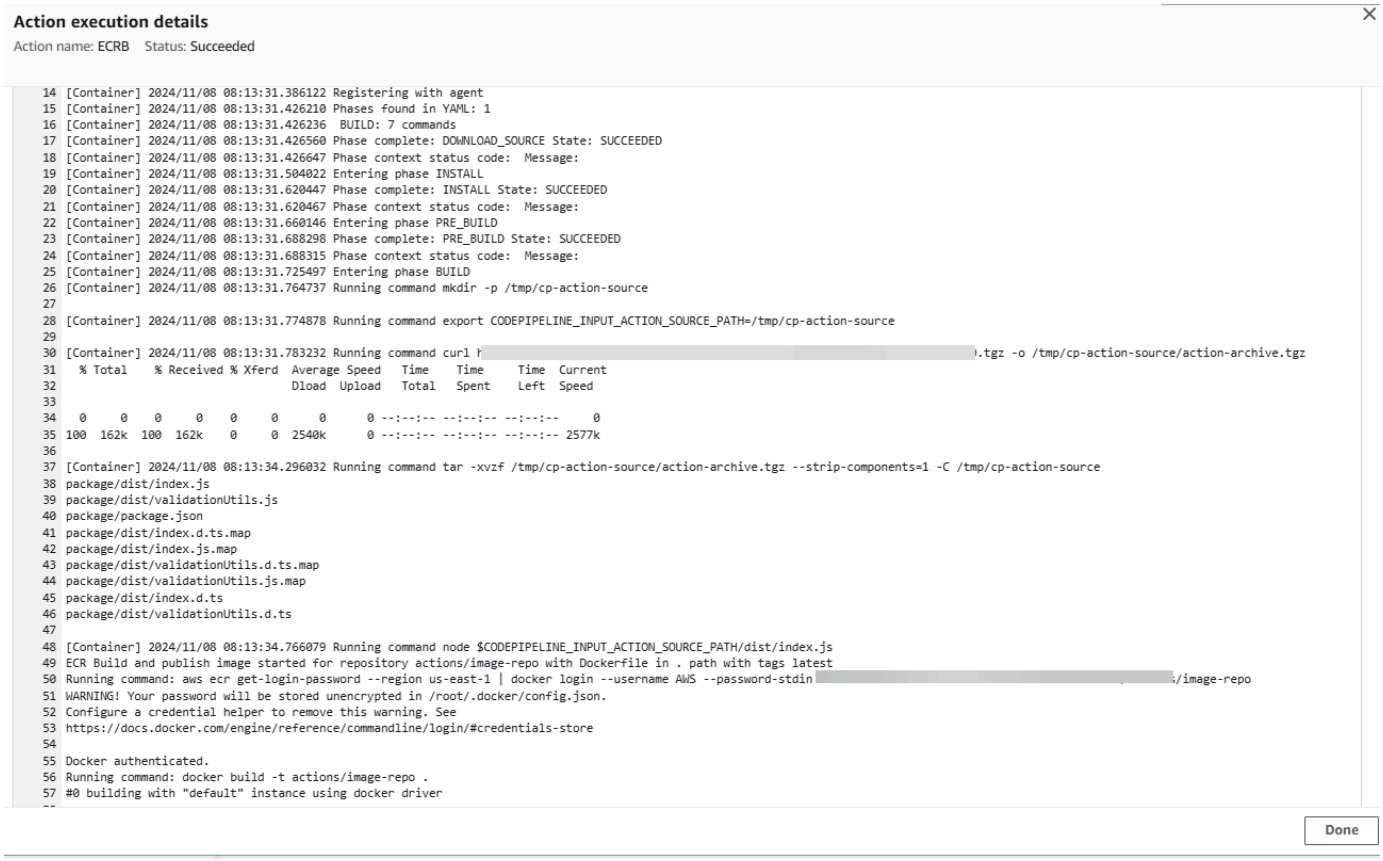
    j. No estágio que está editando, escolha Done (Concluído). No painel do AWS CodePipeline , escolha Save (Salvar) e selecione Save (Salvar) na mensagem de aviso.

    k. Para enviar suas alterações e iniciar uma compilação do pipeline, selecione Liberar alteração e, depois, Liberar.

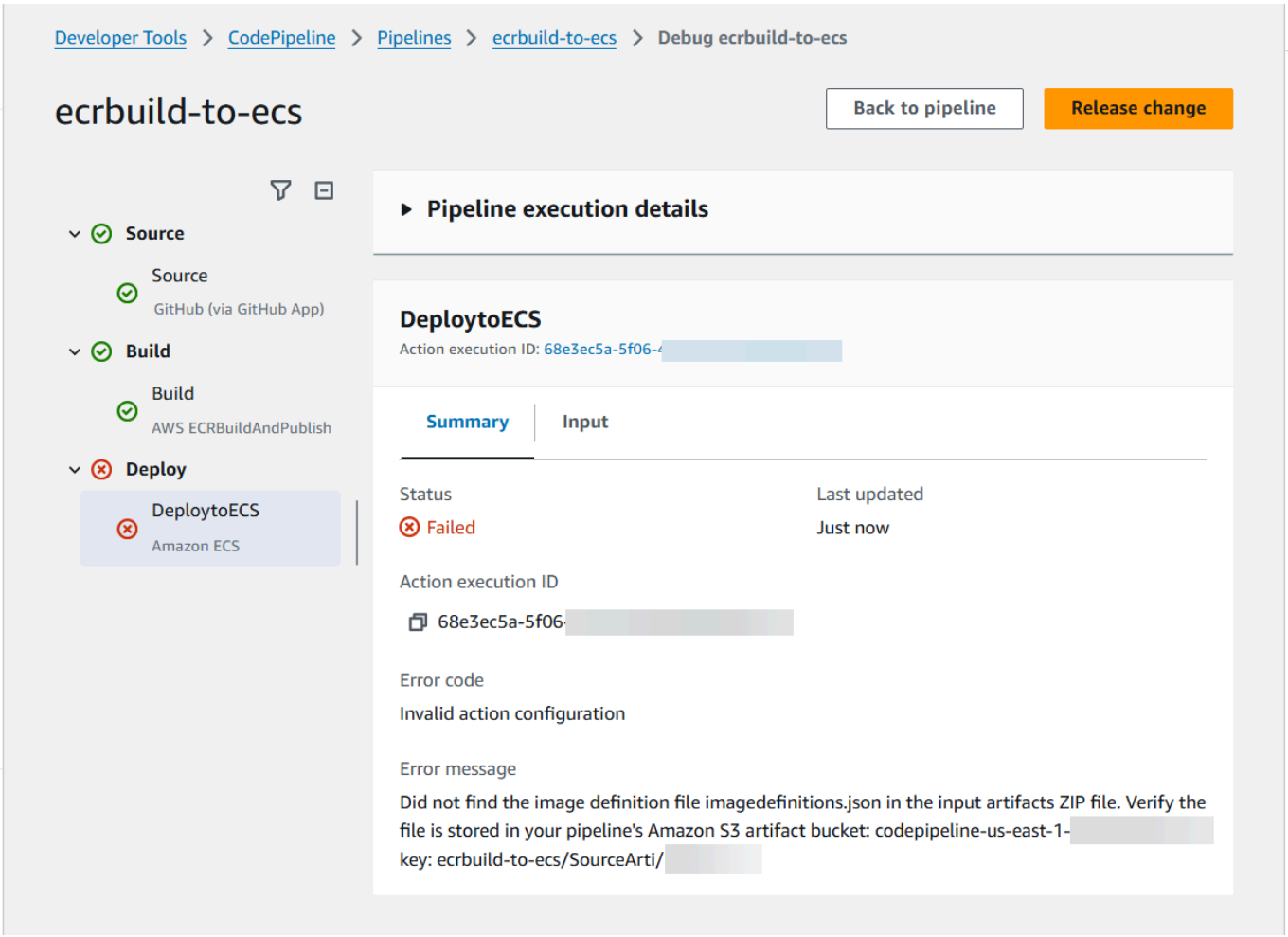
14. Depois que o pipeline for executado, exiba a estrutura e o status do pipeline.



15. Depois que o pipeline for executado com êxito, escolha Visualizar detalhes para exibir os logs na ação a fim de exibir a saída da ação da computação gerenciada.



16. Solucione o problema de qualquer falha. Por exemplo, a ação de implantação do ECS poderá falhar se o arquivo imagedefinitions.json não estiver no repositório de origem. Este é um exemplo da mensagem de erro exibida quando o arquivo imagedefinitions.json não é encontrado.



Etapa 4: Testar o pipeline

Seu pipeline deve ter tudo para executar uma implantação AWS contínua nativa de ponta a ponta. Agora, teste a funcionalidade enviando uma alteração de código ao repositório de origem.

Para testar o pipeline

1. Faça uma alteração no código no repositório de origem configurado, confirme e envie a alteração.
2. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
3. Escolha o pipeline na lista.
4. Observe a evolução do pipeline pelos respectivos estágios. O pipeline deve ser concluído e a ação, enviar por push a imagem do Docker que foi criada a partir da alteração do código.

Tutorial: Implante no Amazon EKS com CodePipeline

Este tutorial ajuda você a criar uma ação de implantação na CodePipeline qual implanta seu código em um cluster que você configurou no Amazon EKS.

A ação do EKS oferece suporte a clusters do EKS públicos e privados. Os clusters privados são do tipo recomendado pelo EKS. No entanto, ambos os tipos são compatíveis.

> **Note**
>
> Como parte da criação de um pipeline no console, um bucket de artefatos do S3 será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

> **Note**
>
> Essa ação usa CodeBuild computação CodePipeline gerenciada para executar comandos em um ambiente de compilação. A execução da ação Comandos incorrerá em cobranças separadas no AWS CodeBuild.

> **Note**
>
> A ação de implantação EKS está disponível somente para pipelines do tipo V2.

Pré-requisitos

Para você usar este tutorial para criar seu pipeline de CD, alguns recursos precisam estar em operação. Veja aqui estão os itens que você precisa para começar:

> **Note**
>
> Todos esses recursos devem ser criados na mesma AWS região.

- Um repositório de controle de origem (usado neste tutorial GitHub) onde você adicionará um `deployment.yaml` arquivo de amostra.
- Você deve usar uma função CodePipeline de serviço existente que você atualizará com as permissões para essa ação usando Etapa 3: atualizar a política da função de CodePipeline serviço no IAM abaixo. As permissões necessárias se baseiam no tipo de cluster que você criou. Para obter mais informações, consulte Permissões de política de perfil de serviço.
- Uma imagem de trabalho e uma tag de repositório que você enviou por push para o ECR ou o repositório de imagens.

Assim que você atender a esses pré-requisitos, poderá continuar com o tutorial e criar seu pipeline de CD.

Etapa 1: (Opcional) Criar um cluster no Amazon EKS

Você pode optar por criar um cluster do EKS com um endpoint público ou privado.

Nas etapas a seguir, você criará um cluster público ou privado no EKS. Esta etapa será opcional se você já tiver criado o cluster.

Criar um cluster público no Amazon EKS

Nesta etapa, você cria um cluster no EKS.

Criar um cluster público

1. Abra o console do EKS e escolha Criar cluster.
2. Em Nome, nomeie o cluster. Escolha Próximo.
3. Escolha Criar.

Criar um cluster privado no Amazon EKS

Caso você opte por criar um cluster com um endpoint privado, conecte somente as sub-redes privadas e certifique-se de que elas tenham uma conexão com a Internet.

Siga as próximas cinco sub-etapas para criar um cluster com um endpoint privado.

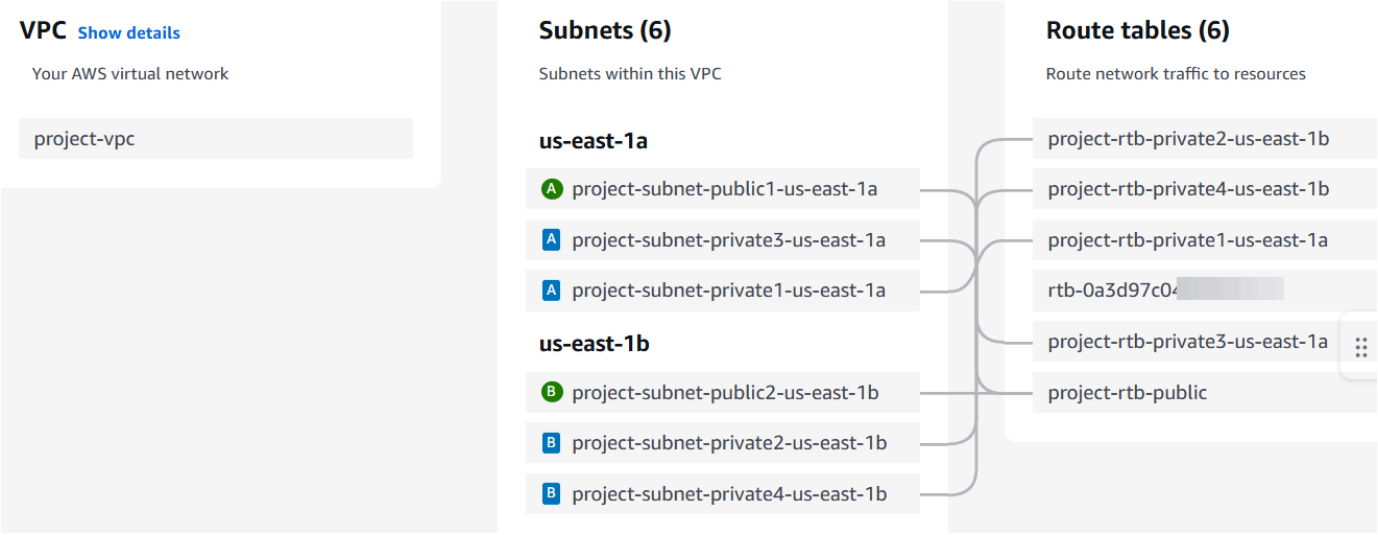
Criar uma VPC no console

1. Abra o console da VPC e escolha Criar VPC.

2. Em Configurações da VPC, escolha VPC e mais.
3. Opte por criar uma sub-rede pública e 4 privadas. Escolha Criar VPC.
4. Na página de sub-redes, escolha Privada.

Determinar as sub-redes privadas na VPC

1. Navegue até a VPC e escolha a ID da VPC para abrir a página de detalhes da VPC.
2. Na página Detalhes da VPC, escolha a guia Mapa de recursos.
3. Consulte o diagrama e anote as sub-redes privadas. As sub-redes são exibidas com rótulos para indicar o status público ou privado, e cada sub-rede é mapeada para uma tabela de rotas.



   Um cluster privado terá todas as sub-redes privadas.
4. Crie uma sub-rede pública para hospedar o gateway NAT. Só é possível anexar um gateway da Internet a uma VPC de cada vez.

Criar um gateway NAT em uma sub-rede pública

1. Na sub-rede pública, crie um gateway NAT. Navegue até o console da VPC e escolha Gateways da Internet. Escolha Criar gateway da Internet.
2. Em Nome, digite um nome para o gateway da internet. Escolha Criar gateway da Internet.

Atualize a tabela de rotas da sub-rede privada a fim de direcionar o tráfego para o gateway NAT.

Adicionar o gateway NAT às tabelas de rotas das sub-redes privadas

1. Navegue até o console da VPC e escolha Sub-redes.
2. Para cada sub-rede privada, escolha-a e, em seguida, escolha a tabela de rotas para essa sub-rede na página de detalhes. Escolha Editar tabela de rotas.
3. Atualize a tabela de rotas da sub-rede privada a fim de direcionar o tráfego de internet para o gateway NAT. Selecione Adicionar rota. Escolha o Gateway NAT dentre as opções a serem adicionadas. Escolha o gateway da internet criado por você.
4. Para a sub-rede pública, crie a tabela de rotas personalizada. Verifique se a lista de controle de acesso (ACL) da rede da sub-rede pública permite o tráfego de entrada vindo da sub-rede privada.
5. Escolha Salvar alterações.

Nesta etapa, você cria um cluster no EKS.

Criar um cluster privado

1. Abra o console do EKS e escolha Criar cluster.
2. Em Nome, nomeie o cluster. Escolha Próximo.
3. Especifique a VPC e as outras informações de configuração. Escolha Criar.

O cluster do EKS pode ser um cluster público ou privado. Esta etapa é para clusters que tenham APENAS um endpoint privado. Verifique se o cluster é privado.

Etapa 2: Configurar o cluster privado no Amazon EKS

Esta etapa será aplicável somente se você tiver criado um cluster privado. Esta etapa é para clusters que tenham APENAS um endpoint privado.

Configure seu cluster

1. Só anexe sub-redes privadas no cluster do EKS na guia Redes. Anexe as sub-redes privadas capturadas na seção Determinar as sub-redes privadas na VPC em Etapa 1: (Opcional) Criar um cluster no Amazon EKS.
2. Certifique-se de que as sub-redes privadas tenham acesso à Internet, pois CodePipeline armazenam e recuperam artefatos do bucket de artefatos do S3 para seu pipeline.

Etapa 3: atualizar a política da função de CodePipeline serviço no IAM

Nesta etapa, você atualizará uma função de CodePipeline serviço existente, como, por exemplo**`cp-service-role`**, com as permissões exigidas CodePipeline para se conectar ao seu cluster. Se você não tiver uma função existente, crie uma nova.

Atualize sua função CodePipeline de serviço com as etapas a seguir.

Para atualizar sua política CodePipeline de função de serviço

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel do console, escolha Roles (Funções).
3. Procure sua função CodePipeline de serviço, como**`cp-service-role`**.
4. Adicione uma nova política em linha.
5. No Editor de políticas, insira a opção a seguir.

   - Para um cluster público, adicione as permissões a seguir.

     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 um cluster privado, adicione as permissões a seguir. Os clusters privados exigirão permissões adicionais para a VPC, se aplicável.

  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. Escolha Atualizar política.

Etapa 4: criar uma entrada de acesso para a função CodePipeline de serviço

Nesta etapa, você cria uma entrada de acesso no cluster que adicionará a função de CodePipeline serviço que você atualizou na Etapa 3, junto com uma política de acesso gerenciado.

1. Abra o console do EKS e navegue até o cluster.
2. Escolha a guia Acesso.
3. Em Entradas de acesso do IAM, escolha Criar entrada de acesso.
4. No ARN da entidade principal do IAM, insira a função recém-atualizada por você para a ação, como **`cp-service-role`**. Escolha Próximo.
5. Na página Etapa 2: Adicionar política de acesso, em Nome da política, escolha a política gerenciada para acesso, como `AmazonEKSClusterAdminPolicy`. Escolha Add policy. Escolha Próximo.

   > **Note**
   >
   > Essa é a política que a CodePipeline ação usa para conversar com o Kubernetes. Como prática recomendada, para diminuir o escopo das permissões na política com o privilégio mínimo em vez da política administrativa, anexe uma política personalizada.

6. Na página de revisão, escolha Criar.

Etapa 5: Crie um repositório de origem e adicione os arquivos de configuração do gráfico Helm

Nesta etapa, você cria um arquivo de configuração que seja indicado para a ação (arquivos de manifesto do Kubernetes ou chart do Helm) e armazena o arquivo de configuração no repositório de origem. Use o arquivo indicado para a configuração. Para acessar mais informações, consulte https://kubernetes.io/docs/reference/kubectl/quick-reference/ ou https://helm.sh/docs/topics/charts/.

- Para Kubernetes, use um arquivo de manifesto.
- Para Helm, use um chart do Helm.

1. Crie ou use um GitHub repositório existente.
2. Crie uma nova estrutura em seu repositório para seus arquivos de gráfico do Helm, conforme mostrado no exemplo abaixo.

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

3. Adicione o arquivo ao nível raiz do repositório.

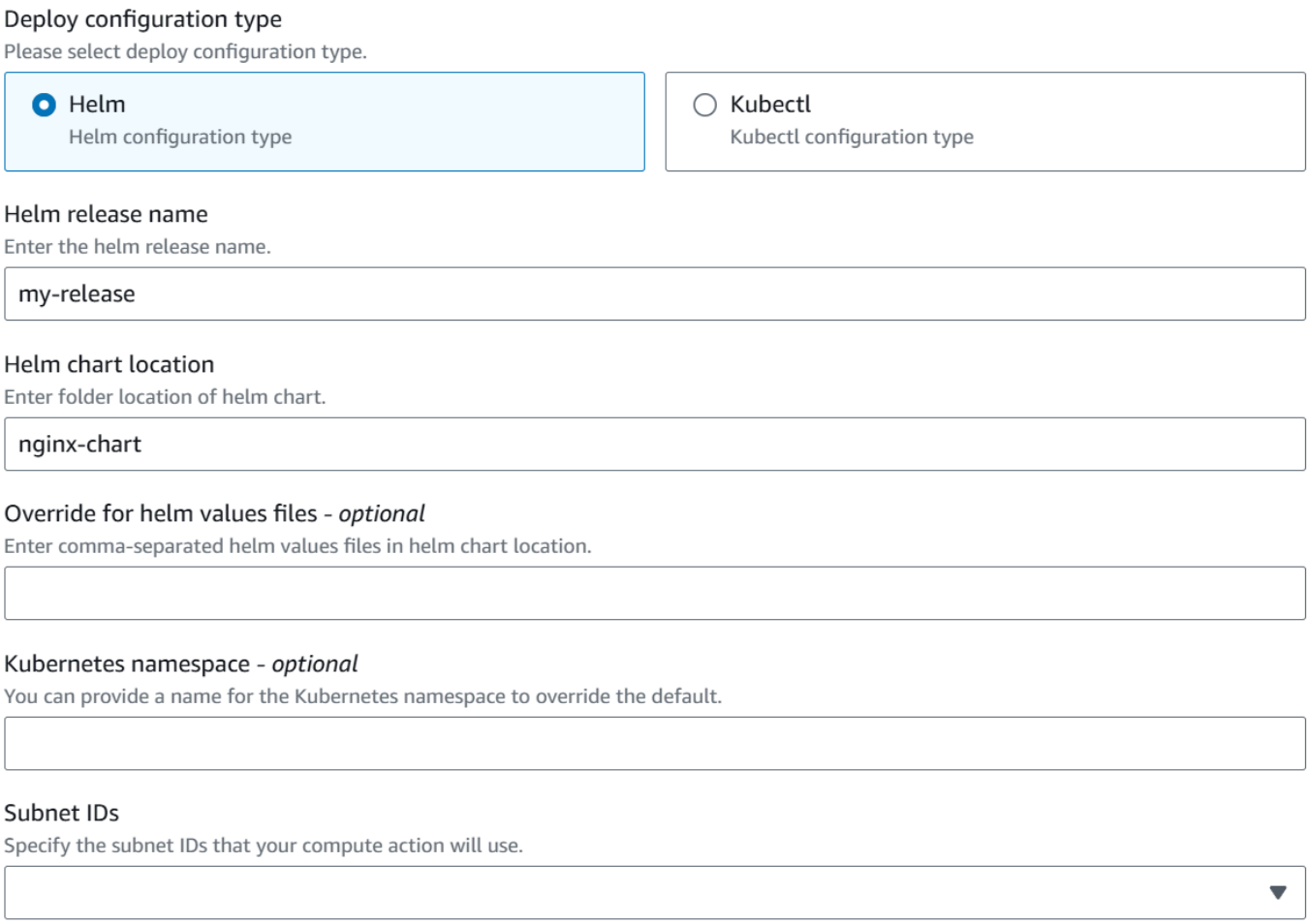
Etapa 6: Criação do pipeline

Use o CodePipeline assistente para criar seus estágios de pipeline e conectar seu repositório de origem.

Para criar o pipeline

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).

3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyEKSPipeline`**.
5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.
6. Em Perfil de serviço, escolha o perfil de serviço atualizado por você na etapa 3.
7. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
8. Na página Etapa 3: Adicionar estágio de origem, para Provedor de origem, escolha criar uma conexão com seu GitHub repositório.
9. Na página Etapa 4: Adicionar etapa de compilação, escolha Ignorar.
10. Na página Etapa 5: Adicionar etapa de implantação, escolha Amazon EKS.

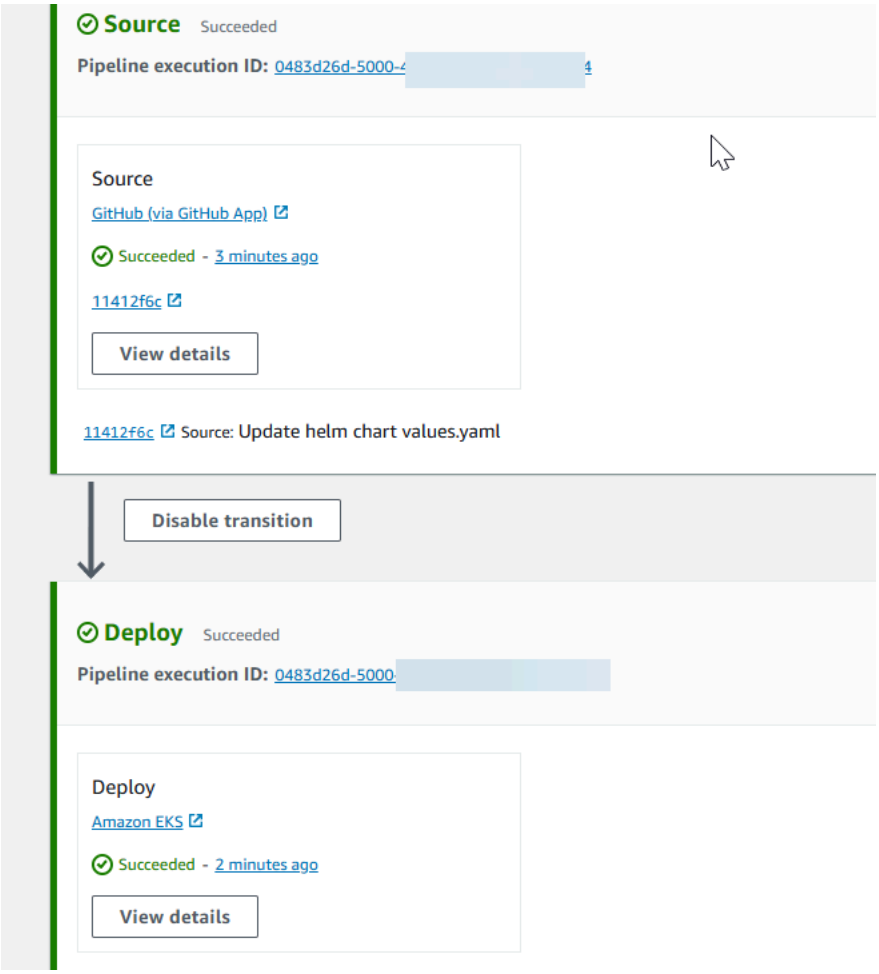


a. Em Tipo de configuração da implantação, escolha Helm.

b. Em Local de chart do Helm, insira o nome da versão, como `my-release`. Para a localização do gráfico do Helm, insira o caminho para os arquivos do gráfico do Helm, como. `mychart`

c. Escolha Próximo.

11. Na página Step 6: Review, revise a configuração do pipeline e escolha Create pipeline para criá-lo.



12. Depois que o pipeline for executado com êxito, escolha Visualizar detalhes para exibir os logs na ação a fim de exibir a saída da ação.

Tutorial: Criar um pipeline que executa comandos com computação (tipo V2)

Neste tutorial, você vai configurar um pipeline que realiza a execução contínua de comandos de compilação fornecidos usando a ação Commands em um estágio de compilação. Para ter mais informações sobre a ação Commands, consulte Referência da ação Commands.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Pré-requisitos

Você já deve ter o seguinte:

- Um GitHub repositório. Você pode usar o GitHub repositório no Tutorial: use o clone completo com uma fonte de GitHub pipeline qual você criou.

Etapa 1: Crie arquivos de origem e envie para seu GitHub repositório

Nesta seção, você cria e envia seus arquivos de origem de exemplo para o repositório que o pipeline usa para seu estágio de origem. Neste exemplo, você produz e envia o seguinte:

- Um arquivo `README.txt`.

Como criar arquivos de origem

1. Crie um arquivo com o seguinte texto:

   ```
   Sample readme file
   ```

2. Salve o arquivo como `README.txt`.

Para enviar arquivos para o seu GitHub repositório

1. Enviar ou carregar os arquivos ao repositório do . Esses arquivos são o artefato de origem gerado pelo assistente Create Pipeline (Criar pipeline) para a ação de implantação no AWS CodePipeline. Os arquivos devem ter a seguinte aparência em seu diretório local:

   ```
   README.txt
   ```

2. Para usar a linha de comando Git a partir de um repositório clonado no computador local:

   a. Execute o seguinte comando para organizar todos os seus arquivos de uma só vez:

      ```
      git add -A
      ```

   b. Execute o seguinte comando para confirmar os arquivos com uma mensagem de confirmação.

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

   c. Execute o seguinte comando para enviar os arquivos de seu repositório local para seu repositório do :

      ```
      git push
      ```

Etapa 2: Criar o pipeline

Nesta seção, você criará um pipeline com as seguintes ações:

- Um estágio de origem com uma ação GitHub (via GitHub aplicativo) para o repositório em que os arquivos de origem são armazenados.
- Um estágio de compilação com a ação Commands.

Criar um pipeline com o assistente

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).

3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyCommandsPipeline`**.
5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.
6. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.

   > **Note**
   >
   > Caso utilize um perfil de serviço já existente, será preciso incluir as permissões a seguir ao perfil de serviço para habilitar o uso da ação Commands. Restrinja as permissões ao nível de recurso do pipeline aplicando permissões baseadas em recurso na declaração de política do perfil de serviço. Para ter mais informações, consulte o exemplo de política em Permissões de política de perfil de serviço.
   >
   > - registros: CreateLogGroup
   > - registros: CreateLogStream
   > - registros: PutLogEvents

7. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
8. Na página Etapa 3: Adicionar etapa de origem, adicione uma etapa de origem:
   a. Em Provedor de origem, escolha GitHub (via GitHub aplicativo).
   b. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulteGitHub conexões.
   c. Em Repository name (Nome do repositório), selecione o nome do seu repositório do GitHub.com .
   d. Em Ramificação padrão, selecione a ramificação que você deseja especificar quando o pipeline é iniciado manualmente ou com um evento de origem que não seja uma tag do Git. Se a origem da alteração não for o gatilho ou se a execução de um pipeline tiver sido iniciada manualmente, a alteração usada será a confirmação HEAD da ramificação padrão.

Você também pode especificar webhooks com filtragem (acionadores). Para obter mais informações, consulte Automatizar a inicialização de pipelines usando gatilhos e filtragem.

Escolha Próximo.

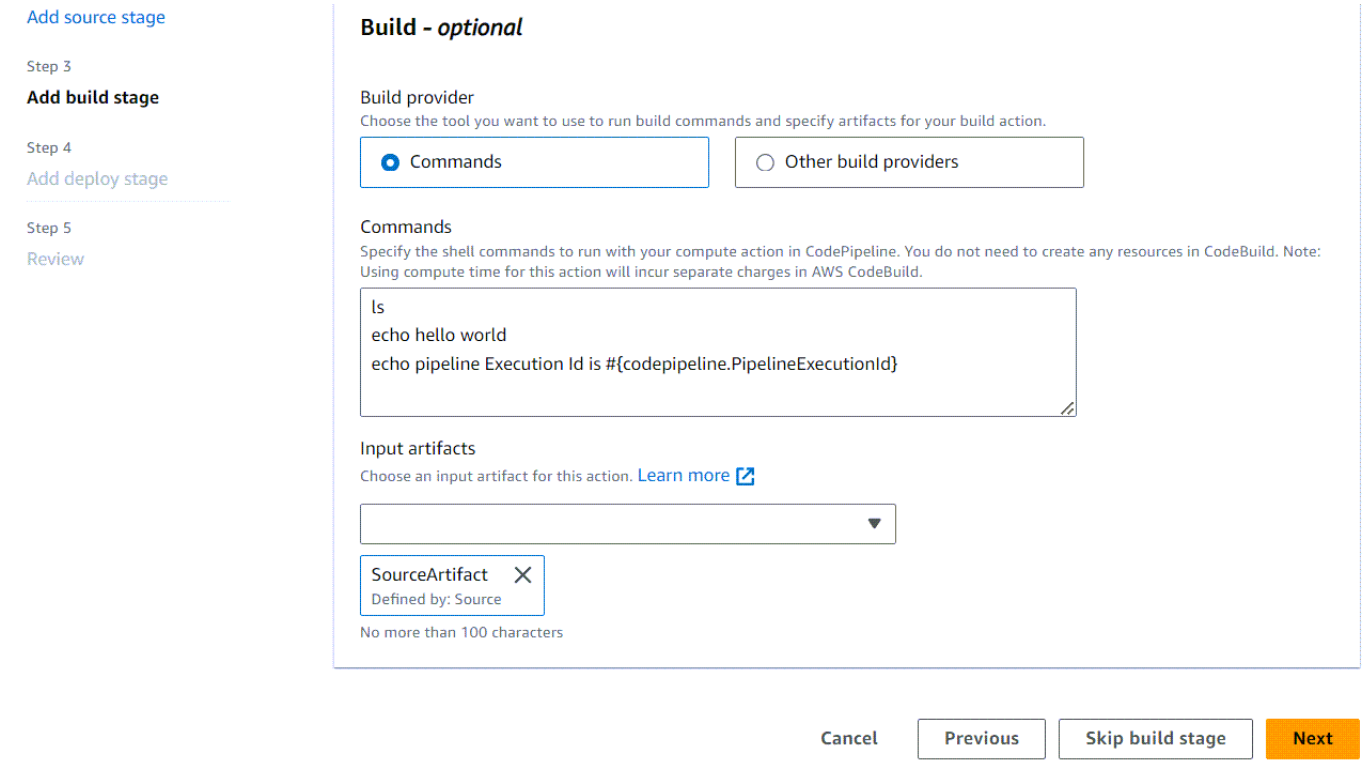
9. Em Etapa 4: adicionar estágio de compilação, escolha Comandos.

> **Note**
>
> A execução da ação Commands acarretará cobranças separadas em AWS CodeBuild.

Insira os comandos a seguir:

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

Escolha Próximo.



10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

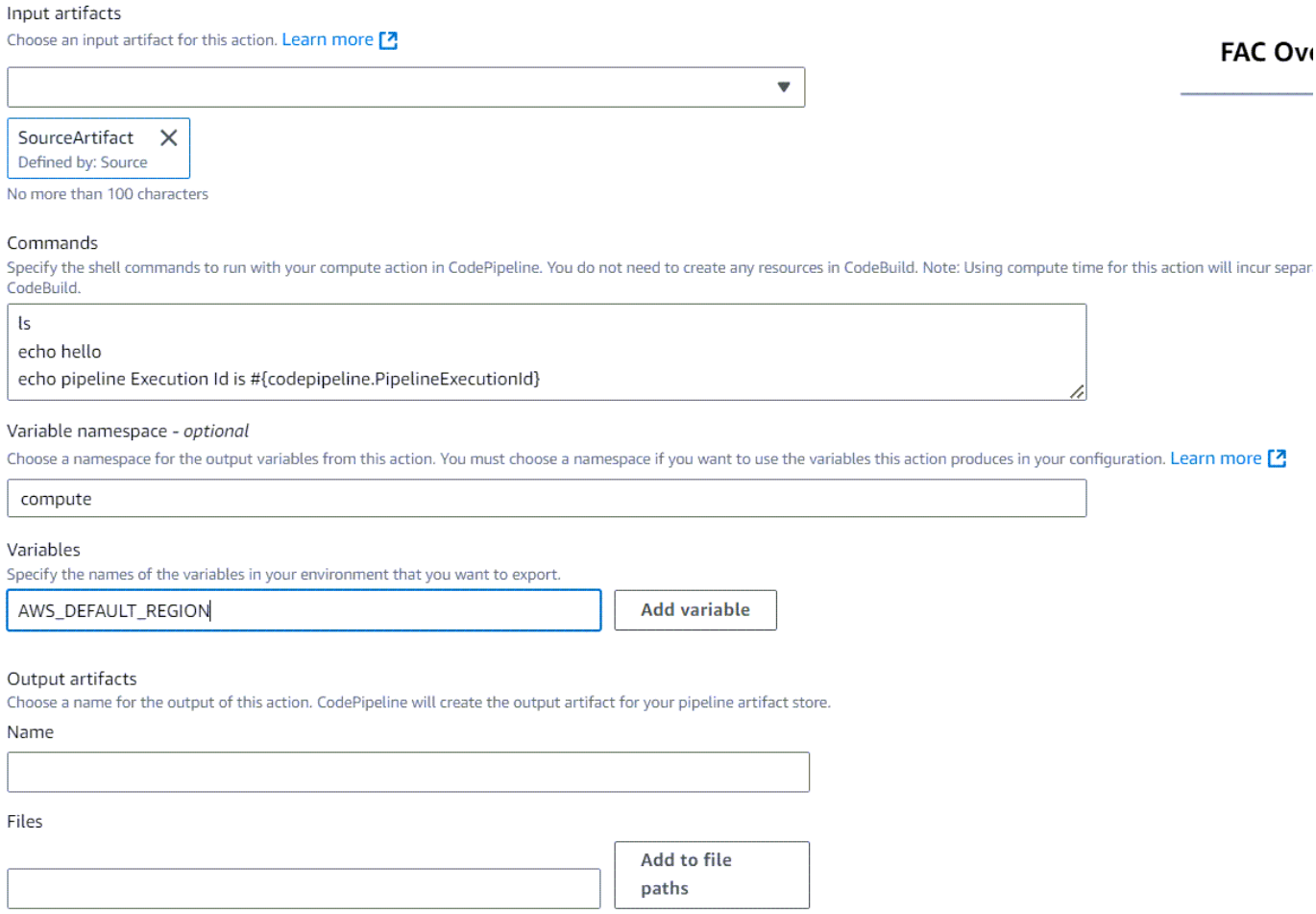
11. Na página Etapa 6: Adicionar etapa de implantação, escolha Ignorar etapa de implantação e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

12. Na Etapa 7: Revisar, revise as informações e escolha Criar funil.

13. Como etapa final para criar a ação, adicione uma variável de ambiente à ação que resultará em uma variável de saída para a ação. Na ação Commands, selecione Editar. Na tela Editar, especifique um namespace de variável para sua ação inserindo `compute` no campo Namespace de variável.

    Adicione a variável CodeBuild `AWS_Default_Region` de saída e escolha Adicionar variável.



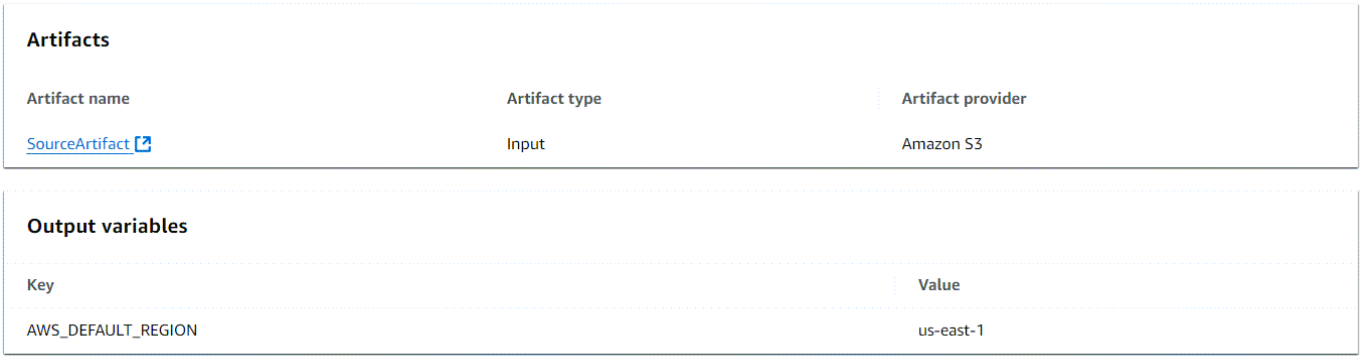
Etapa 3: executar o pipeline e verificar os comandos de compilação

Implemente uma alteração para iniciar a execução do pipeline. Verifique se os comandos de compilação foram executados visualizando o histórico de execução, os logs de compilação e as variáveis de saída.

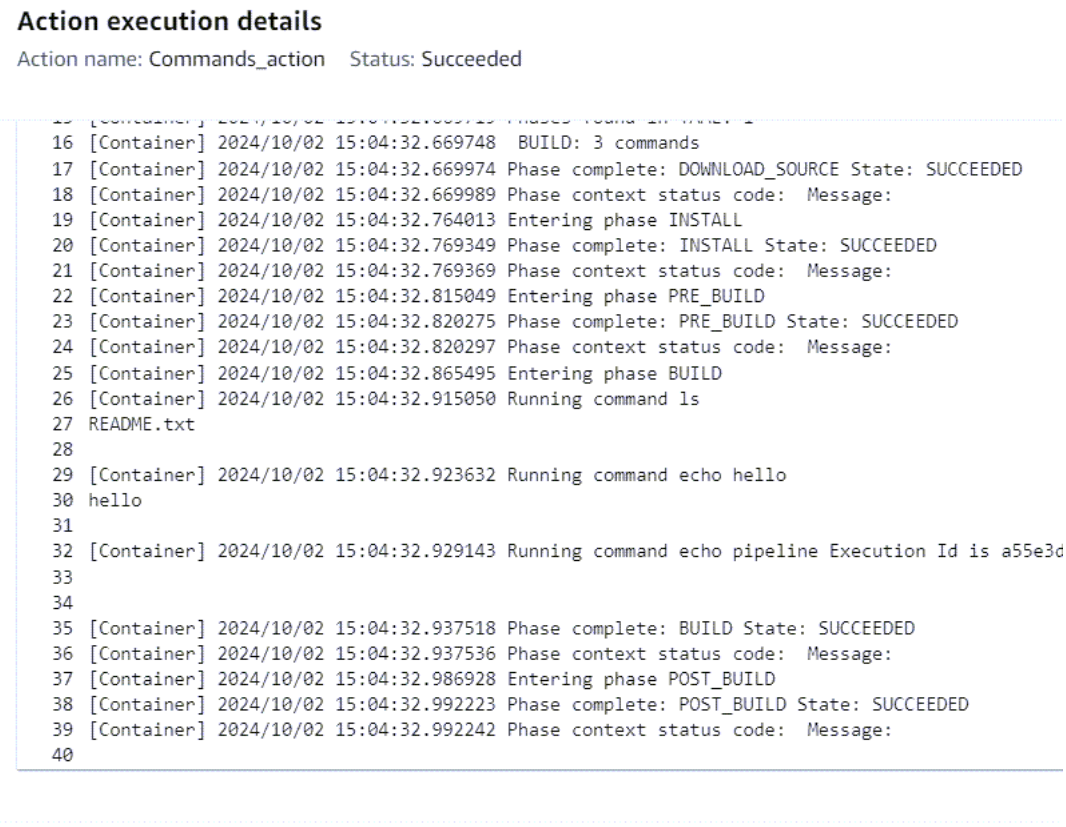
Como visualizar logs de ação e variáveis de saída

1. Após a execução bem-sucedida do pipeline, você poderá visualizar os logs e a saída da ação.
2. Para visualizar as variáveis de saída da ação, escolha Histórico > Linha do tempo.

   Visualize a variável de saída que foi adicionada à ação. A saída da ação Commands exibe a variável de saída resolvida para a região da ação.



3. Para visualizar os logs da ação, escolha Visualizar detalhes na ação Commands bem-sucedida. Visualize os logs da ação Commands.



Tutorial: Usar tags do Git para iniciar o pipeline

Neste tutorial, você criará um pipeline que se conecta ao seu GitHub repositório onde a ação de origem é configurada para o tipo de gatilho de tags Git. Quando uma tag do Git é criada em uma confirmação, o pipeline é iniciado. Este exemplo mostra como criar um pipeline que permite a filtragem de tags com base na sintaxe do nome da tag. Para obter mais informações sobre a filtragem de padrões glob, consulte Trabalhar com padrões glob na sintaxe.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Este tutorial se conecta a GitHub por meio do tipo de `CodeStarSourceConnection` ação.

> **Note**
>
> Este atributo não está disponível nas regiões Ásia-Pacífico (Hong Kong), África (Cidade do Cabo), Oriente Médio (Bahrein) ou Europa (Zurique). Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline.
> Para considerações sobre essa ação na região Europa (Milão), consulte a nota em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

Tópicos

- Pré-requisitos
- Etapa 1: abrir CloudShell e clonar seu repositório
- Etapa 2: Criar um pipeline para acionar as tags do Git
- Etapa 3: Marcar as confirmações para lançamento
- Etapa 4: Lançar alterações e visualizar logs

Pré-requisitos

Antes de começar, é necessário fazer o seguinte:

- Crie um GitHub repositório com sua GitHub conta.
- Tenha suas GitHub credenciais prontas. Quando você usa o Console de gerenciamento da AWS para configurar uma conexão, é solicitado que você faça login com suas GitHub credenciais.

Etapa 1: abrir CloudShell e clonar seu repositório

É possível usar uma interface de linha de comandos para clonar o repositório, fazer confirmações e adicionar tags. Este tutorial inicia uma CloudShell instância para a interface da linha de comando.

1. Faça login no Console de gerenciamento da AWS.
2. Na barra de navegação superior, escolha o AWS ícone. A página principal dos Console de gerenciamento da AWS é exibida.

3. Na barra de navegação superior, escolha o AWS CloudShell ícone. CloudShell abre. Espere enquanto o CloudShell ambiente é criado.

> **Note**
>
> Se você não vê o CloudShell ícone, verifique se você está em uma região suportada por CloudShell. Este tutorial pressupõe que você esteja na região do Oeste dos EUA (Oregon).

4. Em GitHub, navegue até seu repositório. Selecione Editar e HTTPS. Copie o caminho. O endereço para clonar o repositório Git é copiado na área de transferência.
5. Execute o comando a seguir para clonar o repositório.

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

6. Entre na sua GitHub conta `Username` e `Password` quando solicitado. Para a entrada `Password`, é necessário usar um token criado pelo usuário em vez da senha da conta.

Etapa 2: Criar um pipeline para acionar as tags do Git

Nesta seção, você criará um pipeline com as seguintes ações:

- Um estágio de origem com uma conexão com seu GitHub repositório e ação.
- Um estágio de construção com uma ação de AWS CodeBuild construção.

Criar um pipeline com o assistente

1. Faça login no CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyGitHubTagsPipeline`**.
5. Em Tipo de pipeline, mantenha a seleção padrão em V2. Os tipos de pipeline diferem em características e preços. Para obter mais informações, consulte Tipos de pipeline.

6. Em Service role (Função do serviço), selecione New service role (Nova função de serviço).

   > **Note**
   >
   > Se, em vez disso, você optar por usar sua função de CodePipeline serviço existente, certifique-se de ter adicionado a permissão `codestar-connections:UseConnection` do IAM à sua política de função de serviço. Para obter instruções sobre a função CodePipeline de serviço, consulte Adicionar permissões à função CodePipeline de serviço.

7. Em Configurações avançadas mantenha os padrões. Em Artifact store (Armazenamento de artefatos), selecione Default location (Local padrão) para usar o armazenamento de artefatos padrão, como o bucket de artefatos do Amazon S3 designado como padrão, para o pipeline na região que você selecionou.

   > **Note**
   >
   > Este não é o bucket de origem para seu código-fonte. Este é o armazenamento de artefatos para o pipeline. Um armazenamento de artefatos separado, como um bucket do S3, é necessário para cada pipeline.

   Escolha Próximo.

8. Na página Etapa 3: Adicionar etapa de origem, adicione uma etapa de origem:

   a. Em Provedor de origem, escolha GitHub (via GitHub aplicativo).

   b. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulteGitHub conexões.

   c. Em Repository name (Nome do repositório), selecione o nome do seu repositório do GitHub.

   d. Em Ramificação padrão, selecione a ramificação que você deseja especificar quando o pipeline é iniciado manualmente ou com um evento de origem que não seja uma tag do Git. Se a origem da alteração não for o gatilho ou se a execução de um pipeline tiver sido iniciada manualmente, a alteração usada será a confirmação HEAD da ramificação padrão.

   e. Em Eventos de webhook, em Tipo de filtro, escolha Tags.

      No campo Tags ou padrões, insira `release*`.

> **Important**
>
> Os pipelines que começam com um tipo de gatilho de tags do Git serão configurados para eventos WebhookV2 e não usarão o evento do Webhook (detecção de alterações em todos os eventos push) para iniciar o pipeline.

Escolha Próximo.

9. Em Add build stage (Adicionar estágio de compilação), adicione um estágio de compilação:

   a. Em Build provider (Provedor de compilação), escolha AWS CodeBuild. Permita que Region (Região) seja definida para a região do pipeline.

   b. Escolha Criar projeto.

   c. Em Nome do projeto, insira um nome para esse projeto de compilação.

   d. Em Environment image (Imagem do ambiente), escolha Managed image (Imagem gerenciada). Para Operating system, selecione Ubuntu.

   e. Em Runtime (Tempo de execução), selecione Standard (Padrão). Em Imagem, escolha aws/codebuild /standard:5.0.

   f. Em Service role (Função de serviço), selecione New service role (Nova função de serviço).

   > **Note**
   >
   > Anote o nome da sua função CodeBuild de serviço. Você precisará do nome do perfil para a etapa final deste tutorial.

   g. Em Buildspec, para Build specifications (Especificações da compilação), escolha Insert build commands (Inserir comandos de compilação). Selecione Alternar para o editor e cole o seguinte em Comandos de compilação.

```
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. Escolha Continuar para CodePipeline. Isso retorna ao CodePipeline console e cria um CodeBuild projeto que usa seus comandos de construção para configuração. O projeto de construção usa uma função de serviço para gerenciar AWS service (Serviço da AWS) permissões. Essa etapa pode levar alguns minutos.

i. Escolha Próximo.

10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

Escolha Próximo.

11. Na página Etapa 6: Adicionar estágio de implantação, escolha Ignorar estágio de implantação e aceite a mensagem de aviso escolhendo Ignorar novamente. Escolha Próximo.
12. Na Etapa 7: Revisão, escolha Criar funil.

Etapa 3: Marcar as confirmações para lançamento

Depois de criar seu pipeline e especificar as tags do Git, você pode marcar os commits no seu GitHub repositório. Nestas etapas, você marcará uma confirmação com a tag `release-1`. Cada confirmação em um repositório do Git deve ter uma tag exclusiva do Git. Ao selecionar a confirmação e marcá-la, isso permite incorporar alterações de diferentes ramificações na implantação do pipeline. Observe que o nome da tag release não se aplica ao conceito de lançamento em GitHub.

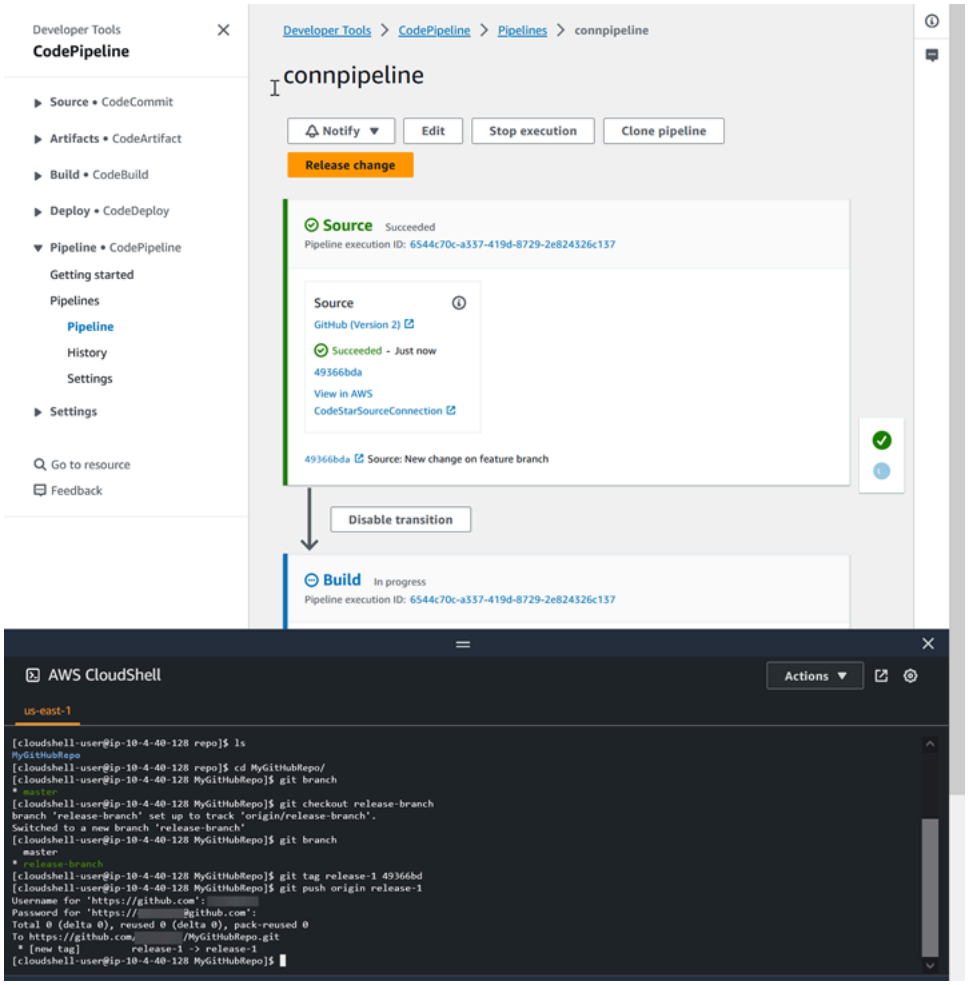
1. Faça referência aos IDs de confirmação copiadas que você deseja marcar. Para visualizar os commits em cada ramificação, no CloudShell terminal, digite o seguinte comando para capturar os IDs de commit que você deseja marcar:

   ```
   git log
   ```

2. No CloudShell terminal, digite o comando para marcar seu commit e enviá-lo para a origem. Depois de marcar a confirmação, use o comando git push para enviar a tag à origem. No exemplo a seguir, insira este comando para usar a tag `release-1` para a segunda confirmação com o ID `49366bd`. Essa tag será filtrada pelo filtro de tags `release*` do pipeline e iniciará o pipeline.

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

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



Etapa 4: Lançar alterações e visualizar logs

1. Depois que o pipeline for executado com êxito, no estágio de criação bem-sucedido, selecione Visualizar log.

   Em Logs, veja a saída da CodeBuild compilação. Os comandos geram o valor da variável inserida.

2. Na página Histórico, visualize a coluna Gatilhos. Veja o tipo de gatilho GitTag : release-1.

Tutorial: Filtrar nomes de ramificação para solicitações pull a fim de iniciar o pipeline (tipo V2)

Neste tutorial, você criará um pipeline que se conecta ao seu GitHub.com repositório, onde a ação de origem é configurada para iniciar seu pipeline com uma configuração de gatilho que filtra as pull requests. Quando um evento de solicitação pull especificado ocorre para uma ramificação específica, o pipeline é iniciado. Este exemplo mostra como criar um pipeline que permite a filtragem de nomes de ramificações. Para ter mais informações sobre como trabalhar com gatilhos, consulte Adicionar filtros para tipos de evento push e pull request (CLI). Para ter mais informações sobre como filtrar com padrões regex no formato glob, consulte Trabalhar com padrões glob na sintaxe.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Este tutorial se conecta a GitHub.com por meio do tipo de `CodeStarSourceConnection` ação.

Tópicos

- Pré-requisitos
- Etapa 1: criar um pipeline para iniciar uma solicitação pull para ramificações especificadas
- Etapa 2: crie e mescle uma pull request GitHub.com para iniciar suas execuções de pipeline

Pré-requisitos

Antes de começar, é necessário fazer o seguinte:

- Crie um GitHub.com repositório com sua GitHub.com conta.
- Tenha suas GitHub credenciais prontas. Quando você usa o Console de gerenciamento da AWS para configurar uma conexão, é solicitado que você faça login com suas GitHub credenciais.

Etapa 1: criar um pipeline para iniciar uma solicitação pull para ramificações especificadas

Nesta seção, você criará um pipeline com as seguintes ações:

- Um estágio de origem com uma conexão com seu GitHub.com repositório e ação.
- Um estágio de construção com uma ação de AWS CodeBuild construção.

Criar um pipeline com o assistente

1. Faça login no CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyFilterBranchesPipeline`**.
5. Em Tipo de pipeline, mantenha a seleção padrão em V2. Os tipos de pipeline diferem em características e preços. Para obter mais informações, consulte Tipos de pipeline.
6. Em Service role (Função do serviço), selecione New service role (Nova função de serviço).

   > **Note**
   >
   > Se, em vez disso, você optar por usar sua função de CodePipeline serviço existente, certifique-se de ter adicionado a permissão `codeconnections:UseConnection` do IAM à sua política de função de serviço. Para obter instruções sobre a função CodePipeline de serviço, consulte Adicionar permissões à função CodePipeline de serviço.

7. Em Configurações avançadas mantenha os padrões. Em Artifact store (Armazenamento de artefatos), selecione Default location (Local padrão) para usar o armazenamento de artefatos padrão, como o bucket de artefatos do Amazon S3 designado como padrão, para o pipeline na região que você selecionou.

> **Note**
>
> Este não é o bucket de origem para seu código-fonte. Este é o armazenamento de artefatos para o pipeline. Um armazenamento de artefatos separado, como um bucket do S3, é necessário para cada pipeline.

Escolha Próximo.

8. Na página Etapa 3: Adicionar etapa de origem, adicione uma etapa de origem:

   a. Em Provedor de origem, escolha GitHub (via GitHub aplicativo).

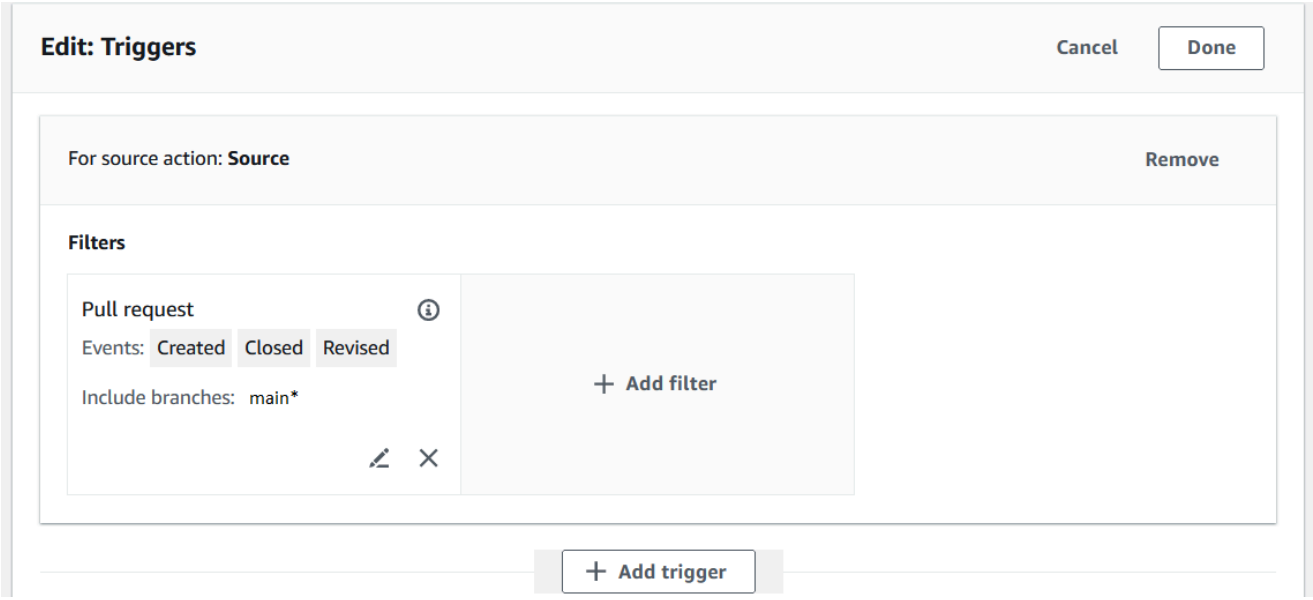
   b. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulteGitHub conexões.

   c. Em Repository name (Nome do repositório), selecione o nome do seu repositório do GitHub.com.

   d. Em Tipo de gatilho, escolha Especificar filtro.

      Em Tipo de evento, escolha Solicitação pull. Selecione todos os eventos em solicitação pull para que sejam acionados ao criar, atualizar ou fechar as solicitações pull.

      Em Ramificações, no campo Incluir, insira `main*`.



> **Important**
>
> Os pipelines que começam com esse tipo de gatilho serão configurados para eventos WebhookV2 e não usarão o evento Webhook (detecção de alterações em todos os eventos push) para iniciar o pipeline.

Escolha Próximo.

9. Em Etapa 4: Adicionar etapa de compilação, em Provedor de compilação, escolha AWS CodeBuild. Permita que Region (Região) seja definida para a região do pipeline. Escolha ou crie o projeto de compilação conforme as instruções em Tutorial: Usar tags do Git para iniciar o pipeline. Esta ação será usada neste tutorial apenas como o segundo estágio necessário para criar o pipeline.
10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

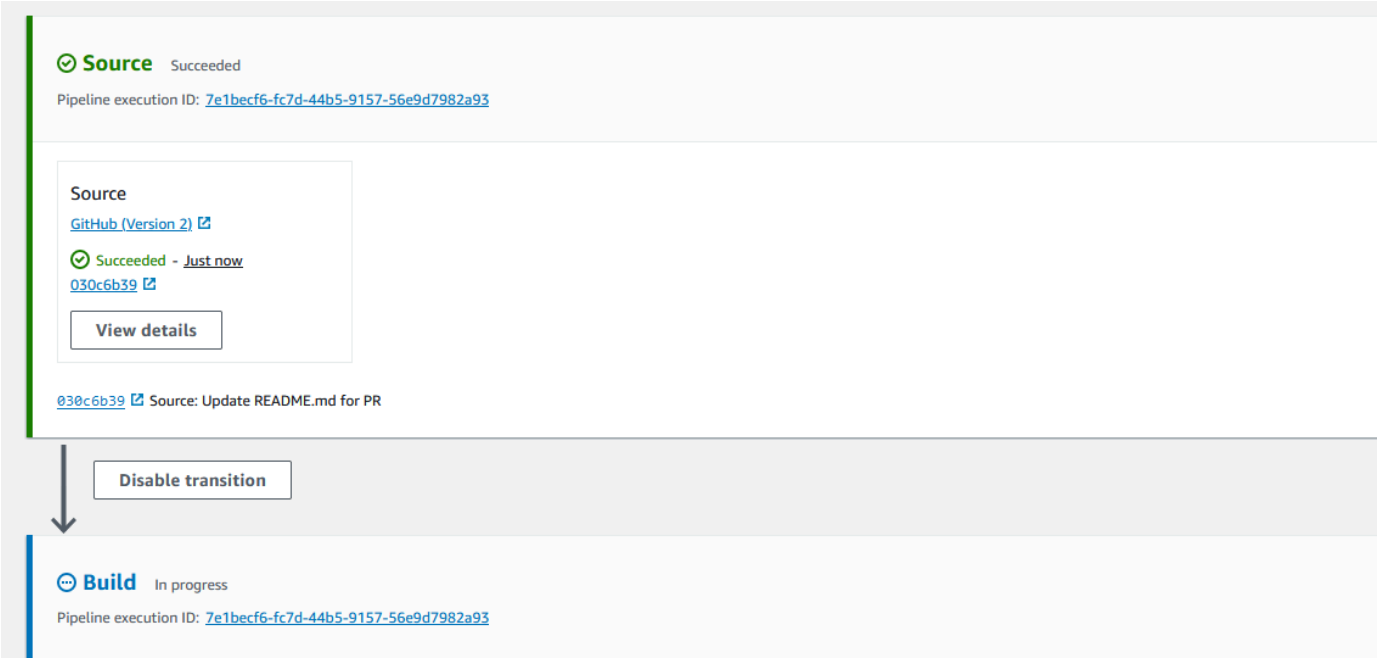
    Escolha Próximo.
11. Na página Etapa 6: Adicionar estágio de implantação, escolha Ignorar estágio de implantação e aceite a mensagem de aviso escolhendo Ignorar novamente. Escolha Próximo.
12. Na Etapa 7: Revisão, escolha Criar funil.

Etapa 2: crie e mescle uma pull request GitHub.com para iniciar suas execuções de pipeline

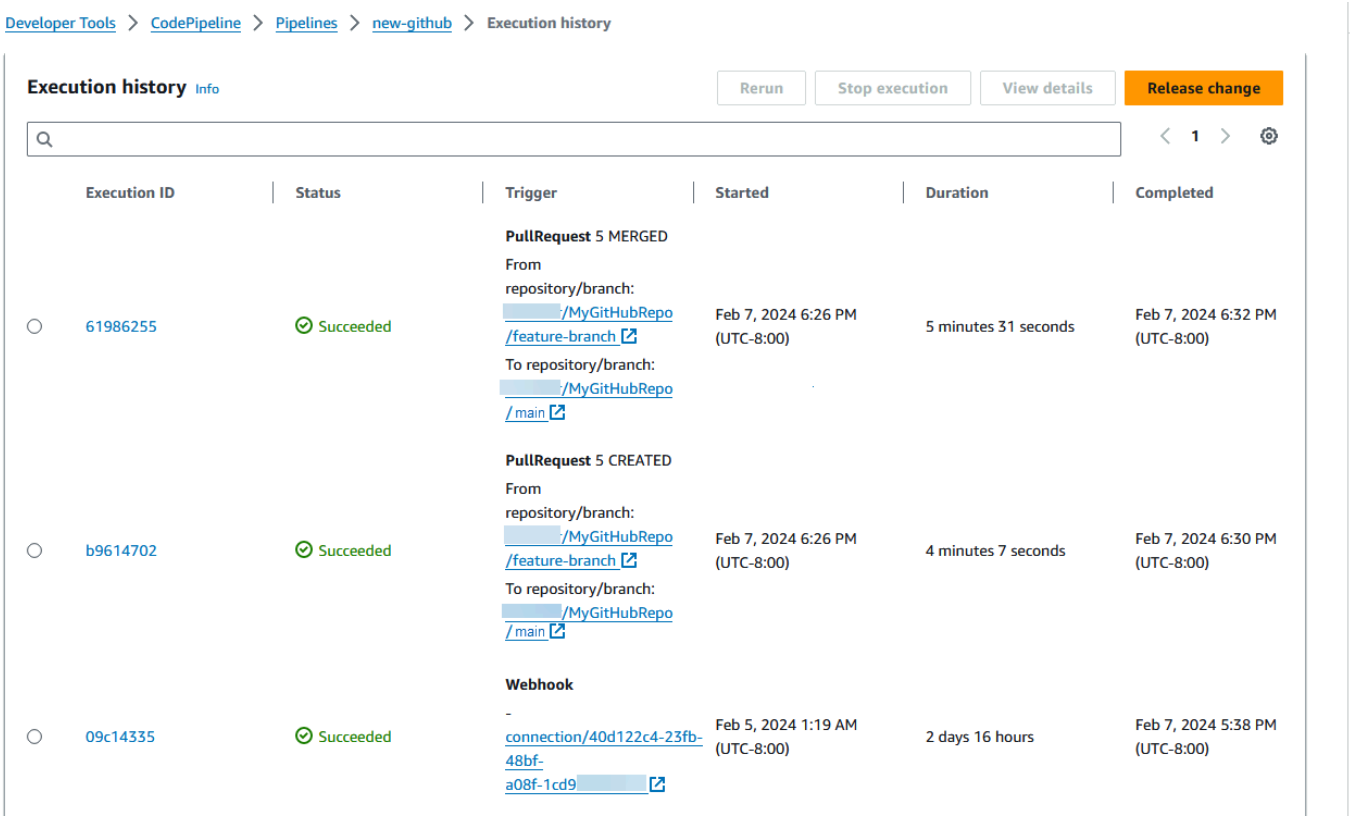
Nesta seção, você cria e mescla uma solicitação pull. Isso inicia o pipeline, com uma execução para solicitação pull aberta e uma execução para solicitação pull fechada.

Como criar uma solicitação pull e iniciar o pipeline

1. Em GitHub.com, crie uma pull request fazendo uma alteração na ramificação README.md on a feature e gerando uma pull request para a `main` ramificação. Confirme a alteração com uma mensagem como `Update README.md for PR`.
2. O pipeline começa com a revisão da fonte mostrando a mensagem de origem da pull request como Atualização README.md para PR.



3. Escolha History (Histórico). No histórico de execução do pipeline, visualize os eventos de status de solicitação pull CREATED e MERGED que iniciaram as execuções do pipeline.



Tutorial: Usar variáveis no nível do pipeline

Neste tutorial, você criará um pipeline no qual adicionará uma variável no nível do pipeline e executará uma ação de CodeBuild construção que gera o valor da variável.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Tópicos

- Pré-requisitos
- Etapa 1: Criar seu pipeline e compilar o projeto
- Etapa 2: Lançar alterações e visualizar logs

Pré-requisitos

Antes de começar, é necessário fazer o seguinte:

- Crie um CodeCommit repositório.
- Adicione um arquivo .txt a um repositório.

Etapa 1: Criar seu pipeline e compilar o projeto

Nesta seção, você criará um pipeline com as seguintes ações:

- Um estágio de origem com uma conexão com seu CodeCommit repositório.
- Um estágio de construção com uma ação de AWS CodeBuild construção.

Criar um pipeline com o assistente

1. Faça login no CodePipeline console em https://console.aws.amazon.com/codepipeline/.

2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyVariablesPipeline`**.
5. Em Tipo de pipeline, mantenha a seleção padrão em V2. Os tipos de pipeline diferem em características e preços. Para obter mais informações, consulte Tipos de pipeline.
6. Em Service role (Função do serviço), selecione New service role (Nova função de serviço).

   > **Note**
   >
   > Se, em vez disso, você optar por usar sua função de CodePipeline serviço existente, certifique-se de ter adicionado a permissão `codeconnections:UseConnection` do IAM à sua política de função de serviço. Para obter instruções sobre a função CodePipeline de serviço, consulte Adicionar permissões à função CodePipeline de serviço.

7. Em Variáveis, selecione Adicionar variável. Em Nome, insira `timeout`. Em Padrão, insira 1000. Em Descrição, insira a seguinte descrição: **`Timeout`**.

   Isso criará uma variável na qual você poderá declarar o valor quando a execução do pipeline começar. Os nomes das variáveis devem corresponder a `[A-Za-z0-9@\-_]+` e podem ser qualquer coisa, exceto uma string vazia.
8. Em Configurações avançadas mantenha os padrões. Em Artifact store (Armazenamento de artefatos), selecione Default location (Local padrão) para usar o armazenamento de artefatos padrão, como o bucket de artefatos do Amazon S3 designado como padrão, para o pipeline na região que você selecionou.

   > **Note**
   >
   > Este não é o bucket de origem para seu código-fonte. Este é o armazenamento de artefatos para o pipeline. Um armazenamento de artefatos separado, como um bucket do S3, é necessário para cada pipeline.

   Escolha Próximo.

9. Na página Etapa 3: Adicionar etapa de origem, adicione uma etapa de origem:

   a. Em Source provider (Provedor de código-fonte), selecione AWS CodeCommit.

   b. Em Nome do repositório e Nome da ramificação, escolha seu repositório e ramificação.

   Escolha Próximo.

10. Em Etapa 4: Adicionar etapa de compilação, adicione um estágio de compilação:

    a. Em Build provider (Provedor de compilação), escolha AWS CodeBuild. Permita que Region (Região) seja definida para a região do pipeline.

    b. Escolha Criar projeto.

    c. Em Nome do projeto, insira um nome para esse projeto de compilação.

    d. Em Environment image (Imagem do ambiente), escolha Managed image (Imagem gerenciada). Para Operating system, selecione Ubuntu.

    e. Em Runtime (Tempo de execução), selecione Standard (Padrão). Em Imagem, escolha aws/codebuild /standard:5.0.

    f. Em Service role (Função de serviço), selecione New service role (Nova função de serviço).

    > **Note**
    >
    > Anote o nome da sua função CodeBuild de serviço. Você precisará do nome do perfil para a etapa final deste tutorial.

    g. Em Buildspec, para Build specifications (Especificações da compilação), escolha Insert build commands (Inserir comandos de compilação). Selecione Alternar para o editor e cole o seguinte em Comandos de compilação. No buildspec, a variável do cliente `$CUSTOM_VAR1` será usada para gerar a variável do pipeline no log de criação. Você criará a variável de saída `$CUSTOM_VAR1` como uma variável de ambiente na etapa a seguir.

```
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. Escolha Continuar para CodePipeline. Isso retorna ao CodePipeline console e cria um CodeBuild projeto que usa seus comandos de construção para configuração. O projeto de construção usa uma função de serviço para gerenciar AWS service (Serviço da AWS) permissões. Essa etapa pode levar alguns minutos.

i. Em Variáveis de ambiente - opcional, para criar uma variável de ambiente como uma variável de entrada para a ação de criação que será resolvida pela variável em nível de pipeline, selecione Adicionar variável de ambiente. Isso criará a variável especificada no

buildspec como `$CUSTOM_VAR1`. Em Nome, insira `CUSTOM_VAR1`. Em Valor, informe `#{variables.timeout}`. Em Tipo, escolha`Plaintext`.

O `#{variables.timeout}` valor da variável de ambiente é baseado no namespace da variável no nível do pipeline `variables` e na variável no nível do pipeline criada para o pipeline na etapa 7. `timeout`

j. Escolha Próximo.

11. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

12. Na página Etapa 6: Adicionar estágio de implantação, escolha Ignorar estágio de implantação e aceite a mensagem de aviso escolhendo Ignorar novamente. Escolha Próximo.

13. Na Etapa 7: Revisão, escolha Criar funil.

Etapa 2: Lançar alterações e visualizar logs

1. Depois que o pipeline for executado com êxito, no estágio de criação bem-sucedido, selecione Visualizar detalhes.

   Na página de detalhes, selecione a guia Logs. Veja a saída da CodeBuild compilação. Os comandos geram o valor da variável inserida.

2. No painel de navegação à esquerda, selecione Histórico.

   Escolha a execução recente e, depois, selecione a guia Variáveis. Visualize o valor resolvido para a variável do pipeline.

Tutorial: Criar um pipeline simples (bucket do S3)

A maneira mais fácil de criar um pipeline é usar o assistente Create Pipeline no AWS CodePipeline console.

Neste tutorial, você cria um pipeline de dois estágios que usa um bucket de origem do S3 com versão e lança um aplicativo de CodeDeploy amostra.

> **Note**
>
> Quando o Amazon S3 é o provedor de origem do pipeline, é possível compactar o(s) arquivo(s) de origem em um único .zip e fazer upload do .zip para o bucket de origem. Também é possível fazer upload de um único arquivo descompactado; no entanto, ocorrerão falha nas ações downstream que aguardam um arquivo .zip.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Após criar esse pipeline simples, adicione outro estágio e, em seguida, desabilite e habilite a transição entre os estágios.

> **Important**
>
> Muitas das ações que você adiciona ao seu funil neste procedimento envolvem AWS recursos que você precisa criar antes de criar o funil. AWS os recursos para suas ações de origem sempre devem ser criados na mesma AWS região em que você cria seu funil. Por exemplo, se você criar seu funil na região Leste dos EUA (Ohio), seu CodeCommit repositório deverá estar na região Leste dos EUA (Ohio).
> Você pode adicionar ações entre regiões ao criar seu funil. AWS os recursos para ações entre regiões devem estar na mesma AWS região em que você planeja executar a ação. Para obter mais informações, consulte Adicione uma ação entre regiões em CodePipeline.

Antes de começar, você deve cumprir os pré-requisitos em Começando com CodePipeline.

Tópicos

- Etapa 1: criar um bucket de origem do S3 para a aplicação
- Etapa 2: Crie instâncias Windows do Amazon EC2 e instale o agente CodeDeploy

- Etapa 3: criar um aplicativo no CodeDeploy
- Etapa 4: crie seu primeiro funil no CodePipeline
- (Opcional) Etapa 5: Adicionar outra etapa ao pipeline
- (Opcional) Etapa 6: desativar e ativar as transições entre os estágios no CodePipeline
- Etapa 7: Limpar os recursos

Etapa 1: criar um bucket de origem do S3 para a aplicação

É possível armazenar os aplicativos ou arquivos de origem em qualquer local versionado. Neste tutorial, você criará um bucket do S3 para os exemplos de arquivo de aplicação e habilitar o versionamento nesse bucket. Após habilitar o versionamento, copie os aplicativos de exemplo para esse bucket.

Para criar um bucket do S3

1. Faça login no console em Console de gerenciamento da AWS. Abra o console do S3.
2. Escolha Create bucket (Criar bucket).
3. No Bucket name (Nome do bucket), insira um nome para o seu bucket (por exemplo, **`awscodepipeline-demobucket-example-date`**).

   > **Note**
   >
   > Como todos os nomes de bucket no Amazon S3 devem ser exclusivos, use um dos próprios nomes e não o nome exibido no exemplo. Você pode alterar o nome de exemplo simplesmente adicionando uma data. Anote esse nome, pois você precisará usá-lo mais adiante neste tutorial.

   Em Região, selecione a região onde você pretende criar o pipeline, como Oeste dos EUA (Oregon) e, depois, selecione Criar bucket.
4. Depois que o bucket é criado, um banner de sucesso é exibido. Escolha Go to bucket details (Ir para detalhes do bucket).
5. Na guia Properties (Propriedades) escolha Versioning (Versionamento). Escolha Enable versioning (Ativar versionamento) e escolha Save (Salvar).

Quando o versionamento é habilitado, o Amazon S3 salva todas as versões de cada objeto no bucket.

6. Na guia Permissions (Permissões) deixe os valores padrão. Para obter mais informações sobre permissões de bucket e objeto do S3, consulte Especificar permissões em uma política.
7. Depois, faça download de um exemplo e salve-o em uma pasta ou um diretório no computador local.

   a. Escolha uma das opções a seguir. Escolha `SampleApp_Windows.zip` se deseja seguir as etapas deste tutorial para instâncias do Windows Server.

   - Se você quiser implantar em instâncias Amazon Linux usando CodeDeploy, baixe o aplicativo de amostra aqui: SampleApp _ Linux.zip.
   - Se você quiser implantar em instâncias do Windows Server usando CodeDeploy, baixe o aplicativo de amostra aqui: SampleApp _ Windows.zip.

   O aplicativo de amostra contém os seguintes arquivos para implantação com CodeDeploy:

   - `appspec.yml`— O arquivo de especificação do aplicativo (AppSpecarquivo) é um arquivo formatado em YAML usado pelo CodeDeploy para gerenciar uma implantação. Para obter mais informações sobre o AppSpec arquivo, consulte Referência CodeDeploy AppSpec do arquivo no Guia AWS CodeDeploy do usuário.
   - `index.html`: o arquivo de índice contém a página inicial da aplicação de exemplo implantada.
   - `LICENSE.txt`: o arquivo de licença contém informações de licença da aplicação de exemplo.
   - Arquivos para scripts: a aplicação de exemplo usa scripts para gravar arquivos de texto em um local na instância. Um arquivo é gravado para cada um dos vários eventos do ciclo de vida da CodeDeploy implantação da seguinte forma:
     - Pasta `scripts` (somente exemplo do Linux): a pasta contém os seguintes scripts de shell para instalar dependências e iniciar e interromper a aplicação de exemplo para a implantação automatizada: `install_dependencies`, `start_server` e `stop_server`.
     - (Somente exemplo do Windows) `before-install.bat`: um script em lote para o evento de ciclo de vida de implantação `BeforeInstall`, que será executado para

remover os arquivos antigos gravados durante implantações anteriores deste exemplo e criar um local na instância onde os novos arquivos serão gravados.

b. Faça download do arquivo compactado. Não descompacte o arquivo.

8. No console do Amazon S3, para o bucket, faça upload do arquivo:

a. Escolha Carregar.

b. Arraste e solte o arquivo ou escolha Add files (Adicionar arquivos) e navegue até o arquivo.

c. Escolha Carregar.

Etapa 2: Crie instâncias Windows do Amazon EC2 e instale o agente CodeDeploy

> **Note**
>
> Este tutorial fornece etapas de exemplo para criar instâncias do Windows do Amazon EC2. Para obter etapas de exemplo para criar instâncias do Linux do Amazon EC2, consulte Etapa 3: Crie uma instância Linux do Amazon EC2 e instale o agente CodeDeploy. Quando solicitado o número de instâncias a serem criadas, especifique 2 instâncias.

Nesta etapa, você vai criar as instâncias do Amazon EC2 do Windows Server nas quais implantará uma aplicação de exemplo. Como parte desse processo, você cria uma função de instância com políticas que permitem a instalação e o gerenciamento do CodeDeploy agente nas instâncias. O CodeDeploy agente é um pacote de software que permite que uma instância seja usada em CodeDeploy implantações. Você também anexa políticas que permitem que a instância busque arquivos que o CodeDeploy agente usa para implantar seu aplicativo e permitir que a instância seja gerenciada pelo SSM.

Como criar uma função de instância

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel do console, escolha Roles (Funções).
3. Selecione Criar perfil.
4. Em Selecionar tipo de entidade confiável, selecione AWS service (Serviço da AWS). Em Choose a use case (Escolher um caso de uso), selecione EC2 e escolha Next: Permissions (Próximo: permissões).

5. Procure e selecione a política chamada **`AmazonEC2RoleforAWSCodeDeploy`**.
6. Procure e selecione a política chamada **`AmazonSSMManagedInstanceCore`**. Escolha Próximo: tags.
7. Selecione Próximo: revisar. Forneça um nome para a função (por exemplo, **`EC2InstanceRole`**).

   > **Note**
   >
   > Anote o nome da função para a próxima etapa. Escolha essa função ao criar a instância.

   Selecione Criar perfil.

Para executar instâncias

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Na navegação lateral, escolha Instâncias e selecione Executar instâncias na parte superior da página.
3. Em Nome e tags, em Nome, insira **`MyCodePipelineDemo`**. Isso atribui à instância uma tag Chave de **`Name`** e uma tag Valor de **`MyCodePipelineDemo`**. Posteriormente, você cria um CodeDeploy aplicativo que implanta o aplicativo de amostra nas instâncias. CodeDeployseleciona instâncias a serem implantadas com base nas tags.
4. Em Imagens de aplicações e sistemas operacionais (imagem de máquina da Amazon), selecione a opção Windows. (Essa AMI é descrita como Microsoft Windows Server 2019 Base e é rotulada como “Elegível para o nível gratuito” e pode ser encontrada em Início rápido.)
5. Em Tipo de instância, selecione o tipo `t2.micro` elegível para o nível gratuito como configuração de hardware para a instância.
6. Na seção Par de chaves (login), selecione um par de chaves ou crie um.

   Também é possível selecionar Prosseguir sem um par de chaves.

   > **Note**
   >
   > Para os fins deste tutorial, é possível prosseguir sem um par de chaves. Para usar o SSH para se conectar às instâncias, crie ou use um par de chaves.

7. Em Configurações de rede, faça o seguinte:

   Em IP Auto-assign público, verifique se o status é Ativado.

   - Ao lado de Assign a security group (Atribuir um grupo de segurança), selecione Create a new security group (Criar novo grupo de segurança).
   - Na linha para SSH, em Tipo de origem, selecione Meu IP.
   - Selecione Adicionar grupo de segurança, selecione HTTP e, depois, em Tipo de origem, selecione Meu IP.
8. Expanda Advanced details (Detalhes avançados). Em Perfil de instância do IAM, selecione o perfil do IAM criado no procedimento anterior (por exemplo, **`EC2InstanceRole`**).
9. Em Resumo, em Número de instâncias, insira 2.
10. Escolha Iniciar instância.
11. Escolha View all instances (Visualizar todas as instâncias) para fechar a página de confirmação e voltar ao console.
12. É possível visualizar o status da ativação na página Instâncias. Ao executar uma instância, seu estado inicial é `pending`. Após o início da instância, seu estado muda para `running` e ela recebe um nome DNS público. (Se a coluna DNS público não for exibida, escolha o Show/Hide ícone e selecione DNS público.)
13. Pode levar alguns minutos até que a instância esteja pronta para sua conexão. Verifique se a instância passou nas verificações de status. Você pode visualizar essas informações na coluna Status Checks (Verificações de status).

Etapa 3: criar um aplicativo no CodeDeploy

Em CodeDeploy, um aplicativo é um identificador, na forma de um nome, para o código que você deseja implantar. CodeDeploy usa esse nome para garantir que a combinação correta de revisão, configuração de implantação e grupo de implantação seja referenciada durante uma implantação. Você seleciona o nome do CodeDeploy aplicativo criado nesta etapa ao criar seu funil posteriormente neste tutorial.

Primeiro, você cria uma função de serviço CodeDeploy para usar. Se você já criou um perfil de serviço, não precisará criar outro.

Para criar uma função CodeDeploy de serviço

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel do console, escolha Roles (Funções).
3. Selecione Criar perfil.
4. Em Selecionar entidade confiável, escolha AWS service (Serviço da AWS). Em Use case (Caso de uso), escolha CodeDeploy. CodeDeployEscolha entre as opções listadas. Escolha Próximo. A política gerenciada `AWSCodeDeployRole` já está anexada à função.
5. Escolha Próximo.
6. Insira um nome para a função (por exemplo, **`CodeDeployRole`**) e escolha Create role (Criar função).

Para criar um aplicativo no CodeDeploy

1. Abra o CodeDeploy console em https://console.aws.amazon.com/codedeploy.
2. Se a página Aplicativos não for exibida, no AWS CodeDeploy menu, escolha Aplicativos.
3. Escolha Criar aplicativo.
4. Em Nome do aplicativo, insira `MyDemoApplication`.
5. Em Compute Platform, escolha EC2/On-premises.
6. Selecione Criar aplicativo.

Para criar um grupo de implantação no CodeDeploy

1. Na página que mostra o aplicativo, selecione Create deployment group (Criar grupo de implantação).
2. Em Nome do grupo de implantação, insira **`MyDemoDeploymentGroup`**.
3. Em Perfil de serviço, selecione o perfil de serviço criado anteriormente. Você deve usar uma função de serviço que AWS CodeDeploy confie, no mínimo, na confiança e nas permissões descritas em Criar uma função de serviço para CodeDeploy. Para obter o ARN da função de serviço, consulte Obter ARN da função de serviço (console).
4. Em Tipo de implantação, escolha In-place.
5. Em Environment configuration (Configuração do ambiente), selecione Amazon EC2 Instances (Instâncias do Amazon EC2). Escolha Name (Nome) no campo Key (Chave) e, no campo Value (Valor) informe **`MyCodePipelineDemo`**.

> **Important**
>
> É necessário selecionar o mesmo valor para a chave Nome atribuída à instância do EC2 quando criada. Se você marcou instâncias com algo diferente de **`MyCodePipelineDemo`**, certifique-se de usar a tag aqui.

6. Em Configuração do agente com o AWS Systems Manager, escolha Agora e agende atualizações. Isso vai instalar o agente na instância. A instância do Windows já está configurada com o agente SSM e agora será atualizada com o CodeDeploy agente.
7. Em Configurações da implantação, selecione `CodeDeployDefault.OneAtaTime`.
8. Em Balanceador de carga, verifique se a caixa Habilitar balanceamento de carga não está selecionada. Você não precisa configurar um load balancer ou escolher um grupo de destino para este exemplo. Depois de desmarcar a caixa de seleção, as opções do balanceador de carga não são exibidas.
9. Na seção Avançado deixe os padrões.
10. Selecione Criar grupo de implantação.

Etapa 4: crie seu primeiro funil no CodePipeline

Nesta parte do tutorial, você vai criar o pipeline. O exemplo executa automaticamente no pipeline.

Para criar um processo de liberação CodePipeline automatizado

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyFirstPipeline`**.

> **Note**
>
> Se você escolher outro nome para o pipeline, certifique-se de usar esse nome em vez de **`MyFirstPipeline`** no restante deste tutorial. Depois de criar um pipeline, não é possível alterar o nome dele. Os nomes de pipelines estão sujeitos à algumas limitações. Para obter mais informações, consulte Cotas em AWS CodePipeline.

5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.
6. Em Perfil de serviço, faça um dos seguintes procedimentos:
   - Escolha Nova função de serviço para permitir CodePipeline a criação de uma nova função de serviço no IAM.
   - Escolha Existing service role (Função de serviço existente) para usar uma função de serviço já criada no IAM. Em Role name (Nome da função), selecione a função de serviço na lista.
7. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
8. Em Etapa 3: Adicionar etapa de origem, em Provedor de origem, escolha Amazon S3. Em Bucket, insira o nome do bucket do S3 que você criou em Etapa 1: criar um bucket de origem do S3 para a aplicação. Em S3 object key (Chave do objeto do S3), insira a chave do objeto com ou sem um caminho de arquivo, e lembre-se de incluir a extensão do arquivo. Por exemplo, para `SampleApp_Windows.zip`, insira o nome do arquivo de exemplo como mostrado neste exemplo:

   ```
   SampleApp_Windows.zip
   ```

   Escolha Próxima etapa.

   Em Change detection options (Alterar opções de detecção), deixe os valores padrão. Isso permite CodePipeline usar o Amazon CloudWatch Events para detectar alterações em seu bucket de origem.

   Escolha Próximo.
9. Em Etapa 4: Adicionar etapa de compilação, escolha Ignorar etapa de compilação e aceite a mensagem de aviso escolhendo Ignorar novamente. Escolha Próximo.

10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

11. Na Etapa 6: Adicionar estágio de implantação, em Provedor de implantação, escolha CodeDeploy . O padrão do campo Região é o mesmo do seu Região da AWS funil. Em Application name (Nome do aplicativo), insira `MyDemoApplication` ou selecione o botão Refresh (Atualizar) e selecione o nome do aplicativo na lista. Em Deployment group (Grupo de implantação), insira **`MyDemoDeploymentGroup`** ou selecione-o na lista e selecione Next (Próximo).

    > **Note**
    >
    > O nome Implantação é o nome padrão dado ao estágio criado em Step 4: Add deploy stage (Etapa 4: adicionar estágio de implantação), assim como Origem é o nome dado ao primeiro estágio do pipeline.

12. Na Etapa 7: Revisar, revise as informações e escolha Criar funil.
13. O pipeline começa a ser executado. Você pode visualizar as mensagens de progresso, sucesso e falha à medida que a CodePipeline amostra implanta uma página da web em cada uma das instâncias do Amazon EC2 na implantação. CodeDeploy

Parabéns! Você acabou de criar um funil simples em CodePipeline. O pipeline tem dois estágios:

- Um estágio de origem chamado Source (Origem), que detecta as alterações no aplicativo de exemplo com versionamento armazenado no bucket do S3 e obtém essas alterações para o pipeline.
- Um estágio Deploy (Implantar) que implanta essas alterações nas instâncias do EC2 com o CodeDeploy.

Agora, verifique os resultados.

Como verificar se seu pipeline foi executado com êxito

1. Visualize o progresso inicial do pipeline. O status de cada estágio muda de No executions yet (Ainda não executado) para In Progress (Em andamento) e, então, para Succeeded (Bem-

sucedido) ou Failed (Falhou). O pipeline deve concluir a primeira execução dentro de alguns minutos.

2. Após a exibição do status Com êxito para o status da ação, na área do status da etapa Implantar, selecione Detalhes. Isso abre o CodeDeploy console.
3. Na guia Grupo de implantação, em Eventos de ciclo de vida da implantação, selecione o ID. Isso abre o console do EC2.
4. Na guia Description (Descrição), em Public DNS (DNS público), copie o endereço e cole-o na barra de endereços de seu navegador da Web. Visualize a página de índice para o aplicativo de exemplo que você carregou para o bucket do S3.

   A página da web exibe a aplicação de exemplo que você baixou para o bucket do S3.

Para obter mais informações sobre os estágios, as ações e o funcionamento dos pipelines, consulte CodePipeline conceitos.

(Opcional) Etapa 5: Adicionar outra etapa ao pipeline

Agora, adicione outro estágio ao pipeline para implantar dos servidores de preparação para os servidores de produção usando a CodeDeploy. Primeiro, você cria outro grupo de implantação no CodePipelineDemoApplication in CodeDeploy. Em seguida, adicione um estágio que inclui uma ação que usa esse grupo de implantação. Para adicionar outro estágio, use o CodePipeline console ou o AWS CLI para recuperar e editar manualmente a estrutura do pipeline em um arquivo JSON e, em seguida, execute o update-pipeline comando para atualizar o pipeline com suas alterações.

Tópicos

- Crie um segundo grupo de implantação no CodeDeploy
- Adicionar o grupo de implantação como outro estágio ao pipeline

Crie um segundo grupo de implantação no CodeDeploy

> **Note**
>
> Nesta parte do tutorial, você vai criar um segundo grupo de implantação, mas o implantará nas mesmas instâncias do Amazon EC2 usadas anteriormente. Isso é apenas para fins de demonstração. Ele foi projetado propositadamente para não mostrar como os erros são exibidos em. CodePipeline

Para criar um segundo grupo de implantação no CodeDeploy

1. Abra o CodeDeploy console em https://console.aws.amazon.com/codedeploy.
2. Selecione Applications (Aplicativos) e, na lista de aplicativos, selecione `MyDemoApplication`.
3. Selecione a guia Deployment groups (Grupos de implantação), depois escolha Create deployment group (Criar grupo de implantação).
4. Na página Create deployment group (Criar grupo de implantação), em Deployment group name (Nome do grupo de implantação), insira um nome para o segundo grupo de implantação (por exemplo, **`CodePipelineProductionFleet`**).
5. Em Função de serviço, escolha a mesma função de CodeDeploy serviço que você usou para a implantação inicial (não a função CodePipeline de serviço).
6. Em Tipo de implantação, escolha In-place.
7. Em Environment configuration (Configuração do ambiente), selecione Amazon EC2 Instances (Instâncias do Amazon EC2). Escolha Nome na caixa Chave e, na caixa Valor, escolha `MyCodePipelineDemo` na lista. Deixe a configuração padrão para Deployment settings (Configurações da implantação).
8. Em Deployment configuration (Configuração de implantação), selecione `CodeDeployDefault.OneAtaTime`.
9. Em Load Balancer, desmarque Enable load balancing (Habilitar balanceamento de carga).
10. Selecione Criar grupo de implantação.

Adicionar o grupo de implantação como outro estágio ao pipeline

Agora que você possui outro grupo de implantação, poderá adicionar um estágio que usa esse grupo para implantar nas mesmas instâncias do EC2 usadas anteriormente. Você pode usar o CodePipeline console ou o AWS CLI para adicionar esse estágio.

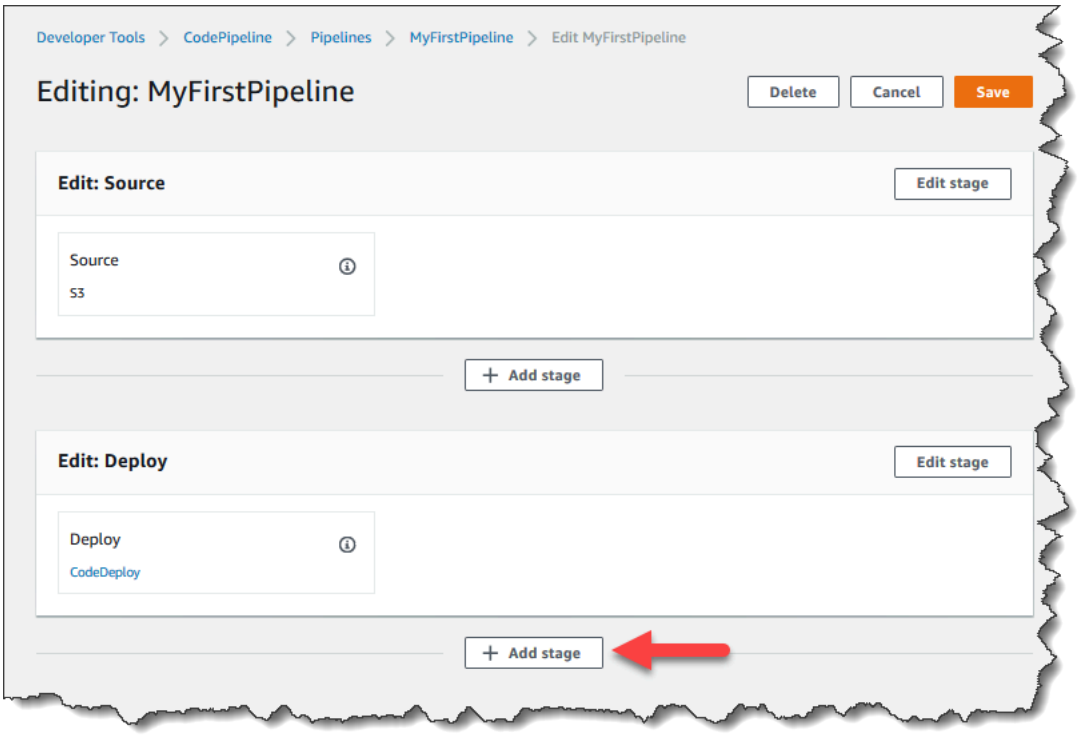
Tópicos

- Criar um terceiro estágio (console)
- Criar um terceiro estágio (CLI)

Criar um terceiro estágio (console)

Você pode usar o CodePipeline console para adicionar um novo estágio que usa o novo grupo de implantação. Como esse grupo de implantação está implantando nas instâncias do EC2 que você já usou, a ação de implantação neste estágio falhará.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Em Name (Nome), selecione o nome do pipeline que você criou, MyFirstPipeline.
3. Na página de detalhes do pipeline, selecione Editar.
4. Na página Edit (Editar), escolha + Add stage (+ Adicionar estágio) para adicionar um estágio logo depois do estágio de implantação.



5. Em Add stage (Adicionar estágio), em Stage name (Nome do estágio), insira **`Production`**. Escolha Add stage (Adicionar estágio).
6. No novo estágio, escolha + Add action group (+ Adicionar grupo de ação).
7. Em Edit action (Editar ação), em Action name (Nome da ação), insira **`Deploy-Second-Deployment`**. Em Provedor de ações, em Implantar, escolha CodeDeploy.
8. Na CodeDeploy seção, em Nome `MyDemoApplication` do aplicativo, escolha na lista suspensa, como você fez quando criou o pipeline. Em Deployment group (Grupo de implantação), selecione o grupo de implantação que você acabou de criar,

`CodePipelineProductionFleet`. Em Input artifacts (Artefatos de entrada), escolha o artefato de entrada da ação de origem. Escolha Salvar.

9. Na página Edit (Editar), escolha Save (Salvar). Em Save pipeline changes (Salvar alterações de pipeline), escolha Save (Salvar).
10. Embora o novo estágio tenha sido adicionado ao seu pipeline, um status de Ainda não executado é exibido, pois nenhuma alteração acionou outra execução do pipeline. Você deve reexecutar manualmente a última revisão para ver como o pipeline editado é executado. Na página de detalhes do pipeline, selecione Lançar alteração e, depois, Lançar quando solicitado. Essa ação executa a revisão mais recente disponível em cada local de origem especificado em uma ação de origem do pipeline.

    Como alternativa, para usar o AWS CLI para executar novamente o pipeline, a partir de um terminal em sua máquina Linux, macOS ou Unix local, ou de um prompt de comando em sua máquina Windows local, execute o start-pipeline-execution comando especificando o nome do pipeline. Isso executa o aplicativo em seu bucket de origem por meio do pipeline pela segunda vez.

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

    Esse comando retorna um `pipelineExecutionId` objeto.
11. Retorne ao CodePipeline console e, na lista de pipelines, opte por MyFirstPipeline abrir a página de visualização.

    O pipeline mostra três estágios e o estado do artefato em execução através destes três estágios. Pode levar até cinco minutos para que o pipeline execute todas os estágios. Você vê que a implantação foi bem-sucedida nos dois primeiros estágios, assim como antes, mas o estágio de produção mostra que a Deploy-Second-Deployment ação falhou.
12. Na Deploy-Second-Deployment ação, escolha Detalhes. Você é redirecionado para a página da implantação do CodeDeploy. Nesse caso, a falha resulta da implantação do primeiro grupo de instâncias em todas as instâncias do EC2, sem deixar qualquer instância para o segundo grupo de implantação.

    > **Note**
    >
    > Esta falha é por projeto, para demonstrar o que acontece quando ocorre uma falha em um estágio do pipeline.

Criar um terceiro estágio (CLI)

Embora usar o AWS CLI para adicionar um estágio ao seu pipeline seja mais complexo do que usar o console, ele fornece mais visibilidade da estrutura do pipeline.

Para criar um terceiro estágio para seu pipeline

1. Abra uma sessão de terminal na máquina local do Linux, do macOS ou do Unix ou um prompt de comando na máquina local do Windows e execute o comando get-pipeline para exibir a estrutura de pipeline que você acabou de criar. Em **MyFirstPipeline**, digite o seguinte comando:

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

   Esse comando retorna a estrutura do MyFirstPipeline. A primeira parte da saída deve ser semelhante ao seguinte:

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

   A última parte da saída inclui os metadados do pipeline e deve ser semelhante ao seguinte:

   ```
       ...
           ],
           "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 e cole essa estrutura em um editor de texto plano e salve o arquivo como **`pipeline.json`**. Para sua conveniência, salve este arquivo no mesmo diretório onde você executa os comandos aws codepipeline.

> **Note**
>
> Você pode canalizar o JSON diretamente em um arquivo com o comando get-pipeline da seguinte forma:
>
> ```
> aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
> ```

3. Copie a seção da etapa Implantar e cole-a após as duas primeiras etapas. Como é uma etapa de implantação, assim como a etapa Implantar, você a usará como um modelo para a terceira etapa.

4. Altere o nome do estágio e os detalhes do grupo de implantação.

   O exemplo a seguir mostra o JSON que será adicionado ao arquivo pipeline.json após a etapa de implantação. Edite os elementos enfatizados com novos valores. Lembre-se de incluir uma vírgula para separar as definições das etapas Implantar e Produção.

```
,
{
    "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. Se você estiver trabalhando com a estrutura do pipeline recuperada por meio do comando get-pipeline, é necessário remover as linhas `metadata` do arquivo JSON. Caso contrário, o comando update-pipeline não é capaz de utilizá-la. Remova as linhas `"metadata": { }`, `"created"`, `"pipelineARN"` e os campos `"updated"`.

   Por exemplo, remova as seguintes linhas da estrutura:

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

   Salve o arquivo.

6. Execute o comando update-pipeline especificando o arquivo JSON do pipeline, de modo semelhante ao seguinte:

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

   Este comando retorna toda a estrutura do pipeline atualizado.

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

7. Execute o comando start-pipeline-execution, especificando o nome do pipeline. Isso executa o aplicativo em seu bucket de origem por meio do pipeline pela segunda vez.

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

   Esse comando retorna um `pipelineExecutionId` objeto.

8. Abra o CodePipeline console e escolha MyFirstPipeline na lista de pipelines.

   O pipeline mostra três estágios e o estado do artefato em execução através destes três estágios. Pode levar até cinco minutos para que o pipeline execute todas os estágios. Embora a implantação seja bem-sucedida nos dois primeiros estágios, assim como antes, o estágio de produção mostra que a Deploy-Second-Deployment ação falhou.

9. Na Deploy-Second-Deployment ação, escolha Detalhes para ver os detalhes da falha. Você será redirecionado para a página de detalhes da CodeDeploy implantação. Nesse caso, a falha resulta da implantação do primeiro grupo de instâncias em todas as instâncias do EC2, sem deixar qualquer instância para o segundo grupo de implantação.

   > **Note**
   >
   > Esta falha é por projeto, para demonstrar o que acontece quando ocorre uma falha em um estágio do pipeline.

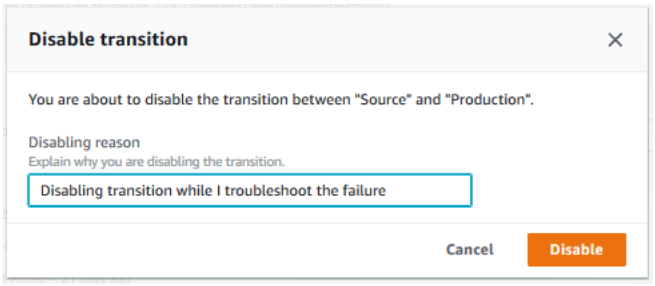
(Opcional) Etapa 6: desativar e ativar as transições entre os estágios no CodePipeline

Você pode habilitar ou desabilitar a transição entre estágios em um pipeline. Desabilitar a transição entre os estágios permite que você controle manualmente as transições entre um estágio e outro. Por exemplo, você talvez queira executar os primeiros dois estágios de um pipeline, mas desativar as transições para o terceiro estágio até que você esteja pronto para implantar para produção ou ao solucionar um problema ou uma falha com esse estágio.

Para desativar e ativar as transições entre os estágios em um pipeline CodePipeline

1. Abra o CodePipeline console e escolha MyFirstPipeline na lista de pipelines.
2. Na página de detalhes do pipeline, selecione o botão Desativar transição entre a segunda etapa, (Implantar) e a terceira etapa adicionada na seção anterior, (Produção).
3. Em Disable transition (Desabilitar transição), insira um motivo para desabilitar a transição entre os estágios e depois selecione Disable (Desabilitar).

   A seta entre estágios exibe um ícone e uma alteração de cor e o botão Enable transition (Permitir transição) é exibido.



4. Faça o upload de sua amostra novamente para o bucket do S3. Como o bucket é versionado, essa mudança inicia o pipeline.
5. Volte para a página de detalhes de seu pipeline e veja o status dos estágios. A visualização do pipeline é alterada para mostrar o andamento e o sucesso nos primeiros dois estágios, mas nenhuma alteração ocorre no terceiro estágio. Esse processo pode levar alguns minutos.
6. Permita a transição selecionando o botão Enable transition (Permitir transição) entre os dois estágios. Na caixa de diálogo Permitir transição, selecione Permitir. O estágio começa a ser executado em alguns minutos e tenta processar o artefato que foi executado pelos dois primeiros estágios de pipeline.

> **Note**
>
> Se você quiser que esse terceiro estágio seja bem-sucedido, edite o grupo de CodePipelineProductionFleet implantação antes de ativar a transição e especifique um conjunto diferente de instâncias do EC2 em que o aplicativo está implantado. Para mais informações sobre como fazer isso, consulte Alterar configurações do grupo de implantação. Se você criar mais instâncias do EC2, poderá incorrer em custos adicionais.

Etapa 7: Limpar os recursos

Você pode usar alguns dos recursos criados neste tutorial para o Tutorial: Criar um pipeline de quatro estágios. Por exemplo, você pode reutilizar o CodeDeploy aplicativo e a implantação. Você pode configurar uma ação de criação com um provedor CodeBuild, como, por exemplo, um serviço de compilação totalmente gerenciado na nuvem. Você também pode configurar uma ação de criação que use um provedor com um servidor ou um sistema de criação, como o Jenkins.

No entanto, após finalizar este e outros tutoriais, você deverá excluir o pipeline e os recursos usados para não ser cobrado pelo uso contínuo desses recursos. Primeiro, exclua o pipeline, depois o CodeDeploy aplicativo e suas instâncias associadas do Amazon EC2 e, finalmente, o bucket do S3.

Para limpar os recursos usados neste tutorial

1. Para limpar seus CodePipeline recursos, siga as instruções em Excluir um funil em AWS CodePipeline.
2. Para limpar seus CodeDeploy recursos, siga as instruções em Para limpar recursos (console).
3. Para excluir o bucket do S3, siga as instruções em Excluir ou esvaziar um bucket. Se você não pretende criar mais pipelines, exclua o bucket do S3 criado para o armazenamento de seus artefatos do pipeline. Para mais informações sobre esse bucket, consulte CodePipeline conceitos.

Tutorial: criar um pipeline simples (CodeCommit repositório)

Neste tutorial, você usa CodePipeline para implantar o código mantido em um CodeCommit repositório em uma única instância do Amazon EC2. Seu funil é acionado quando você envia uma alteração para o CodeCommit repositório. O pipeline implanta suas alterações em uma instância do Amazon EC2 usando CodeDeploy como serviço de implantação.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

O pipeline tem dois estágios:

- Um estágio de origem (Fonte) para sua ação CodeCommit de origem.
- Um estágio de implantação (Implantação) para sua ação CodeDeploy de implantação.

A maneira mais fácil de começar AWS CodePipeline é usar o assistente Create Pipeline no CodePipeline console.

> **Note**
>
> Antes de começar, certifique-se de ter configurado seu cliente Git para trabalhar com CodeCommit ele. Para obter instruções, consulte Configuração para o CodeCommit.

Etapa 1: criar um CodeCommit repositório

Primeiro, você cria um repositório no CodeCommit. Seu pipeline obtém o código-fonte desse repositório quando for executado. Você também cria um repositório local onde mantém e atualiza o código antes de enviá-lo ao CodeCommit repositório.

Para criar um CodeCommit repositório

1. Abra o CodeCommit console em https://console.aws.amazon.com/codecommit/.
2. No seletor de região, escolha Região da AWS onde você deseja criar o repositório e o pipeline. Para obter mais informações, consulte Regiões da AWS e endpoints.
3. Na página Repositories (Repositórios), selecione Create repository (Criar repositório).
4. Na página Create repository (Criar repositório), em Repository name (Nome do repositório), insira um nome para o repositório (por exemplo **`MyDemoRepo`**).
5. Escolha Criar.

> **Note**
>
> As etapas restantes neste tutorial são usadas **`MyDemoRepo`** para o nome do seu CodeCommit repositório. Se você escolher um nome diferente, certifique-se de usá-lo durante todo este tutorial.

Como configurar um repositório local

Nesta etapa, você configurará um repositório local para se conectar ao repositório remoto do CodeCommit.

> **Note**
>
> Não é necessário configurar um repositório local. Você também pode usar o console do para fazer upload dos arquivos, conforme descrito em Etapa 2: adicionar código de amostra ao seu CodeCommit repositório.

1. Com o novo repositório aberto no console, escolha Clone URL (Clonar URL) no canto superior direito da página e, depois, escolha Clone SSH (Clonar SSH). O endereço para clonar o repositório Git é copiado na área de transferência.
2. No terminal ou na linha de comando, navegue até um diretório local onde você deseja que seu repositório local seja armazenado. Neste tutorial, usamos `/tmp`.
3. Execute o comando a seguir para clonar o repositório, substituindo o endereço SSH pelo que você copiou na etapa anterior. Esse comando cria um diretório chamado `MyDemoRepo`. Copie um aplicativo de exemplo nesse diretório.

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

Etapa 2: adicionar código de amostra ao seu CodeCommit repositório

Nesta etapa, você baixa o código de um aplicativo de amostra que foi criado para uma CodeDeploy amostra passo a passo e o adiciona ao seu CodeCommit repositório.

1. Depois, faça download de um exemplo e salve-o em uma pasta ou um diretório no computador local.

   a. Escolha uma das opções a seguir. Selecione `SampleApp_Linux.zip` se deseja seguir as etapas deste tutorial para instâncias do Linux.

   - Se você quiser implantar em instâncias Amazon Linux usando CodeDeploy, baixe o aplicativo de amostra aqui: SampleApp _ Linux.zip.
   - Se você quiser implantar em instâncias do Windows Server usando CodeDeploy, baixe o aplicativo de amostra aqui: SampleApp _ Windows.zip.

   O aplicativo de amostra contém os seguintes arquivos para implantação com CodeDeploy:

- `appspec.yml`— O arquivo de especificação do aplicativo (AppSpecarquivo) é um arquivo formatado em YAML usado pelo CodeDeploy para gerenciar uma implantação. Para obter mais informações sobre o AppSpec arquivo, consulte Referência CodeDeploy AppSpec do arquivo no Guia AWS CodeDeploy do usuário.
- `index.html`: o arquivo de índice contém a página inicial da aplicação de exemplo implantada.
- `LICENSE.txt`: o arquivo de licença contém informações de licença da aplicação de exemplo.
- Arquivos para scripts: a aplicação de exemplo usa scripts para gravar arquivos de texto em um local na instância. Um arquivo é gravado para cada um dos vários eventos do ciclo de vida da CodeDeploy implantação da seguinte forma:
  - Pasta `scripts` (somente exemplo do Linux): a pasta contém os seguintes scripts de shell para instalar dependências e iniciar e interromper a aplicação de exemplo para a implantação automatizada: `install_dependencies`, `start_server` e `stop_server`.
  - (Somente exemplo do Windows) `before-install.bat`: um script em lote para o evento de ciclo de vida de implantação `BeforeInstall`, que será executado para remover os arquivos antigos gravados durante implantações anteriores deste exemplo e criar um local na instância onde os novos arquivos serão gravados.

b. Faça download do arquivo compactado.

2. Descompacte os arquivos de SampleApp _ Linux.zip no diretório local que você criou anteriormente (por exemplo, `/tmp/MyDemoRepo` ou`c:\temp\MyDemoRepo`).

   Certifique-se de alocar os arquivos diretamente em seu repositório local. Não inclua uma pasta `SampleApp_Linux`. Na máquina Linux, macOS ou Unix local, por exemplo, a hierarquia de arquivos e diretórios deve ser semelhante a:

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

3. Para fazer upload dos arquivos para o repositório, use um dos métodos a seguir.

   a. Para usar o CodeCommit console para fazer upload de seus arquivos:

      i. Abra o CodeCommit console e escolha seu repositório na lista Repositórios.

      ii. Selecione Add file (Adicionar arquivo) e clique em Upload file (Carregar arquivo).

      iii. Selecione Choose file (Escolher arquivo) e procure o arquivo. Para adicionar um arquivo em uma pasta, selecione Criar arquivo e, depois, insira o nome da pasta com o nome do arquivo, como `scripts/install_dependencies`. Cole o conteúdo do arquivo no novo arquivo.

      Informe seu nome de usuário e endereço de e-mail para confirmar a alteração.

      Escolha Commit changes (Confirmar alterações).

      iv. Repita esta etapa para cada arquivo.

      O conteúdo do repositório deve ter a seguinte aparência:

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

   b. Para usar os comandos do git para fazer upload dos arquivos:

      i. Altere diretórios para o seu repositório local:

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

      ii. Execute o seguinte comando para organizar todos os seus arquivos de uma só vez:

```
git add -A
```

      iii. Execute o seguinte comando para confirmar os arquivos com uma mensagem de confirmação:

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

iv. Execute o seguinte comando para enviar os arquivos de seu repositório local para seu repositório do CodeCommit:

```
git push
```

4. Os arquivos que você baixou e adicionou ao seu repositório local agora foram adicionados à `main` ramificação em seu CodeCommit `MyDemoRepo` repositório e estão prontos para serem incluídos em um pipeline.

Etapa 3: Crie uma instância Linux do Amazon EC2 e instale o agente CodeDeploy

Nesta etapa, você vai criar a instância do Amazon EC2 na qual implantará uma aplicação de exemplo. Como parte desse processo, crie uma função de instância que permita a instalação e o gerenciamento do CodeDeploy agente na instância. O CodeDeploy agente é um pacote de software que permite que uma instância seja usada em CodeDeploy implantações. Você também anexa políticas que permitem que a instância busque arquivos que o CodeDeploy agente usa para implantar seu aplicativo e permitir que a instância seja gerenciada pelo SSM.

Como criar uma função de instância

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel do console, escolha Roles (Funções).
3. Selecione Criar perfil.
4. Em Selecionar tipo de entidade confiável, selecione AWS service (Serviço da AWS). Em Escolha um caso de uso, selecione EC2. Em Select your use case (Selecione seu caso de uso), escolha EC2. Escolha Próximo: Permissões.
5. Procure e selecione a política chamada **`AmazonEC2RoleforAWSCodeDeploy`**.
6. Procure e selecione a política chamada **`AmazonSSMManagedInstanceCore`**. Escolha Próximo: tags.
7. Selecione Próximo: revisar. Forneça um nome para a função (por exemplo, **`EC2InstanceRole`**).

> **Note**
>
> Anote o nome da função para a próxima etapa. Escolha essa função ao criar a instância.

Selecione Criar perfil.

Como iniciar uma instância

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Na navegação lateral, escolha Instâncias e selecione Executar instâncias na parte superior da página.
3. Em Nome, insira **`MyCodePipelineDemo`**. Isso atribui à instância uma tag Chave de **`Name`** e uma tag Valor de **`MyCodePipelineDemo`**. Posteriormente, você cria um CodeDeploy aplicativo que implanta o aplicativo de amostra nessa instância. CodeDeployseleciona instâncias a serem implantadas com base nas tags.
4. Em Imagens de aplicativos e sistemas operacionais (Amazon Machine Image), localize a opção Amazon Linux AMI com o AWS logotipo e verifique se ela está selecionada. (Essa AMI é descrita como AMI do Amazon Linux 2 (HVM) e é identificada como “Elegível para o nível gratuito”.)
5. Em Tipo de instância, selecione o tipo `t2.micro` elegível para o nível gratuito como configuração de hardware para a instância.
6. Na seção Par de chaves (login), selecione um par de chaves ou crie um.

   Também é possível selecionar Prosseguir sem um par de chaves.

   > **Note**
   >
   > Para os fins deste tutorial, é possível prosseguir sem um par de chaves. Para usar o SSH para se conectar às instâncias, crie ou use um par de chaves.

7. Em Configurações de rede, faça o seguinte:

   Em IP Auto-assign público, verifique se o status é Ativado.

   Selecione o grupo de segurança criado, escolha HTTP e, em Tipo de origem, escolha Meu IP.

8. Expanda Advanced details (Detalhes avançados). Em Perfil de instância do IAM, selecione o perfil do IAM criado no procedimento anterior (por exemplo, **`EC2InstanceRole`**).
9. Em Resumo, em Número de instâncias, insira `1`.
10. Escolha Iniciar instância.
11. É possível visualizar o status da ativação na página Instâncias. Ao executar uma instância, seu estado inicial é `pending`. Após o início da instância, seu estado muda para `running` e ela recebe um nome DNS público. (Se a coluna DNS público não for exibida, escolha o Show/Hide ícone e selecione DNS público.)

Etapa 4: criar um aplicativo no CodeDeploy

Em CodeDeploy, um aplicativo é um recurso que contém o aplicativo de software que você deseja implantar. Posteriormente, você usa esse aplicativo com CodePipeline para automatizar as implantações do aplicativo de amostra em sua instância do Amazon EC2.

Primeiro, você cria uma função que permite CodeDeploy realizar implantações. Depois, crie um aplicativo do CodeDeploy .

Para criar uma função CodeDeploy de serviço

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel do console, escolha Roles (Funções).
3. Selecione Criar perfil.
4. Em Selecionar entidade confiável, escolha AWS service (Serviço da AWS). Em Use case (Caso de uso), escolha CodeDeploy. CodeDeployEscolha entre as opções listadas. Escolha Próximo. A política gerenciada `AWSCodeDeployRole` já está anexada à função.
5. Escolha Próximo.
6. Insira um nome para a função (por exemplo, **`CodeDeployRole`**) e escolha Create role (Criar função).

Para criar um aplicativo no CodeDeploy

1. Abra o CodeDeploy console em https://console.aws.amazon.com/codedeploy.
2. Se a página Aplicações não for exibida, no menu, selecione Aplicações.
3. Escolha Criar aplicativo.

4. Em Nome do aplicativo, insira **`MyDemoApplication`**.
5. Em Compute Platform, escolha EC2/On-premises.
6. Selecione Criar aplicativo.

Para criar um grupo de implantação no CodeDeploy

Um grupo de implantação é um recurso que define configurações relacionadas à implantação, como em quais instâncias implantar e com que rapidez implantá-las.

1. Na página que mostra o aplicativo, selecione Create deployment group (Criar grupo de implantação).
2. Em Nome do grupo de implantação, insira **`MyDemoDeploymentGroup`**.
3. Em Perfil de serviço, selecione o perfil de serviço criado anteriormente (por exemplo, **`arn:aws:iam::`*`account_ID`*`:role/CodeDeployRole`**).
4. Em Tipo de implantação, escolha In-place.
5. Em Environment configuration (Configuração do ambiente), selecione Amazon EC2 Instances (Instâncias do Amazon EC2). No campo, Chave, insira **`Name`**. No campo Valor, insira o nome usado para marcar a instância (por exemplo, **`MyCodePipelineDemo`**).
6. Em Configuração do agente com o AWS Systems Manager, escolha Agora e agende atualizações. Isso vai instalar o agente na instância. A instância Linux já está configurada com o agente SSM e agora será atualizada com o CodeDeploy agente.
7. Em Deployment configuration (Configuração de implantação), selecione `CodeDeployDefault.OneAtaTime`.
8. Em Balanceador de carga, verifique se Habilitar balanceamento de carga não está selecionado. Você não precisa configurar um load balancer ou escolher um grupo de destino para este exemplo.
9. Selecione Criar grupo de implantação.

Etapa 5: crie seu primeiro funil no CodePipeline

Agora, você está pronto para criar e executar seu primeiro pipeline. Nesta etapa, você cria um pipeline que é executado automaticamente quando o código é enviado ao seu CodeCommit repositório.

Para criar um CodePipeline funil

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyFirstPipeline`**.
5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.
6. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.
7. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
8. Em Etapa 3: Adicionar etapa de origem, em Provedor de origem, escolha CodeCommit. Em Nome do repositório, escolha o nome do CodeCommit repositório em que você criou. Etapa 1: criar um CodeCommit repositório Em Branch name (Nome da ramificação), escolha `main` e, depois, selecione Next step (Próxima etapa).

   Depois de selecionar o nome e a ramificação do repositório, uma mensagem exibe a regra do Amazon CloudWatch Events a ser criada para esse pipeline.

   Em Change detection options (Alterar opções de detecção), deixe os valores padrão. Isso permite CodePipeline usar o Amazon CloudWatch Events para detectar alterações em seu repositório de origem.

   Escolha Próximo.
9. Em Etapa 4: Adicionar etapa de compilação, escolha Ignorar etapa de compilação e aceite a mensagem de aviso escolhendo Ignorar novamente. Escolha Próximo.

> **Note**
>
> Neste tutorial, você está implantando um código que não requer um serviço de compilação, portanto, ignore esta etapa. No entanto, se o código-fonte precisar ser compilado antes de ser implantado em instâncias, você poderá configurar CodeBuild nesta etapa.

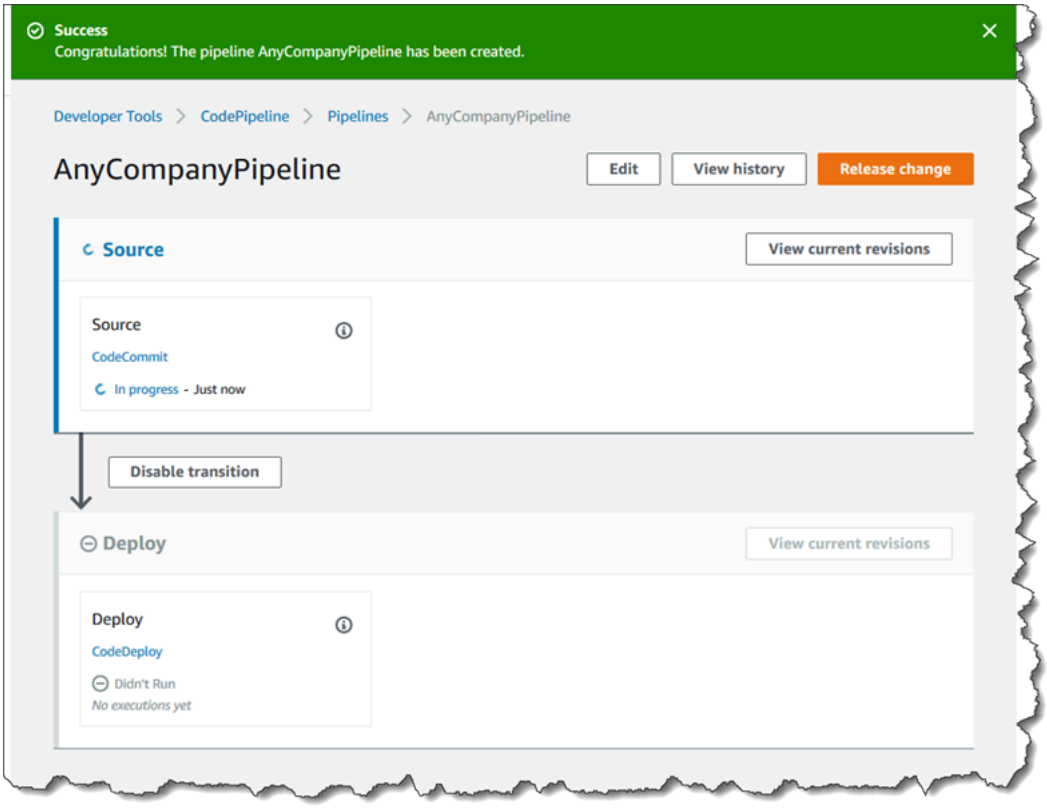
10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

11. Na Etapa 6: Adicionar estágio de implantação, em Provedor de implantação, escolha CodeDeploy. Em Application name (Nome do aplicativo), escolha **`MyDemoApplication`**. Em Deployment group (Grupo de implantação), escolha **`MyDemoDeploymentGroup`** e, depois, selecione Next step (Próxima etapa).

12. Na Etapa 7: Revisar, revise as informações e escolha Criar funil.

13. O pipeline começa a ser executado depois de ser criado. Ele baixa o código do seu CodeCommit repositório e cria uma CodeDeploy implantação na sua instância do EC2. Você pode ver as mensagens de progresso, sucesso e falha à medida que a CodePipeline amostra implanta a página da web na instância do Amazon EC2 na implantação. CodeDeploy



Parabéns! Você acabou de criar um funil simples em CodePipeline.

Depois, você verifica os resultados.

Para verificar se o seu pipeline foi executado com êxito

1. Visualize o progresso inicial do pipeline. O status de cada estágio muda de No executions yet (Ainda não executado) para In Progress (Em andamento) e, então, para Succeeded (Bem-sucedido) ou Failed (Falhou). O pipeline deve concluir a primeira execução dentro de alguns minutos.
2. Depois que Suceeded for exibido para o status do pipeline, na área de status do estágio Implantação, escolha CodeDeploy. Isso abre o CodeDeploy console. Se Sucedido não for exibido, consulteSolução de problemas CodePipeline.
3. Na guia Implantações, selecione o ID de implantação. Na página da implantação, em Eventos de ciclo de vida da implantação, selecione o ID da instância. Isso abre o console do EC2.
4. Na guia Description (Descrição), em Public DNS (DNS público), copie o endereço (por exemplo, `ec2-192-0-2-1.us-west-2.compute.amazonaws.com`) e cole-o na barra de endereços de seu navegador da Web.

A página da web é exibida para o aplicativo de amostra que você baixou e enviou para o seu CodeCommit repositório.

Para obter mais informações sobre os estágios, as ações e o funcionamento dos pipelines, consulte CodePipeline conceitos.

Etapa 6: modificar o código em seu CodeCommit repositório

O pipeline está configurado para ser executado sempre que forem feitas alterações de código no repositório do CodeCommit. Nesta etapa, você faz alterações no arquivo HTML que faz parte do CodeDeploy aplicativo de amostra no CodeCommit repositório. Quando essas alterações são enviadas, o pipeline é executado novamente, e as alterações feitas ficam visíveis no endereço da web acessado anteriormente.

1. Altere diretórios para o seu repositório local:

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

2. Use um editor de texto para modificar o arquivo `index.html`:

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

3. Revise o conteúdo do arquivo `index.html` para alterar a cor do plano de fundo e parte do texto na página da web e depois salve o arquivo.

```
<!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. Confirme e envie suas alterações para o seu CodeCommit repositório executando os seguintes comandos, um de cada vez:

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

```
git push
```

Como verificar se seu pipeline foi executado com êxito

1. Visualize o progresso inicial do pipeline. O status de cada estágio muda de No executions yet (Ainda não executado) para In Progress (Em andamento) e, então, para Succeeded (Bem-sucedido) ou Failed (Falhou). A execução do pipeline deve ser concluída em alguns minutos.

2. Depois que Succeeded (Bem-sucedido) for exibido para o status da ação, atualize a página de demonstração acessada anteriormente no navegador.

   A página da web atualizada é exibida.

Etapa 7: Limpar os recursos

É possível usar alguns dos recursos criados neste tutorial para outros tutoriais neste guia. Por exemplo, você pode reutilizar o CodeDeploy aplicativo e a implantação. No entanto, depois de finalizar este e outros tutoriais, você deverá excluir o pipeline e os recursos usados para não ser cobrado pelo uso contínuo desses recursos. Primeiro, exclua o pipeline, depois o CodeDeploy aplicativo e sua instância associada do Amazon EC2 e, finalmente, o CodeCommit repositório.

Para limpar os recursos usados neste tutorial

1. Para limpar seus CodePipeline recursos, siga as instruções em Excluir um funil em AWS CodePipeline.
2. Para limpar seus CodeDeploy recursos, siga as instruções em Recursos passo a passo de implantação do Clean Up.
3. Para excluir o CodeCommit repositório, siga as instruções em Excluir um CodeCommit repositório.

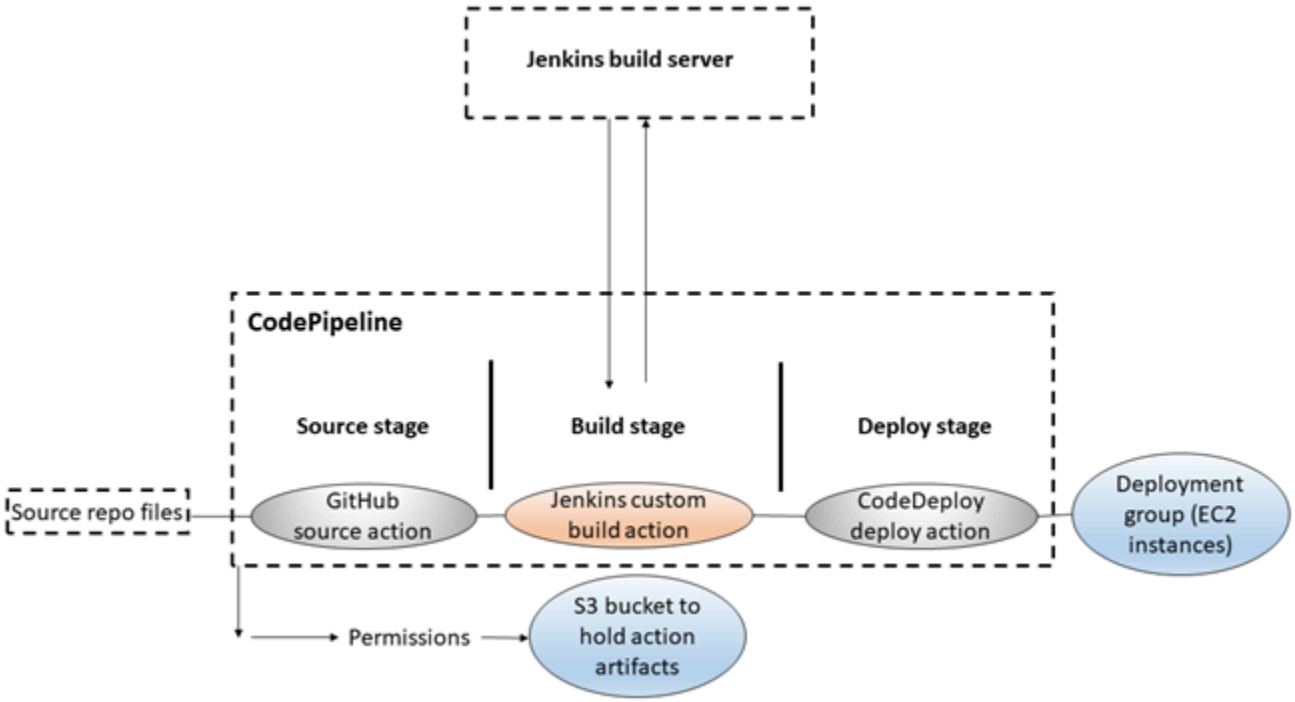
Etapa 8: Outras fontes de leitura

Saiba mais sobre como CodePipeline funciona:

- Para obter mais informações sobre os estágios, as ações e o funcionamento dos pipelines, consulte CodePipeline conceitos.
- Para obter informações sobre as ações que você pode realizar usando CodePipeline, consulteIntegrações com tipos de CodePipeline ação.
- Experimente este tutorial mais avançado, Tutorial: Criar um pipeline de quatro estágios. Ele cria um pipeline de vários estágios incluindo uma etapa que cria o código antes de ser implantado.

Tutorial: Criar um pipeline de quatro estágios

Agora que criou o primeiro pipeline em Tutorial: Criar um pipeline simples (bucket do S3) ou Tutorial: criar um pipeline simples (CodeCommit repositório), você pode começar a criar pipelines mais complexos. Este tutorial orientará você na criação de um pipeline de quatro estágios que usa um GitHub repositório para sua fonte, um servidor de compilação Jenkins para criar o projeto e um CodeDeploy aplicativo para implantar o código criado em um servidor de teste. O diagrama a seguir mostra o pipeline inicial de três estágios.



Após a criação do pipeline, você o editará para adicionar um estágio com uma ação de teste para testar o código, também usando Jenkins.

Antes de criar esse pipeline, você deve configurar os recursos necessários. Por exemplo, se você quiser usar um GitHub repositório para seu código-fonte, deverá criar o repositório antes de adicioná-lo a um pipeline. Como parte da configuração, este tutorial orienta você a configurar o Jenkins em uma instância do EC2 para fins de demonstração.

> **Important**
>
> Muitas das ações que você adiciona ao seu funil neste procedimento envolvem AWS recursos que você precisa criar antes de criar o funil. AWS os recursos para suas ações de origem sempre devem ser criados na mesma AWS região em que você cria seu funil.

Por exemplo, se você criar seu funil na região Leste dos EUA (Ohio), seu CodeCommit repositório deverá estar na região Leste dos EUA (Ohio).
Você pode adicionar ações entre regiões ao criar seu funil. AWS os recursos para ações entre regiões devem estar na mesma AWS região em que você planeja executar a ação. Para obter mais informações, consulte Adicione uma ação entre regiões em CodePipeline.

Important

Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Antes de iniciar este tutorial, você já deve ter concluído os pré-requisitos gerais em Começando com CodePipeline.

Tópicos

- Etapa 1: Concluir os pré-requisitos
- Etapa 2: criar um funil no CodePipeline
- Etapa 3: Adicionar outro estágio ao pipeline
- Etapa 4: Limpar os recursos

Etapa 1: Concluir os pré-requisitos

Para integrar com o Jenkins, é AWS CodePipeline necessário instalar o CodePipeline Plugin for Jenkins em qualquer instância do Jenkins com a qual você queira usar. CodePipeline Você também deve configurar um usuário ou função do IAM dedicado para usar nas permissões entre seu projeto Jenkins e. CodePipeline A maneira mais fácil de integrar o Jenkins CodePipeline é instalar o Jenkins em uma instância do EC2 que usa uma função de instância do IAM que você criou para a integração com o Jenkins. Para a conexão dos links no pipeline das ações do Jenkins ocorrer com êxito, é necessário definir as configurações de proxy e firewall no servidor ou na instância do EC2 para permitir conexões de entrada na porta usada pelo projeto do Jenkins. Verifique se você configurou o Jenkins para autenticar usuários e aplicar o controle de acesso antes de permitir conexões nessas

portas (por exemplo, 443 e 8443 se você fixou Jenkins para usar somente conexões HTTPS, ou 80 e 8080 se você permite conexões HTTP). Para obter mais informações, consulte Como fixar Jenkins.

> **Note**
>
> Este tutorial usa uma amostra de código e configura as etapas de criação que convertem a amostra de Haml em HTML. Você pode baixar o código de amostra de código aberto do GitHub repositório seguindo as etapas em. Copiar ou clonar a amostra em um repositório GitHub Você precisará da amostra inteira em seu GitHub repositório, não apenas do arquivo.zip.
>
> Este tutorial também pressupõe que:
>
> - Você está familiarizado com a instalação e administração do Jenkins e com a criação de projetos do Jenkins.
> - Você instalou o Rake e o gem de Haml para Ruby no mesmo computador ou instância que hospeda o projeto do Jenkins.
> - Você definiu as variáveis do ambiente do sistema exigidas para que os comandos do Rake possam ser executados do terminal ou da linha de comando (por exemplo, em sistemas Windows, alterar a variável PATH para incluir o diretório em que o Rake está instalado).

Tópicos

- Copiar ou clonar a amostra em um repositório GitHub
- Criar um perfil do IAM a ser usado na integração do Jenkins
- Instale e configure o Jenkins e o CodePipeline plug-in para Jenkins

Copiar ou clonar a amostra em um repositório GitHub

Para clonar a amostra e enviá-la para um repositório GitHub

1. Baixe o código de amostra do GitHub repositório ou clone os repositórios em seu computador local. Há dois pacotes de amostra:

   - Se você for implantar a amostra em instâncias do Amazon Linux, do RHEL ou do Ubuntu Server, selecione codepipeline-jenkins-aws-codedeploy_linux.zip.
   - Se você estiver implantando sua amostra em instâncias do Windows Server, escolha CodePipeline-Jenkins-AWSCodeDeploy _ Windows.zip.

2. No repositório, escolha Fork para clonar o repositório de amostra em um repositório em sua conta. GitHub Para obter mais informações, consulte a documentação do GitHub.

Criar um perfil do IAM a ser usado na integração do Jenkins

Como melhor prática, considere iniciar uma instância do EC2 para hospedar seu servidor Jenkins e usar uma função do IAM para conceder à instância as permissões necessárias para interagir. CodePipeline

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação do console do IAM, selecione Perfis e, depois, Criar perfil.
3. Em Select type of trusted entity (Selecionar o tipo de entidade confiável), escolha AWS service (Serviço da AWS). Em Choose the service that will use this role (Escolha o serviço que usará essa função), escolha EC2. Em Select your use case (Selecione seu caso de uso), escolha EC2.
4. Escolha Próximo: Permissões. Na página Attach permissions policies (Anexar políticas de permissões), selecione a política gerenciada `AWSCodePipelineCustomActionAccess` e escolha Next: Tags (Próximo: tags). Escolha Próximo: revisar.
5. Na página Review (Revisar) em Role name (Nome da função), insira o nome da função a ser criada especificamente para integração do Jenkins (por exemplo *`JenkinsAccess`*) e escolha Create role (Criar função).

Ao criar a instância do EC2 em que você instalará o Jenkins, em Step 3: Configure Instance Details (Etapa 3: configurar os detalhes da instância), selecione a função da instância (por exemplo, *`JenkinsAccess`*).

Para obter mais informações sobre os perfis de instância e o Amazon EC2, consulte Funções do IAM para Amazon EC2, Uso de uma função do IAM para conceder permissões a aplicações em execução em instâncias do Amazon EC2 e Criar uma função para delegar permissões a um AWS service (Serviço da AWS).

Instale e configure o Jenkins e o CodePipeline plug-in para Jenkins

Para instalar o Jenkins e o CodePipeline plug-in para Jenkins

1. Crie uma instância do EC2 em que você instalará o Jenkins e, em Step 3: Configure Instance Details (Etapa 3: configurar os detalhes da instância), selecione a função da instância que

você criou (por exemplo *JenkinsAccess*). Para obter mais informações sobre a criação de instâncias do EC2, consulte Launch an Amazon EC2 instance no Guia do usuário do Amazon EC2.

> **Note**
>
> Se você já tem recursos do Jenkins que deseja utilizar, pode usá-los, mas deve criar um usuário especial do IAM, aplicar a política gerenciada `AWSCodePipelineCustomActionAccess` a esse usuário e, depois, configurar e usar as credenciais de acesso para esse usuário no recurso do Jenkins. Se deseja utilizar a IU Jenkins para fornecer as credenciais, configure o Jenkins para permitir apenas HTTPS. Para obter mais informações, consulte Solução de problemas CodePipeline.

2. Instale o Jenkins na instância do EC2. Para obter mais informações, consulte a documentação do Jenkins para instalar o Jenkins e iniciar e acessar o Jenkins, assim como details of integration with Jenkins em Integrações de produtos e serviços com CodePipeline.
3. Inicie o Jenkins e, na página inicial, selecione Gerenciar Jenkins.
4. Na página Gerenciar Jenkins, selecione Gerenciar plug-ins.
5. Selecione a guia Available (Disponível) e, na caixa de pesquisa Filter (Filtro), insira **`AWS CodePipeline`**. Escolha CodePipeline Plugin for Jenkins na lista e escolha Baixar agora e instalar após reiniciar.
6. Na Plugins/Upgrades página Instalação, selecione Reiniciar o Jenkins quando a instalação estiver concluída e nenhum trabalho estiver em execução.
7. Selecione Voltar para o painel.
8. Na página principal, selecione Novo item.
9. Em Nome do item, insira um nome para o projeto Jenkins (por exemplo,*MyDemoProject*). Selecione Projeto estilo livre e depois OK.

> **Note**
>
> Verifique se o nome do projeto atende aos requisitos do CodePipeline. Para obter mais informações, consulte Cotas em AWS CodePipeline.

10. Na página de configuração do projeto, selecione a caixa de seleção Executar construções simultâneas se necessário. Em Source Code Management (Gerenciamento de código-fonte), selecione AWS CodePipeline. Se você instalou o Jenkins em uma instância do EC2 e a

configurou AWS CLI com o perfil do usuário do IAM que você criou para integração entre CodePipeline e Jenkins, deixe todos os outros campos vazios.

11. Escolha Avançado e, em Provedor, insira um nome para o provedor da ação conforme ela aparecerá em CodePipeline (por exemplo,*MyJenkinsProviderName*). Certifique-se de que esse nome seja exclusivo e fácil de lembrar. Você o utilizará quando adicionar uma ação de construção para o pipeline mais adiante neste tutorial e novamente quando adicionar uma ação de teste.

    > **Note**
    >
    > Este nome de ação deve atender aos requisitos de nomenclatura para ações no CodePipeline. Para obter mais informações, consulte Cotas em AWS CodePipeline.

12. Em Construir triggers, desmarque todas as caixas de seleção e depois selecione Apurar SCM. Em Schedule (Programação), insira cinco asteriscos separados por espaços, conforme a seguir:

    ```
    * * * * *
    ```

    Essa pesquisa é feita a CodePipeline cada minuto.

13. Em Construir, selecione Adicionar etapa da construção. Selecione Executar shell (Amazon Linux, RHEL ou Ubuntu Server) Executar comando em lote (Windows Server) e, depois, digite o seguinte:

    ```
    rake
    ```

    > **Note**
    >
    > Verifique se o ambiente está configurado com as variáveis e as configurações necessárias para executar rake; caso contrário, a construção falhará.

14. Escolha Adicionar ação pós-compilação e, em seguida, escolha AWS CodePipeline Publisher. Selecione Adicionar, e em Criar pontos de saída, deixe o local em branco. Esta configuração é o padrão. Ela criará um arquivo compactado no final do processo de construção.
15. Selecione Salvar para salvar seu projeto Jenkins.

Etapa 2: criar um funil no CodePipeline

Nesta parte do tutorial, você criará o pipeline usando o assistente Create Pipeline (Criar pipeline).

Para criar um processo de liberação CodePipeline automatizado

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Se necessário, use o seletor de região para alterar a região para aquela em que os recursos do pipeline estão localizados. Por exemplo, se você criou recursos para o tutorial anterior em `us-east-2`, verifique se o seletor de região está definido como Leste dos EUA (Ohio).

   Para obter mais informações sobre as regiões e os endpoints disponíveis CodePipeline, consulte AWS CodePipeline Endpoints e cotas.
3. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
4. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
5. Em Etapa 2: escolher a página de configurações do pipeline, em Nome do pipeline, insira o nome do seu pipeline.
6. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.
7. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.
8. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
9. Na página Etapa 3: Adicionar estágio de origem, em Provedor de origem, escolha GitHub.
10. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulteGitHub conexões.
11. Em Etapa 4: adicionar estágio de compilação, escolha Adicionar Jenkins. Em Nome do provedor, insira o nome da ação que você forneceu no CodePipeline Plugin para Jenkins (por exemplo*`MyJenkinsProviderName`*). Esse nome deve corresponder exatamente ao nome no CodePipeline Plugin for Jenkins. Em Server URL (URL do servidor), digite o URL da instância do EC2 onde Jenkins está instalado. Em Nome do projeto, insira o nome do projeto que você criou no Jenkins, como*`MyDemoProject`*, e escolha Avançar.

12. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

13. Na Etapa 6: Adicione o estágio de implantação, reutilize o CodeDeploy aplicativo e o grupo de implantação que você criou. Tutorial: Criar um pipeline simples (bucket do S3) Em Deploy provider (Fornecedor de implantação), escolha CodeDeploy. Em Application name (Nome do aplicativo), insira **`CodePipelineDemoApplication`** ou selecione o botão de atualização e selecione o nome do aplicativo na lista. Em Deployment group (Grupo de implantação), insira **`CodePipelineDemoFleet`** ou selecione-o na lista e selecione Next (Próximo).

    > **Note**
    >
    > Você pode usar seus próprios CodeDeploy recursos ou criar novos, mas poderá incorrer em custos adicionais.

14. Na Etapa 7: Revisar, revise as informações e escolha Criar funil.
15. O pipeline inicia automaticamente e executa a amostra através do pipeline. Você pode visualizar as mensagens de progresso, sucesso e falha à medida que o pipeline cria a amostra Haml em HTML e a implanta como uma página da web em cada uma das instâncias do Amazon EC2 na implantação. CodeDeploy

Etapa 3: Adicionar outro estágio ao pipeline

Agora, você adicionará um estágio de teste e uma ação de teste a esse estágio que use o teste do Jenkins incluído na amostra para determinar se a página da web tem algum conteúdo. Esse teste é apenas para fins de demonstração.

> **Note**
>
> Se não quiser adicionar outro estágio ao pipeline, você poderá adicionar uma ação de teste ao estágio Preparação do pipeline, antes ou depois da ação de implantação.

Adicionar um estágio de teste ao pipeline

Tópicos

- Procurar o endereço IP de uma instância
- Criar um projeto do Jenkins para testar a implantação
- Criar um quarto estágio

Procurar o endereço IP de uma instância

Para verificar o endereço IP de uma instância onde você implantou seu código

1. Após a exibição do status Bem-sucedido para o pipeline, na área do status para o estágio Staging, selecione Detalhes.
2. Na seção Detalhes da implantação, em ID da instância, selecione a ID da instância de uma das instâncias implantadas com êxito.
3. Copie o endereço IP da instância (por exemplo,*192.168.0.4*). Você usará esse endereço IP em seu teste Jenkins.

Criar um projeto do Jenkins para testar a implantação

Para criar o projeto Jenkins

1. Na instância onde você instalou o Jenkins, abra Jenkins e, na página principal, selecione Novo item.
2. Em Nome do item, insira um nome para o projeto Jenkins (por exemplo,*MyTestProject*). Selecione Projeto estilo livre e depois OK.

   > **Note**
   >
   > Certifique-se de que o nome do seu projeto atenda aos CodePipeline requisitos. Para obter mais informações, consulte Cotas em AWS CodePipeline.

3. Na página de configuração do projeto, selecione a caixa de seleção Executar construções simultâneas se necessário. Em Source Code Management (Gerenciamento de código-fonte), selecione AWS CodePipeline. Se você instalou o Jenkins em uma instância do EC2 e a configurou AWS CLI com o perfil do usuário do IAM que você criou para integração entre CodePipeline e Jenkins, deixe todos os outros campos vazios.

> **Important**
>
> Se você estiver configurando um projeto Jenkins e ele não estiver instalado em uma instância do Amazon EC2, ou se estiver instalado em uma instância do EC2 que esteja executando um sistema operacional Windows, preencha os campos conforme exigido pelas configurações de porta e host proxy e forneça as credenciais do usuário ou função do IAM que você configurou para integração entre Jenkins e. CodePipeline

4. Selecione Avançado e, em Categoria, selecione Testar.
5. Em Provedor, insira o mesmo nome que você usou para o projeto de construção (por exemplo,*MyJenkinsProviderName*). Você usará esse nome quando adicionar a ação de teste ao seu pipeline mais adiante neste tutorial.

> **Note**
>
> Esse nome deve atender aos requisitos de CodePipeline nomenclatura das ações. Para obter mais informações, consulte Cotas em AWS CodePipeline.

6. Em Construir triggers, desmarque todas as caixas de seleção e depois selecione Apurar SCM. Em Schedule (Programação), insira cinco asteriscos separados por espaços, conforme a seguir:

```
* * * * *
```

   Essa pesquisa é feita a CodePipeline cada minuto.
7. Em Construir, selecione Adicionar etapa da construção. Se você estiver implantando em instâncias do Amazon Linux, do RHEL ou do Ubuntu Server, selecione Executar shell. Insira o seguinte, onde o endereço IP é o endereço da instância do EC2 que você copiou anteriormente:

```
TEST_IP_ADDRESS=192.168.0.4 rake test
```

   Se você estiver implantando em instâncias do Windows Server, selecione Executar comando em lote e insira o seguinte, em que o endereço IP é o endereço da instância do EC2 que você copiou anteriormente:

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

> **Note**
>
> O teste assume uma porta padrão de 80. Se você deseja especificar uma porta diferente, adicione uma instrução de porta de teste, conforme a seguir:
>
> ```
> TEST_IP_ADDRESS=192.168.0.4 TEST_PORT=8000 rake test
> ```

8. Escolha Adicionar ação pós-compilação e, em seguida, escolha AWS CodePipeline Publisher. Não selecione Adicionar.
9. Selecione Salvar para salvar seu projeto Jenkins.

Criar um quarto estágio

Para adicionar um estágio ao seu pipeline que inclui a ação de teste Jenkins

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Em Name (Nome), selecione o nome do pipeline que você criou, MySecondPipeline.
3. Na página de detalhes do pipeline, selecione Editar.
4. Na página Edit (Editar), selecione + Stage (Estágio) para adicionar um estágio imediatamente após o estágio de compilação.
5. No campo de nome para o novo estágio, insira um nome (por exemplo, **`Testing`**) e escolha + Add action group (+ Adicionar grupo de ações).
6. Em Nome da ação, insira*`MyJenkinsTest-Action`*. Em Provedor de teste, escolha o nome do provedor que você especificou no Jenkins (por exemplo,*`MyJenkinsProviderName`*). Em Nome do projeto, insira o nome do projeto que você criou no Jenkins (por exemplo,*`MyTestProject`*). Em Artefatos de entrada, escolha o artefato da compilação do Jenkins cujo nome padrão é e escolha *`BuildArtifact`* Concluído.

> **Note**
>
> Como a ação de teste do Jenkins é executada no aplicativo criado na etapa de compilação do Jenkins, use o artefato de compilação para o artefato de entrada na ação de teste.

Para obter mais informações sobre artefatos de entrada e saída e a estrutura de pipelines, consulte CodePipeline referência de estrutura de tubulação.

7. Na página Editar, selecione Salvar alterações do pipeline. Na caixa de diálogo da opção Salvar alterações do pipeline, selecione Salvar e continuar.
8. Embora o novo estágio tenha sido adicionado ao seu pipeline, um status de Ainda não executado é exibido para o estágio, pois nenhuma alteração acionou outra execução do pipeline. Para executar a amostra por meio do pipeline revisado, na página de detalhes do pipeline, selecione Lançar alteração.

   A visualização do pipeline exibe os estágios e ações em seu pipeline e o estado da revisão em execução através desses quatro estágios. O tempo necessário para que o pipeline seja executado por todos os estágios dependerá do tamanho dos artefatos, da complexidade de sua construção ações de teste, além de outros fatores.

Etapa 4: Limpar os recursos

Após concluir este tutorial, você deverá excluir o pipeline e os recursos usados para não ser cobrado pelo uso contínuo desses recursos. Se você não pretende continuar usando CodePipeline, exclua o pipeline, depois o CodeDeploy aplicativo e suas instâncias associadas do Amazon EC2 e, finalmente, o bucket do Amazon S3 usado para armazenar artefatos. Você também deve considerar a possibilidade de excluir outros recursos, como o GitHub repositório, caso não pretenda continuar usando-os.

Para limpar os recursos usados neste tutorial

1. Abra uma sessão de terminal em sua máquina Linux, macOS ou Unix local ou um prompt de comando em sua máquina Windows local e execute o comando delete-pipeline para excluir o pipeline que você criou. Em **`MySecondPipeline`**, insira o seguinte comando:

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

   Esse comando não retorna nada.
2. Para limpar seus CodeDeploy recursos, siga as instruções em Limpeza.
3. Para limpar os recursos de sua instância, exclua a instância do EC2 onde você instalou o Jenkins. Para obter mais informações, consulte Limpeza da sua instância.

4. Se você não pretende criar mais pipelines ou usá-los CodePipeline novamente, exclua o bucket do Amazon S3 usado para armazenar artefatos para seu pipeline. Para excluir o bucket, siga as instruções em Excluir um bucket.
5. Se você não pretende usar a outros recursos para esse pipeline novamente, considere excluí-los, seguindo as orientações para esse recurso específico. Por exemplo, se você quiser excluir o GitHub repositório, siga as instruções em Excluindo um repositório no site. GitHub

Tutorial: configurar uma regra de CloudWatch eventos para receber notificações por e-mail sobre mudanças no estado do pipeline

Depois de configurar um funil em AWS CodePipeline, você pode configurar uma regra de CloudWatch eventos para enviar notificações sempre que houver alterações no estado de execução de seus funis ou nas etapas ou ações em seus funis. Para obter mais informações sobre o uso de CloudWatch eventos para configurar notificações para mudanças no estado do pipeline, consulteMonitoramento de CodePipeline eventos.

Neste tutorial, você configura uma notificação para enviar um e-mail quando o estado de um pipeline muda para FAILED. Este tutorial usa um método de transformador de entrada ao criar a regra de CloudWatch eventos. Ele transforma os detalhes de esquema da mensagem para enviar a mensagem em texto legível por humanos.

> **Note**
>
> Ao criar os recursos para este tutorial, como a notificação do Amazon SNS e a regra de CloudWatch eventos, certifique-se de que os recursos sejam criados na mesma AWS região do seu pipeline.

Tópicos

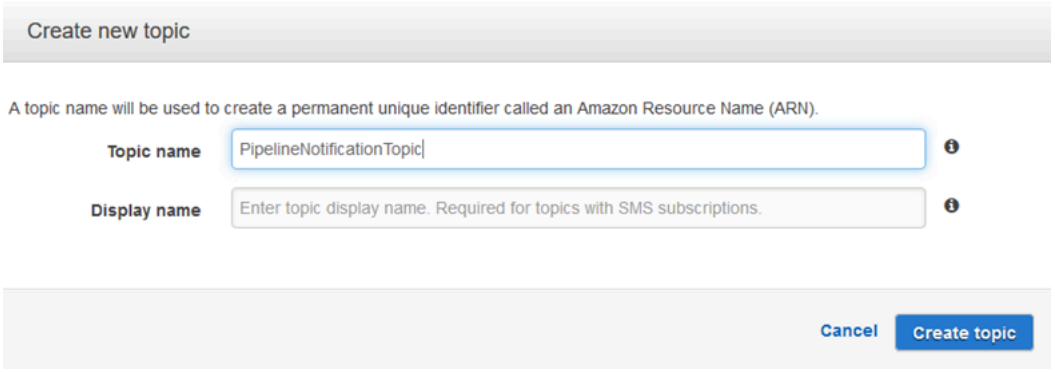
- Etapa 1: Configurar uma notificação por e-mail usando o Amazon SNS
- Etapa 2: Criar uma regra e adicionar o tópico do SNS como destino
- Etapa 3: Limpar os recursos

Etapa 1: Configurar uma notificação por e-mail usando o Amazon SNS

O Amazon SNS coordena o uso de tópicos para enviar mensagens a endpoints ou clientes assinantes. Use o Amazon SNS para criar um tópico de notificação e assinar o tópico usando seu endereço de e-mail. O tópico do Amazon SNS será adicionado como um alvo à sua regra de CloudWatch eventos. Para obter mais informações, consulte o Manual do desenvolvedor do Amazon Simple Notification Service.

Crie ou identifique um tópico no Amazon SNS. CodePipeline usará CloudWatch Eventos para enviar notificações sobre esse tópico por meio do Amazon SNS. Para criar um tópico:

1. Abra o console do Amazon SNS em https://console.aws.amazon.com/sns.
2. Escolha Criar tópico.
3. Na caixa de diálogo Create new topic (Criar novo tópico), em Topic name (Nome do tópico), digite um nome para o tópico (por exemplo, **`PipelineNotificationTopic`**).



4. Escolha Criar tópico.

   Para obter mais informações, consulte Criar um tópico no Guia do desenvolvedor do Amazon SNS.

Faça a assinatura de um ou mais destinatários para o tópico para que recebam notificações por e-mail. Para fazer a assinatura de um destinatário para um tópico:

1. No console do Amazon SNS, na lista Tópicos, marque a caixa de seleção ao lado do novo tópico. Escolha Actions, Subscribe to topic.
2. Na caixa de diálogo Create subscription, verifique se é exibido um ARN em Topic ARN.
3. Em Protocolo, escolha E-mail.
4. Em Endpoint, digite o endereço de e-mail completo do destinatário.

5. Escolha Criar inscrição.
6. O Amazon SNS envia um e-mail de confirmação de assinatura ao destinatário. Para receber notificações de e-mail, o destinatário deve escolher o link Confirm subscription nesse e-mail. Assim que o destinatário clicar no link, se a assinatura tiver sido realizada com êxito, o Amazon SNS exibirá uma mensagem de confirmação no navegador web do destinatário.

   Para obter mais informações, consulte Assinar um tópico no Guia do desenvolvedor do Amazon SNS.

Etapa 2: Criar uma regra e adicionar o tópico do SNS como destino

Crie uma regra de notificação de CloudWatch eventos com CodePipeline a fonte do evento.

1. Abra o CloudWatch console em https://console.aws.amazon.com/cloudwatch/.
2. No painel de navegação, escolha Eventos.
3. Escolha Criar regra. Em Event source (Origem do evento), escolha AWS CodePipeline. Em Event Type, escolha Pipeline Execution State Change.
4. Escolha Specific state(s) (Estado[s] específico[s]) e **`FAILED`**.
5. Escolha Edit para abrir o editor de JSON no painel Event Pattern Preview. Adicione o parâmetro **`pipeline`** com o nome de seu pipeline, como mostrado no exemplo a seguir para um pipeline chamado "myPipeline".

   É possível copiar o padrão de evento aqui e colá-lo no console:

```
{
  "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Pipeline Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "pipeline": [
      "myPipeline"
    ]
  }
```

```
}
```

6. Em Targets, escolha Add target.
7. Na lista de destinos, escolha SNS topic. Em Topic, insira o tópico que você criou.
8. Expanda Configure input e escolha Input Transformer.
9. Na caixa Input Path, digite os pares de chave/valor a seguir.

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

   Na caixa Input Template, digite o seguinte:

   ```
   "The Pipeline <pipeline> has failed."
   ```

10. Escolha Configure details (Configurar detalhes).
11. Na página Configure rule details, digite um nome e, opcionalmente, uma descrição. Em State, deixe a caixa Enabled marcada.
12. Escolha Criar regra.
13. Confirme se agora CodePipeline está enviando notificações de compilação. Por exemplo, verifique se agora os e-mails de notificação de compilação estão em sua caixa de entrada.
14. Para alterar o comportamento de uma regra, no CloudWatch console, escolha a regra e escolha Ações, Editar. Edite a regra e escolha Configure details e, em seguida, Update rule.

    Para parar de usar uma regra para enviar notificações de compilação, no CloudWatch console, escolha a regra e escolha Ações, Desativar.

    Para excluir uma regra, no CloudWatch console, escolha a regra e escolha Ações, Excluir.

Etapa 3: Limpar os recursos

Após concluir este tutorial, você deverá excluir o pipeline e os recursos usados para não ser cobrado pelo uso contínuo desses recursos.

Para obter informações sobre como limpar a notificação do SNS e excluir a regra do Amazon CloudWatch Events, consulte Limpar (cancelar a inscrição de um tópico do Amazon SNS) e fazer referência `DeleteRule` na Referência da API do Amazon CloudWatch Events.

Tutorial: crie um pipeline que crie e teste seu aplicativo Android com AWS Device Farm

Você pode usar AWS CodePipeline para configurar um fluxo de integração contínua no qual seu aplicativo é criado e testado sempre que uma confirmação é enviada. Este tutorial mostra como criar e configurar um pipeline para criar e testar seu aplicativo Android com o código-fonte em um GitHub repositório. O pipeline detecta a chegada de um novo GitHub commit e, em seguida, o usa CodeBuildpara criar o aplicativo e o Device Farm para testá-lo.

Important

Como parte da criação de um pipeline no console, um bucket de artefatos S3 será usado CodePipeline por for artefacts. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 Contas da AWS seja de propriedade de quem é seguro e confiável.

Important

Muitas das ações que você adiciona ao pipeline nesse procedimento envolvem AWS recursos que você precisa criar antes de criar o pipeline. AWS os recursos para suas ações de origem sempre devem ser criados na mesma AWS região em que você cria seu pipeline. Por exemplo, se você criar seu pipeline na região Leste dos EUA (Ohio), seu CodeCommit repositório deverá estar na região Leste dos EUA (Ohio).
Você pode adicionar ações entre regiões ao criar seu pipeline. AWS os recursos para ações entre regiões devem estar na mesma AWS região em que você planeja executar a ação. Para obter mais informações, consulte Adicione uma ação entre regiões em CodePipeline.

Você pode experimentar isso usando seu aplicativo Android e as definições de teste existentes ou pode usar o aplicativo de exemplo e as definições de teste fornecidas pelo Device Farm.

Antes de começar

1. Entre no AWS Device Farm console e escolha Criar um novo projeto.
2. Selecione o projeto. No navegador, copie o URL do novo projeto. O URL contém o ID do projeto.

3. Copie e retenha esse ID de projeto. Você pode usá-lo ao criar o pipeline no CodePipeline.

   Este é um exemplo de URL para um projeto. Para extrair o ID do projeto, copie o valor depois de `projects/`. Neste exemplo, o ID do produto é `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 seus testes do Device Farm

1. Adicione e confirme um arquivo chamado `buildspec.yml` na raiz do código do seu aplicativo e envie-o para o seu repositório. CodeBuild usa esse arquivo para executar comandos e acessar os artefatos necessários para criar seu aplicativo.

   ```
   version: 0.2

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

2. (Opcional) Se você usa o Calabash ou o Appium para testar seu aplicativo, adicione o arquivo de definição de teste ao seu repositório. Em uma etapa posterior, você pode configurar o Device Farm para usar as definições que executarão o pacote de teste.

   Se você usar os testes integrados do Device Farm, poderá ignorar esta etapa.

3. Para criar seu pipeline e adicionar um estágio de origem, faça o seguinte:

   a. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   b. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).

c. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.

d. Em Etapa 2: escolher a página de configurações do pipeline, em Nome do pipeline, insira o nome do seu pipeline.

e. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços de CodePipeline, consulte Preços.

f. Em Service role (Função de serviço), deixe New service role (Nova função de serviço) selecionado e não altere Role name (Nome da função). Você também pode optar por usar uma função de serviço existente se tiver criado uma.

> **Note**
>
> Se você usa uma função CodePipeline de serviço criada antes de julho de 2018, precisa adicionar permissões para o Device Farm. Para fazer isso, abra o console do IAM, localize o perfil e, em seguida, adicione as seguintes permissões à política do perfil. Para obter mais informações, consulte Adicionar permissões à função CodePipeline de serviço.
>
> ```
> {
>      "Effect": "Allow",
>      "Action": [
>         "devicefarm:ListProjects",
>         "devicefarm:ListDevicePools",
>         "devicefarm:GetRun",
>         "devicefarm:GetUpload",
>         "devicefarm:CreateUpload",
>         "devicefarm:ScheduleRun"
>      ],
>      "Resource": "*"
> }
> ```

g. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).

h. Na página Etapa 3: Adicionar estágio de origem, em Provedor de origem, escolha GitHub (via GitHub aplicativo).

i. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulteGitHub conexões.

j. Em Repository (Repositório), selecione o repositório de origem.

k. Em Branch (Ramificação), selecione a ramificação que você deseja usar.

l. Deixe os padrões restantes para a ação de origem. Escolha Próximo.

4. Em Etapa 4: Adicionar etapa de compilação, adicione um estágio de compilação:

a. Em Provedor de compilação, escolha Outros fornecedores de compilação e AWS CodeBuild. Permita que Region (Região) seja definida para a região do pipeline.

b. Escolha Criar projeto.

c. Em Nome do projeto, insira um nome para esse projeto de compilação.

d. Em Environment image (Imagem do ambiente), escolha Managed image (Imagem gerenciada). Para Operating system, selecione Ubuntu.

e. Em Runtime (Tempo de execução), selecione Standard (Padrão). Para Imagem, escolha: 5.0aws/codebuild/standard.

CodeBuild usa essa imagem do sistema operacional, que tem o Android Studio instalado, para criar seu aplicativo.

f. Em Função de serviço, escolha sua função CodeBuild de serviço existente ou crie uma nova.

g. Para Build specifications (Especificações da compilação), escolha Use a buildspec file (Usar um arquivo buildspec).

h. Escolha Continuar para CodePipeline. Isso retorna ao CodePipeline console e cria um CodeBuild projeto que usa o `buildspec.yml` em seu repositório para configuração. O projeto de criação usa um perfil de serviço para gerenciar permissões de AWS service (Serviço da AWS) . Essa etapa pode levar alguns minutos.

i. Escolha Próximo.

5. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

Escolha Próximo.

6. Na página Etapa 6: Adicionar estágio de implantação, escolha Ignorar estágio de implantação e, em seguida, aceite a mensagem de aviso escolhendo Ignorar novamente. Escolha Próximo.

7. Na Etapa 7: Revisão, escolha Criar funil. Você deve ver um diagrama que mostra a origem e os estágios de compilação.
8. Adicione uma ação de teste do Device Farm ao pipeline:

   a. No canto superior direito, escolha Edit (Editar).

   b. Na parte inferior do diagrama, escolha + Add stage (+ Adicionar estágio). Em Nome do estágio, insira um nome, como **`Test`**.

   c. Escolha + Add action group (Adicionar grupo de ação).

   d. Em Nome da ação, insira um nome.

   e. Em Provedor de ação, selecione AWS Device Farm. Permita que Region (Região) seja definida para a região do pipeline.

   f. Em Input artifacts (Artefatos de entrada), selecione o artefato de entrada que corresponde ao artefato de saída do estágio que vem antes do estágio de teste, como `BuildArtifact`.

      No AWS CodePipeline console, você pode encontrar o nome do artefato de saída para cada estágio passando o mouse sobre o ícone de informações no diagrama do pipeline. Se seu pipeline testar seu aplicativo diretamente do estágio de origem, escolha SourceArtifact. Se o pipeline incluir um estágio de construção, escolha BuildArtifact.

   g. Em ProjectId, insira o ID do projeto do Device Farm. Use as etapas no início deste tutorial para recuperar o ID do projeto.

   h. Em DevicePoolArn, insira o ARN do pool de dispositivos. Para obter o pool de dispositivos disponível ARNs para o projeto, incluindo o ARN dos principais dispositivos, use a AWS CLI para inserir o seguinte comando:

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

   i. Em AppType, insira Android.

      Veja a seguir uma lista de valores válidos para AppType:

      - iOS
      - Android
      - Web

j. Em App (Aplicativo), insira o caminho do pacote do aplicativo compilado. O caminho é relativo à raiz do artefato de entrada do estágio de teste. Normalmente, esse caminho é semelhante a `app-release.apk`.

k. Em TestType, insira seu tipo de teste e, em Teste, insira o caminho do arquivo de definição do teste. O caminho é relativo à raiz do artefato de entrada para o teste.

   Veja a seguir uma lista de valores válidos para TestType:

   - APPIUM_JAVA_JUNIT
   - APPIUM_JAVA_TESTING
   - APPIUM_NODE
   - APPIUM_RUBY
   - APPIUM_PYTHON
   - APPIUM_WEB_JAVA_JUNIT
   - APPIUM_WEB_JAVA_TESTING
   - APPIUM_WEB_NODE
   - APPIUM_WEB_RUBY
   - APPIUM_WEB_PYTHON
   - BUILTIN_FUZZ
   - INSTRUMENTATION
   - XCTEST
   - XCTEST_UI

   > **Note**
   >
   > Os nós de ambiente personalizados não são compatíveis.

l. Nos campos restantes, forneça a configuração que seja apropriada para seu teste e tipo de aplicativo.

m. (Opcional) Em Advanced (Avançado), forneça informações de configuração para a execução do teste.

n. Escolha Salvar.

o. No estágio que está editando, escolha Done (Concluído). No painel do AWS CodePipeline , escolha Save (Salvar) e selecione Save (Salvar) na mensagem de aviso.

p. Para enviar suas alterações e iniciar uma compilação do pipeline, selecione Liberar alteração e, depois, Liberar.

Tutorial: crie um pipeline que teste seu aplicativo iOS com AWS Device Farm

Você pode usar AWS CodePipeline para configurar facilmente um fluxo de integração contínua no qual seu aplicativo é testado sempre que o bucket de origem é alterado. Este tutorial mostra como criar e configurar um pipeline para testar seu aplicativo iOS criado a partir de um bucket do S3. O pipeline detecta a chegada de uma alteração salva por meio do Amazon CloudWatch Events e, em seguida, usa o Device Farm para testar o aplicativo criado.

Important

Como parte da criação de um pipeline, um bucket de artefatos S3 fornecido pelo cliente será usado CodePipeline por for artefacts. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 Contas da AWS seja de propriedade de quem é seguro e confiável.

Important

Muitas das ações que você adiciona ao pipeline nesse procedimento envolvem AWS recursos que você precisa criar antes de criar o pipeline. AWS os recursos para suas ações de origem sempre devem ser criados na mesma AWS região em que você cria seu pipeline. Por exemplo, se você criar seu pipeline na região Leste dos EUA (Ohio), seu CodeCommit repositório deverá estar na região Leste dos EUA (Ohio).
Você pode adicionar ações entre regiões ao criar seu pipeline. AWS os recursos para ações entre regiões devem estar na mesma AWS região em que você planeja executar a ação. Para obter mais informações, consulte Adicione uma ação entre regiões em CodePipeline.

Você pode experimentar isso usando seu aplicativo iOS existente ou você pode usar o aplicativo iOS de exemplo.

Antes de começar

1. Entre no AWS Device Farm console e escolha Criar um novo projeto.
2. Selecione o projeto. No navegador, copie o URL do novo projeto. O URL contém o ID do projeto.
3. Copie e retenha esse ID de projeto. Você pode usá-lo ao criar o pipeline no CodePipeline.

   Este é um exemplo de URL para um projeto. Para extrair o ID do projeto, copie o valor depois de `projects/`. Neste exemplo, o ID do produto é `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 seus testes do Device Farm (exemplo do Amazon S3)

1. Crie ou use um bucket do S3 com o versionamento habilitado. Você pode seguir as instruções em Etapa 1: criar um bucket de origem do S3 para a aplicação para criar um bucket do S3.
2. No console do Amazon S3 para o bucket, selecione Carregar e siga as instruções para fazer upload do arquivo .zip.

   Seu aplicativo de amostra deve ser compactado em um arquivo .zip.
3. Para criar seu pipeline e adicionar um estágio de origem, faça o seguinte:

   a. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   b. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).

   c. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.

   d. Em Etapa 2: escolher a página de configurações do pipeline, em Nome do pipeline, insira o nome do seu pipeline.

   e. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações,

consulte Pipeline types. Para obter informações sobre preços de CodePipeline, consulte Preços.

f. Em Service role (Função de serviço), deixe New service role (Nova função de serviço) selecionado e não altere Role name (Nome da função). Você também pode optar por usar uma função de serviço existente se tiver criado uma.

> **Note**
>
> Se você usa uma função CodePipeline de serviço criada antes de julho de 2018, você deve adicionar permissões para o Device Farm. Para fazer isso, abra o console do IAM, localize o perfil e, em seguida, adicione as seguintes permissões à política do perfil. Para obter mais informações, consulte Adicionar permissões à função CodePipeline de serviço.
>
> ```
> {
>      "Effect": "Allow",
>      "Action": [
>         "devicefarm:ListProjects",
>         "devicefarm:ListDevicePools",
>         "devicefarm:GetRun",
>         "devicefarm:GetUpload",
>         "devicefarm:CreateUpload",
>         "devicefarm:ScheduleRun"
>      ],
>      "Resource": "*"
> }
> ```

g. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).

h. Na página Etapa 3: Adicionar etapa de origem, em Provedor de origem, escolha Amazon S3.

i. Em Local do Amazon S3, insira o bucket, como `my-storage-bucket`, e a chave do objeto, como `s3-ios-test-1.zip`, para o arquivo .zip.

j. Escolha Próximo.

4. Em Etapa 4: Adicionar etapa de compilação, crie um estágio de compilação de espaço reservado para o pipeline. Isso permite que você crie o pipeline no assistente. Depois de usar o assistente para criar seu pipeline de dois estágios, você não precisa mais desse estágio

de compilação de espaço reservado. Depois que o pipeline é concluído, o segundo estágio é excluído e o novo estágio de teste é adicionado na etapa 5.

a. Em Build provider (Provedor de compilação), selecione Add Jenkins (Adicionar Jenkins). Essa seleção de compilação é um espaço reservado. Ele não é usado.

b. Em Provider name (Nome do provedor), insira um nome. O nome é um espaço reservado. Ele não é usado.

c. Em Server URL (URL do servidor), insira o texto. O texto é um espaço reservado. Ele não é usado.

d. Em Project name (Nome do projeto), insira um nome. O nome é um espaço reservado. Ele não é usado.

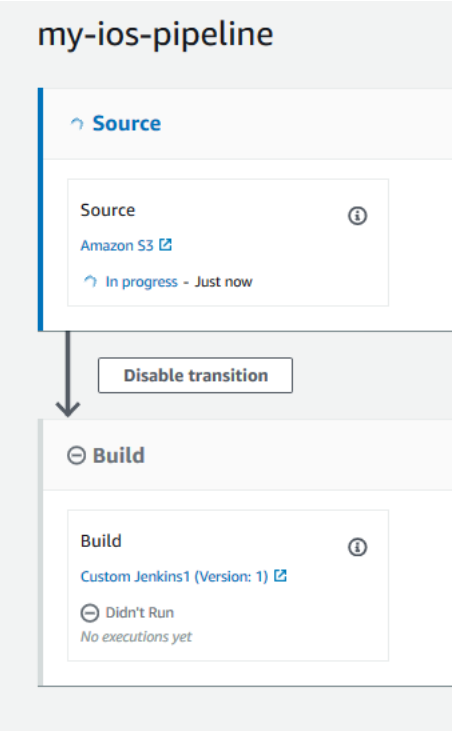
e. Escolha Próximo.

f. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

   Escolha Próximo.

g. Na página Etapa 6: Adicionar estágio de implantação, escolha Ignorar estágio de implantação e, em seguida, aceite a mensagem de aviso escolhendo Ignorar novamente.

h. Na Etapa 7: Revisão, escolha Criar funil. Você deve ver um diagrama que mostra a origem e os estágios de compilação.



5. Adicione uma ação de teste do Device Farm ao pipeline da seguinte maneira:

   a. No canto superior direito, escolha Edit (Editar).

   b. Selecione Edit stage (Editar estágio). Escolha Excluir. Isso exclui o estágio de espaço reservado agora que ele não é mais necessário para a criação do pipeline.

   c. Na parte inferior do diagrama, escolha + Add stage (+ Adicionar estágio).

   d. Em Stage name (Nome do estágio), insira um nome para o estágio, como Teste, e escolha Add stage (Adicionar estágio).

   e. Escolha + Add action group (Adicionar grupo de ação).

   f. Em Nome da ação, insira um nome, como DeviceFarmTest.

   g. Em Provedor de ação, selecione AWS Device Farm. Permita que Region (Região) seja definida para a região do pipeline.

   h. Em Input artifacts (Artefatos de entrada), selecione o artefato de entrada que corresponde ao artefato de saída do estágio que vem antes do estágio de teste, como `SourceArtifact`.

      No AWS CodePipeline console, você pode encontrar o nome do artefato de saída para cada estágio passando o mouse sobre o ícone de informações no diagrama do pipeline. Se seu pipeline testar seu aplicativo diretamente do estágio de origem, escolha SourceArtifact. Se o pipeline incluir um estágio de construção, escolha BuildArtifact.

i. Em ProjectId, escolha o ID do projeto do Device Farm. Use as etapas no início deste tutorial para recuperar o ID do projeto.

j. Em DevicePoolArn, insira o ARN do pool de dispositivos. Para obter o pool de dispositivos disponível ARNs para o projeto, incluindo o ARN dos principais dispositivos, use a AWS CLI para inserir o seguinte comando:

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

k. Em AppType, insira iOS.

Veja a seguir uma lista de valores válidos para AppType:

- iOS
- Android
- Web

l. Em App (Aplicativo), insira o caminho do pacote do aplicativo compilado. O caminho é relativo à raiz do artefato de entrada do estágio de teste. Normalmente, esse caminho é semelhante a `ios-test.ipa`.

m. Em TestType, insira seu tipo de teste e, em Teste, insira o caminho do arquivo de definição do teste. O caminho é relativo à raiz do artefato de entrada para o teste.

Se estiver usando um dos testes integrados do Device Farm, insira o tipo de teste configurado no projeto do Device Farm, como BUILTIN_FUZZ. Em FuzzEventCount, insira um tempo em milissegundos, como 6000. Em FuzzEventThrottle, insira um tempo em milissegundos, como 50.

Se não estiver usando um dos testes integrados do Device Farm, insira o tipo de teste e, em Teste, insira o caminho do arquivo de definição do teste. O caminho é relativo à raiz do artefato de entrada para o teste.

Veja a seguir uma lista de valores válidos para TestType:

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTING
- APPIUM_NODE
- APPIUM_RUBY

- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTING
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- BUILTIN_FUZZ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

> **Note**
>
> Os nós de ambiente personalizados não são compatíveis.

n. Nos campos restantes, forneça a configuração que seja apropriada para seu teste e tipo de aplicativo.

o. (Opcional) Em Advanced (Avançado), forneça informações de configuração para a execução do teste.

p. Escolha Salvar.

q. No estágio que está editando, escolha Done (Concluído). No painel do AWS CodePipeline , escolha Save (Salvar) e selecione Save (Salvar) na mensagem de aviso.

r. Para enviar as alterações e iniciar uma execução de pipeline, selecione Release change (Liberar alteração) e Release (Liberar).

Tutorial: Criar um pipeline que realiza a implantação no Service Catalog

O Service Catalog permite criar e provisionar produtos com base em AWS CloudFormation modelos.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Este tutorial mostra como criar e configurar um pipeline para implantar seu modelo de produto no Service Catalog e entregar as alterações que você fez no seu repositório de origem (já criado no GitHub CodeCommit, ou no Amazon S3).

> **Note**
>
> Quando o Amazon S3 é o provedor de origem do pipeline, você deve fazer upload de todos os arquivos de origem compactados em seu bucket como um único arquivo .zip. Caso contrário, a ação da origem falhará.

Primeiro, crie um produto no Service Catalog e depois crie um pipeline no AWS CodePipeline. Este tutorial fornece duas opções para definir a configuração de implantação:

- Crie um produto no Service Catalog e faça upload de um arquivo de modelo no repositório de origem. Forneça a versão do produto e a configuração de implantação no CodePipeline console (sem um arquivo de configuração separado). Consulte Opção 1: Implantar no Service Catalog sem um arquivo de configuração.

  > **Note**
  >
  > O arquivo de modelo pode ser criado no formato JSON ou YAML.

- Crie um produto no Service Catalog e faça upload de um arquivo de modelo no repositório de origem. Forneça a versão do produto e a configuração de implantação em um arquivo de configuração separado. Consulte Opção 2: Implantar no Service Catalog com um arquivo de configuração.

Opção 1: Implantar no Service Catalog sem um arquivo de configuração

Neste exemplo, você carrega o arquivo de AWS CloudFormation modelo de amostra para um bucket do S3 e, em seguida, cria seu produto no Service Catalog. Em seguida, você cria seu pipeline e especifica a configuração de implantação no CodePipeline console.

Etapa 1: Fazer upload do arquivo do modelo de exemplo no repositório de origem

1. Abra um editor de texto. Crie um modelo de exemplo colando o seguinte no arquivo. Salve o arquivo 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"
    }
  }
}
```

   Esse modelo permite AWS CloudFormation criar um bucket do S3 que pode ser usado pelo Service Catalog.

2. Carregue o arquivo `S3_template.json` no repositório do AWS CodeCommit .

Etapa 2: Criar um produto no Service Catalog

1. Como administrador de TI, faça login no console do Service Catalog, vá para a página Produtos e, depois, selecione Fazer upload de novo produto.
2. Na página Upload new product (Carregar novo produto), conclua o seguinte:
   a. Em Product name (Nome do produto), insira o nome que você deseja usar para o seu novo produto.
   b. Em Description (Descrição), insira a descrição do catálogo de produtos. Essa descrição é mostrada na lista de produtos para ajudar o usuário a escolher o produto correto.
   c. Em Provided by (Oferecido por), digite o nome de seu departamento ou do administrador de TI.
   d. Escolha Próximo.
3. (Opcional) Em Enter support details (Inserir detalhes do suporte), insira as informações de contato para suporte ao produto e selecione Next (Próximo).
4. Em Version details (Detalhes da versão), conclua o seguinte:
   a. Selecione Carregar um arquivo de modelo. Procure o arquivo `S3_template.json` e carregue-o.
   b. Em Version title (Título da versão), insira o nome da versão do produto (por exemplo, **`devops S3 v2`**).
   c. Em Description (Descrição), insira os detalhes que distinguem essa versão em relação às outras.
   d. Escolha Próximo.
5. Na página Review (Revisar), verifique se as informações estão corretas e escolha Create (Criar).
6. Na página Products (Produtos), no navegador, copie o URL do seu novo produto. Ela contém o ID do produto. Copie e retenha esse ID de produto. Você pode usá-lo ao criar o pipeline no CodePipeline.

   Veja a seguir o URL para um produto chamado `my-product`. Para extrair o ID do produto, copie o valor entre o sinal de igual (=) e o E comercial (&). Neste exemplo, o ID do produto é `prod-example123456`.

```
https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
productCreated=prod-example123456&createdProductTitle=my-product
```

> **Note**
>
> Copie o URL para o produto antes de sair da página. Depois de sair dessa página, você deve usar a CLI para obter o ID do produto.

Após alguns segundos, o produto será exibido na página Produtos. Talvez seja necessário atualizar o navegador para ver o produto na lista.

Etapa 3: Criar o pipeline

1. Para nomear o pipeline e selecionar os parâmetros para o pipeline, faça o seguinte:

   a. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   b. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).

   c. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.

   d. Em Etapa 2: escolher as configurações do pipeline, em Nome do pipeline, insira um nome para o pipeline.

   e. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.

   f. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.

   g. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).

2. Para adicionar um estágio de origem na página Etapa 3: Adicionar etapa de origem, faça o seguinte:

   a. Em Source provider (Provedor de código-fonte), selecione AWS CodeCommit.

   b. Em Repository name (Nome do repositório) e Branch name (Nome da ramificação), insira o nome do repositório e a ramificação que deseja usar para sua ação de origem.

c. Escolha Próximo.

3. Em Etapa 4: Adicionar etapa de compilação, escolha Ignorar etapa de compilação e aceite a mensagem de aviso escolhendo Ignorar novamente.
4. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

   Escolha Próximo.
5. Em Etapa 6: Adicionar etapa de implantação, conclua o seguinte:

   a. Em Deploy provider (Fornecedor de implantação), escolha AWS Service Catalog.

   b. Para a configuração de implantação, escolha Enter deployment configuration (Inserir configuração de implantação).

   c. Em ID do produto, cole o ID do produto que você copiou no console do Service Catalog.

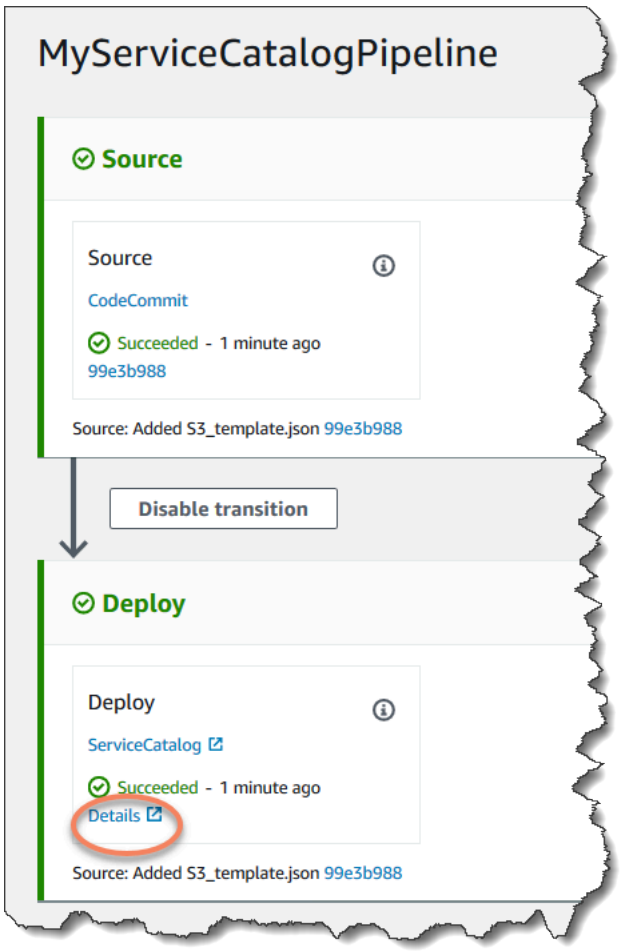
   d. Em Template file path (Caminho do arquivo do modelo), insira o caminho relativo em que o arquivo do modelo é armazenado.

   e. Em Tipo de produto, escolha modelo do CloudFormation .

   f. Em Nome da versão do produto, insira o nome da versão do produto que você especificou no Service Catalog. Se você deseja que a alteração do modelo seja implantada em uma nova versão do produto, insira um nome de versão do produto que não tenha sido usado em uma versão anterior do mesmo produto.

   g. Em Input artifact (Artefato de entrada), selecione o artefato de entrada de origem.

   h. Escolha Próximo.
6. Em Etapa 7: Revisão, examine as configurações do pipeline e, em seguida, escolha Criar.
7. Depois que o pipeline for executado com êxito, no estágio de implantação, selecione Details (Detalhes). O produto será aberto no Service Catalog.



8. Nas informações do produto, escolha o nome da versão para abrir o modelo do produto. Visualize a implantação do modelo.

Etapa 4: Enviar uma alteração e verificar o produto no Service Catalog

1. Visualize seu funil no CodePipeline console e, no estágio de origem, escolha Detalhes. Seu AWS CodeCommit repositório de origem é aberto no console. Selecione Edit (Editar) e faça uma alteração no arquivo (por exemplo, na descrição).

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

2. Confirme e envie a alteração. O pipeline será iniciado depois que você enviar a alteração. Quando a execução do pipeline for concluída, no estágio de implantação, selecione Detalhes para abrir seu produto no Service Catalog.
3. Nas informações do produto, escolha o nome da nova versão para abrir o modelo do produto. Visualize a alteração do modelo implantada.

Opção 2: Implantar no Service Catalog com um arquivo de configuração

Neste exemplo, você carrega o arquivo de AWS CloudFormation modelo de amostra para um bucket do S3 e, em seguida, cria seu produto no Service Catalog. Você também pode carregar um arquivo de configuração separado que especifica a configuração de implantação. Em seguida, crie o pipeline e especifique o local do seu arquivo de configuração.

Etapa 1: Fazer upload do arquivo do modelo de exemplo no repositório de origem

1. Abra um editor de texto. Crie um modelo de exemplo colando o seguinte no arquivo. Salve o arquivo 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"
    }
  }
}
```

   Esse modelo permite AWS CloudFormation criar um bucket do S3 que pode ser usado pelo Service Catalog.

2. Carregue o arquivo `S3_template.json` no repositório do AWS CodeCommit .

Etapa 2: Criar o arquivo de configuração da implantação do produto

1. Abra um editor de texto. Crie o arquivo de configuração para o seu produto. O arquivo de configuração é usado para definir sua implantação do Service Catalog parameters/preferences. Use esse arquivo ao criar o pipeline.

   Este exemplo fornece um `ProductVersionName` de "devops S3 v2" e um `ProductVersionDescription` de `MyProductVersionDescription`. Se você deseja que a alteração do modelo seja implantada em uma nova versão do produto, basta inserir um nome de versão do produto que não tenha sido usado em uma versão anterior do mesmo produto.

   Salve o arquivo como `sample_config.json`.

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

   Esse arquivo criará as informações da versão do produto para você a cada vez que o pipeline for executado.

2. Carregue o arquivo `sample_config.json` no repositório do AWS CodeCommit . Certifique-se de que você carregue esse arquivo no repositório de origem.

Etapa 3: Criar um produto no Service Catalog

1. Como administrador de TI, faça login no console do Service Catalog, vá para a página Produtos e, depois, selecione Fazer upload de novo produto.
2. Na página Upload new product (Carregar novo produto), conclua o seguinte:

   a. Em Product name (Nome do produto), insira o nome que você deseja usar para o seu novo produto.

   b. Em Description (Descrição), insira a descrição do catálogo de produtos. Essa descrição é mostrada na lista de produtos para ajudar o usuário a escolher o produto correto.

c. Em Provided by (Oferecido por), digite o nome de seu departamento ou do administrador de TI.

d. Escolha Próximo.

3. (Opcional) Em Enter support details (Inserir detalhes do suporte), insira as informações de contato para suporte ao produto e, em seguida, selecione Next (Próximo).

4. Em Version details (Detalhes da versão), conclua o seguinte:

   a. Selecione Carregar um arquivo de modelo. Procure o arquivo `S3_template.json` e carregue-o.

   b. Em Version title (Título da versão), insira o nome da versão do produto (por exemplo, "devops S3 v2").

   c. Em Description (Descrição), insira os detalhes que distinguem essa versão em relação às outras.

   d. Escolha Próximo.

5. Na página Revisão, verifique se as informações estão corretas e, em seguida, selecione Confirmar e carregar.

6. Na página Products (Produtos), no navegador, copie o URL do seu novo produto. Ela contém o ID do produto. Copie e retenha esse ID de produto. Você pode usá-lo ao criar o pipeline no CodePipeline.

   Veja a seguir o URL para um produto chamado `my-product`. Para extrair o ID do produto, copie o valor entre o sinal de igual (=) e o E comercial (&). Neste exemplo, o ID do produto é `prod-example123456`.

   ```
   https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
   productCreated=prod-example123456&createdProductTitle=my-product
   ```

   > **Note**
   >
   > Copie o URL para o produto antes de sair da página. Depois de sair dessa página, você deve usar a CLI para obter o ID do produto.

   Após alguns segundos, o produto será exibido na página Produtos. Talvez seja necessário atualizar o navegador para ver o produto na lista.

Etapa 4: Criar o pipeline

1. Para nomear o pipeline e selecionar os parâmetros para o pipeline, faça o seguinte:

   a. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   b. Escolha Conceitos básicos. Selecione Create pipeline (Criar pipeline) e insira um nome para seu pipeline.

   c. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.

   d. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).

2. Para adicionar um estágio de origem, faça o seguinte:

   a. Em Source provider (Provedor de código-fonte), selecione AWS CodeCommit.

   b. Em Repository name (Nome do repositório) e Branch name (Nome da ramificação), insira o nome do repositório e a ramificação que deseja usar para sua ação de origem.

   c. Escolha Próximo.

3. Em Add build stage (Adicionar estágio de compilação), selecione Skip build stage (Ignorar estágio de compilação) e aceite a mensagem de aviso ao clicar novamente em Skip (Ignorar).

4. Em Add deploy stage (Adicionar estágio de implantação), conclua o seguinte:

   a. Em Deploy provider (Fornecedor de implantação), escolha AWS Service Catalog.

   b. Selecione Use configuration file (Usar arquivo de configuração).

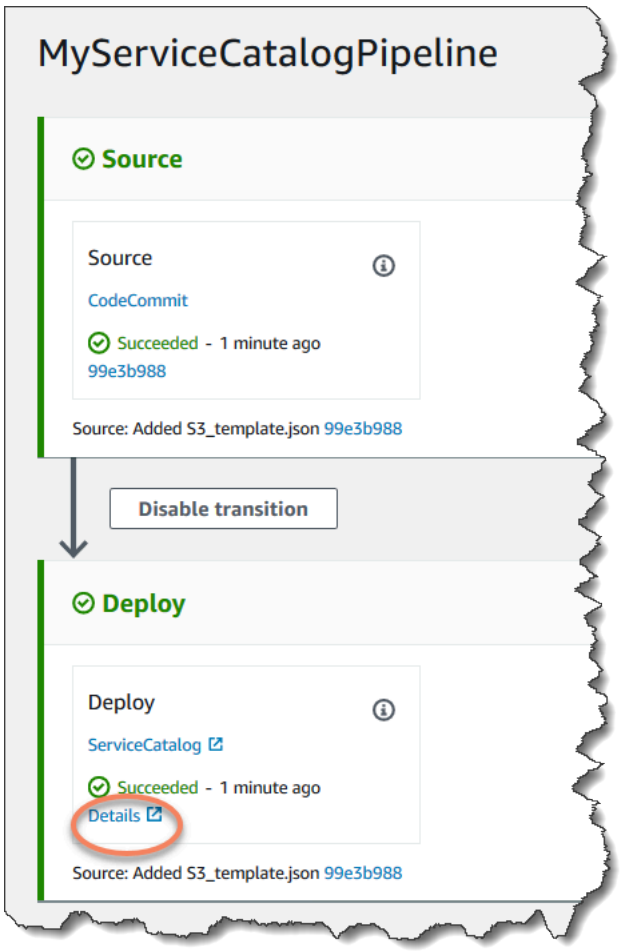
   c. Em ID do produto, cole o ID do produto que você copiou no console do Service Catalog.

   d. No arquivo de Configuration file path (Caminho do arquivo de configuração), insira o caminho do arquivo de configuração em seu repositório.

   e. Escolha Próximo.

5. Em Review (Revisão), revise as configurações do pipeline e, em seguida, selecione Create (Criar).

6. Depois que o pipeline for executado com êxito, no estágio de implantação, selecione Detalhes para abrir seu produto no Service Catalog.



7. Nas informações do produto, escolha o nome da versão para abrir o modelo do produto. Visualize a implantação do modelo.

Etapa 5: Enviar uma alteração e verificar o produto no Service Catalog

1. Visualize seu funil no CodePipeline console e, no estágio de origem, escolha Detalhes. Seu AWS CodeCommit repositório de origem é aberto no console. Selecione Edit (Editar) e, em seguida, faça uma alteração no arquivo (por exemplo, na descrição).

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

2. Confirme e envie a alteração. O pipeline será iniciado depois que você enviar a alteração. Quando a execução do pipeline for concluída, no estágio de implantação, selecione Detalhes para abrir seu produto no Service Catalog.
3. Nas informações do produto, escolha o nome da nova versão para abrir o modelo do produto. Visualize a alteração do modelo implantada.

Tutorial: criar um pipeline com AWS CloudFormation

Os exemplos fornecem modelos de amostra que permitem AWS CloudFormation criar um pipeline que implanta seu aplicativo em suas instâncias sempre que o código-fonte é alterado. O modelo de exemplo cria um pipeline que pode ser visualizado no AWS CodePipeline. O pipeline detecta a chegada de uma alteração salva por meio do Amazon CloudWatch Events.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Tópicos

- Exemplo 1: criar um AWS CodeCommit oleoduto com AWS CloudFormation
- Exemplo 2: Crie um pipeline do Amazon S3 com AWS CloudFormation

Exemplo 1: criar um AWS CodeCommit oleoduto com AWS CloudFormation

Este passo a passo mostra como usar o AWS CloudFormation console para criar uma infraestrutura que inclui um pipeline conectado a um repositório de CodeCommit origem. Neste tutorial, você usa o arquivo de modelo fornecido para criar sua pilha de recursos, que inclui seu armazenamento de artefatos, pipeline e recursos de detecção de alterações, como sua regra do Amazon Events. CloudWatch Depois de criar sua pilha de recursos AWS CloudFormation, você pode visualizar seu funil no AWS CodePipeline console. O pipeline é um pipeline de dois estágios com um estágio de CodeCommit origem e um estágio de CodeDeploy implantação.

Pré-requisitos:

Você deve ter criado os seguintes recursos para usar com o modelo AWS CloudFormation de amostra:

- Você deve ter criado um repositório de origem. Você pode usar o AWS CodeCommit repositório no Tutorial: criar um pipeline simples (CodeCommit repositório) qual você criou.
- Você deve ter criado um CodeDeploy aplicativo e um grupo de implantação. Você pode usar os recursos do CodeDeploy que você criou em Tutorial: criar um pipeline simples (CodeCommit repositório).
- Escolha um desses links para baixar o arquivo de AWS CloudFormation modelo de amostra para criar um pipeline: YAML | JSON samples/codepipeline-codecommit-events-json.zip

  Descompacte o arquivo e coloque-o em seu computador local.
- Faça o download do arquivo SampleApp _ Linux.zip de amostra do aplicativo.

Crie seu funil em AWS CloudFormation

1. Descompacte os arquivos de SampleApp _ Linux.zip e faça o upload dos arquivos para o seu AWS CodeCommit repositório. Você deve carregar os arquivos descompactados no diretório raiz do seu repositório. Você pode seguir as instruções em Etapa 2: adicionar código de amostra ao seu CodeCommit repositório para enviar os arquivos para o repositório.
2. Abra o AWS CloudFormation console e escolha Create Stack. Escolha With new resources (standard) (Com novos recursos [padrão]).
3. Em Especificar modelo, escolha Fazer upload de um modelo. Selecione Escolher arquivo e escolha o arquivo de modelo no seu computador local. Escolha Próximo.
4. Em Stack name (Nome da pilha), insira um nome para o pipeline. Os parâmetros especificados pelo modelo de exemplo são exibidos. Insira os seguintes parâmetros:

   a. Em ApplicationName, insira o nome do seu CodeDeploy aplicativo.

   b. Em BetaFleet, insira o nome do seu grupo CodeDeploy de implantação.

   c. Em BranchName, insira a ramificação do repositório que você deseja usar.

   d. Em RepositoryName, insira o nome do seu repositório CodeCommit de origem.
5. Escolha Próximo. Aceite os padrões na página a seguir e selecione Next (Próximo).
6. Em Capacidades, selecione Eu reconheço que AWS CloudFormation pode criar recursos do IAM e, em seguida, escolha Criar pilha.
7. Após a conclusão da criação da pilha, visualize a lista de eventos para verificar se há erros.

   Solução de problemas

O usuário do IAM que está criando o pipeline em AWS CloudFormation pode precisar de permissões adicionais para criar recursos para o pipeline. As seguintes permissões são exigidas na política para permitir AWS CloudFormation a criação dos recursos necessários do Amazon CloudWatch Events para o CodeCommit pipeline:

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

8. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   Em Pipelines, selecione o pipeline e clique em View (Visualizar). O diagrama mostra os estágios de implantação e a origem do pipeline.

   > **Note**
   >
   > Para visualizar o pipeline criado, localize a coluna ID lógico na guia Recursos da sua pilha no CloudFormation. Anote o nome na coluna ID físico do pipeline. Em CodePipeline, você pode visualizar o pipeline com a mesma ID física (nome do pipeline) na região em que criou sua pilha.

9. No seu repositório de origem, confirme e envie uma alteração. Seus recursos de detecção de alterações capturam a alteração, e o pipeline é iniciado.

Exemplo 2: Crie um pipeline do Amazon S3 com AWS CloudFormation

Este passo a passo mostra como usar o AWS CloudFormation console para criar uma infraestrutura que inclui um pipeline conectado a um bucket de origem do Amazon S3. Neste tutorial, você usa o arquivo de modelo fornecido para criar sua pilha de recursos, que inclui seu bucket de origem,

armazenamento de artefatos, pipeline e recursos de detecção de alterações, como sua regra e trilha do Amazon CloudWatch Events. CloudTrail Depois de criar sua pilha de recursos AWS CloudFormation, você pode visualizar seu funil no AWS CodePipeline console. O pipeline é um pipeline de dois estágios com um estágio de origem do Amazon S3 e um estágio de CodeDeploy implantação.

Pré-requisitos:

Você deve ter os seguintes recursos para usar com o modelo AWS CloudFormation de amostra:

- Você deve ter criado as instâncias do Amazon EC2, nas quais instalou o CodeDeploy agente nas instâncias. Você deve ter criado um CodeDeploy aplicativo e um grupo de implantação. Use o Amazon EC2 e CodeDeploy os recursos que você criou emTutorial: criar um pipeline simples (CodeCommit repositório).
- Escolha os links a seguir para baixar os arquivos de AWS CloudFormation modelo de amostra para criar um pipeline com uma fonte do Amazon S3:
  - Faça download do modelo de exemplo para o seu pipeline: YAML | JSON
  - Baixe o modelo de amostra para seu CloudTrail bucket and trail: YAML | JSON samples/codepipeline-s3-cloudtrail-json.zip
  - Descompacte os arquivos e coloque-os em seu computador local.
- Baixe o aplicativo de amostra em SampleApp _ Linux.zip.

  Salve o arquivo .zip em seu computador local. Você carregará o arquivo .zip após a criação da pilha.

Crie seu funil em AWS CloudFormation

1. Abra o AWS CloudFormation console e escolha Create Stack. Escolha With new resources (standard) (Com novos recursos [padrão]).
2. Em Escolher um modelo, escolha Fazer upload de um modelo. Selecione Escolher arquivo e escolha o arquivo de modelo no seu computador local. Escolha Próximo.
3. Em Stack name (Nome da pilha), insira um nome para o pipeline. Os parâmetros especificados pelo modelo de exemplo são exibidos. Insira os seguintes parâmetros:

   a. Em ApplicationName, insira o nome do seu CodeDeploy aplicativo. É possível substituir o nome padrão do `DemoApplication`.

b. Em BetaFleet, insira o nome do seu grupo CodeDeploy de implantação. É possível substituir o nome padrão do `DemoFleet`.

c. Entre SourceObjectKey, entre`SampleApp_Linux.zip`. Você carrega esse arquivo para o bucket depois que o modelo cria o bucket e o pipeline.

4. Escolha Próximo. Aceite os padrões na página a seguir e selecione Next (Próximo).

5. Em Capacidades, selecione Eu reconheço que AWS CloudFormation pode criar recursos do IAM e, em seguida, escolha Criar pilha.

6. Após a conclusão da criação da pilha, visualize a lista de eventos para verificar se há erros.

   Solução de problemas

   O usuário do IAM que está criando o pipeline em AWS CloudFormation pode precisar de permissões adicionais para criar recursos para o pipeline. As seguintes permissões são exigidas na política para permitir AWS CloudFormation a criação dos recursos necessários do Amazon CloudWatch Events para o pipeline do Amazon S3:

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

7. Em CloudFormation, na guia Recursos da sua pilha, veja os recursos que foram criados para sua pilha.

> **Note**
>
> Para visualizar o pipeline criado, localize a coluna ID lógico na guia Recursos da sua pilha no CloudFormation. Anote o nome na coluna ID físico do pipeline. Em CodePipeline, você pode visualizar o pipeline com a mesma ID física (nome do pipeline) na região em que criou sua pilha.

Escolha o bucket do S3 com um rótulo `sourcebucket` no nome, como `s3-cfn-codepipeline-sourcebucket-y04EXAMPLE.`. Não escolha o bucket de artefato do pipeline.

O bucket de origem está vazio porque o recurso foi criado recentemente pelo CloudFormation. Abra o console do Amazon S3 e localize seu bucket do `sourcebucket`. Escolha Upload (Fazer upload) e siga as instruções para carregar seu arquivo.zip `SampleApp_Linux.zip`.

> **Note**
>
> Quando o Amazon S3 é o provedor de origem do pipeline, você deve fazer upload de todos os arquivos de origem compactados em seu bucket como um único arquivo .zip. Caso contrário, a ação da origem falhará.

8. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   Em Pipelines, selecione o pipeline e clique em View (Visualizar). O diagrama mostra os estágios de implantação e a origem do pipeline.

9. Conclua as etapas dos procedimentos a seguir para criar seus recursos do AWS CloudTrail .

Crie seu AWS CloudTrail recursos em AWS CloudFormation

1. Abra o AWS CloudFormation console e escolha Create Stack.
2. Em Choose a template (Selecionar um modelo), selecione Upload a template to Amazon S3 (Carregar um modelo no Amazon S3). Escolha Procurar e selecione o arquivo de modelo para os AWS CloudTrail recursos do seu computador local. Escolha Próximo.
3. Em Stack name (Nome da pilha), informe um nome para sua pilha de recursos. Os parâmetros especificados pelo modelo de exemplo são exibidos. Insira os seguintes parâmetros:
   - Em SourceObjectKey, aceite o padrão para o arquivo zip do aplicativo de amostra.
4. Escolha Próximo. Aceite os padrões na página a seguir e selecione Next (Próximo).
5. Em Capacidades, selecione Eu reconheço que AWS CloudFormation pode criar recursos do IAM e, em seguida, escolha Criar.
6. Após a conclusão da criação da pilha, visualize a lista de eventos para verificar se há erros.

As seguintes permissões são exigidas na política para permitir AWS CloudFormation a criação dos CloudTrail recursos necessários para o pipeline do Amazon S3:

```
{
     "Effect": "Allow",
     "Action": [
        "cloudtrail:CreateTrail",
        "cloudtrail:DeleteTrail",
        "cloudtrail:StartLogging",
        "cloudtrail:StopLogging",
        "cloudtrail:PutEventSelectors"
     ],
     "Resource": "resource_ARN"
}
```

7. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   Em Pipelines, selecione o pipeline e clique em View (Visualizar). O diagrama mostra os estágios de implantação e a origem do pipeline.

8. No seu bucket de origem, confirme e envie uma alteração. Seus recursos de detecção de alterações capturam a alteração e o pipeline é iniciado.

Tutorial: criar um pipeline que usa variáveis de AWS CloudFormation ações de implantação

Neste tutorial, você usa o AWS CodePipeline console para criar um pipeline com uma ação de implantação. Quando o pipeline é executado, o modelo cria uma pilha e também cria um arquivo `outputs`. As saídas geradas pelo modelo de pilha são as variáveis geradas pela AWS CloudFormation ação em. CodePipeline

Na ação em que é criada a pilha a partir do modelo, um namespace variável é designado. As variáveis produzidas pelo arquivo `outputs` podem então ser consumidas por ações subsequentes. Neste exemplo, você cria um conjunto de alterações com base na `StackName` variável produzida pela AWS CloudFormation ação. Após uma aprovação manual, execute o conjunto de alterações e, depois, crie uma ação de pilha de exclusão que exclui a pilha com base na variável `StackName`.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Tópicos

- Pré-requisitos: criar um AWS CloudFormation função de serviço e um CodeCommit repositório
- Etapa 1: Baixe, edite e faça o upload da amostra AWS CloudFormation modelo
- Etapa 2: Criar o pipeline
- Etapa 3: adicionar um CloudFormation ação de implantação para criar o conjunto de alterações
- Etapa 4: Adicionar uma ação de aprovação manual
- Etapa 5: adicionar uma ação CloudFormation de implantação para executar o conjunto de alterações
- Etapa 6: Adicionar uma ação CloudFormation de implantação para excluir a pilha

Pré-requisitos: criar um AWS CloudFormation função de serviço e um CodeCommit repositório

Você já deve ter o seguinte:

- Um CodeCommit repositório. Você pode usar o AWS CodeCommit repositório no Tutorial: criar um pipeline simples (CodeCommit repositório) qual você criou.
- Neste exemplo é criada uma pilha do Amazon DocumentDB a partir de um modelo. Você deve usar AWS Identity and Access Management (IAM) para criar uma função AWS CloudFormation de serviço com as seguintes permissões para o Amazon DocumentDB.

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

Etapa 1: Baixe, edite e faça o upload da amostra AWS CloudFormation modelo

Faça o download do arquivo AWS CloudFormation de modelo de amostra e faça o upload para o seu CodeCommit repositório.

1. Navegue até o modelo de amostra da região. Por exemplo, use a tabela em https://docs.aws.amazon.com/documentdb/latest/developerguide/quick_start_cfn.html#quick_start_cfn-launch_stack para escolher a região e baixar o modelo. Baixe o modelo de um cluster do Amazon DocumentDB. O nome do arquivo é `documentdb_full_stack.yaml`.
2. Descompacte o arquivo `documentdb_full_stack.yaml` e abra-o em um editor de texto. Faça as alterações a seguir.

   a. Para este exemplo, adicione o parâmetro `Purpose:` a seguir à seção `Parameters` no modelo.

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

   b. Para este exemplo, adicione a saída `StackName` a seguir à seção `Outputs:` no modelo.

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

3. Faça o upload do arquivo de modelo para o seu AWS CodeCommit repositório. É necessário fazer upload do arquivo de modelo descompactado e editado para o diretório raiz do repositório.

   Para usar o CodeCommit console para fazer upload de seus arquivos:

   a. Abra o CodeCommit console e escolha seu repositório na lista Repositórios.

   b. Selecione Add file (Adicionar arquivo) e clique em Upload file (Carregar arquivo).

   c. Selecione Choose file (Escolher arquivo) e procure o arquivo. Informe seu nome de usuário e endereço de e-mail para confirmar a alteração. Escolha Commit changes (Confirmar alterações).

Seu arquivo deve ser parecido com isto no nível raiz do repositório:

```
documentdb_full_stack.yaml
```

Etapa 2: Criar o pipeline

Nesta seção, você criará um pipeline com as seguintes ações:

- Um estágio de origem com uma CodeCommit ação em que o artefato de origem é seu arquivo de modelo.
- Um estágio de implantação com uma ação CloudFormation de implantação.

Cada ação nos estágios de origem e implantação criada pelo assistente recebe um namespace variável, `SourceVariables` e `DeployVariables`, respectivamente. Como as ações têm um namespace atribuído, as variáveis configuradas neste exemplo estão disponíveis para ações downstream. Para obter mais informações, consulte Referência de variáveis.

Criar um pipeline com o assistente

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyCFNDeployPipeline`**.
5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.
6. Em Perfil de serviço, faça um dos seguintes procedimentos:
   - Escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.

- Escolha Existing service role (Função de serviço existente). Em Role name (Nome da função), selecione a função de serviço na lista.

7. Em Artifact store (Armazenamento de artefatos):

   a. Selecione Local padrão para usar o armazenamento de artefatos padrão, como o bucket de artefatos do Amazon S3 designado como padrão, para o pipeline na região selecionada para o pipeline.

   b. Selecione Local personalizado se você já tiver um armazenamento de artefatos, como um bucket de artefatos do Amazon S3 na mesma região do pipeline.

   > **Note**
   >
   > Este não é o bucket de origem para seu código-fonte. Este é o armazenamento de artefatos para o pipeline. Um armazenamento de artefatos separado, como um bucket do S3, é necessário para cada pipeline. Ao criar ou editar um pipeline, você deve ter um bucket de artefatos na região do pipeline e um bucket de artefatos por AWS região em que você está executando uma ação.
   > Para obter mais informações, consulte Artefatos de entrada e saída e CodePipeline referência de estrutura de tubulação.

   Escolha Próximo.

8. Na Etapa 3: Adicionar estágio de origem:

   a. Em Source provider (Provedor de código-fonte), selecione AWS CodeCommit.

   b. Em Nome do repositório, escolha o nome do CodeCommit repositório que você criou. Etapa 1: criar um CodeCommit repositório

   c. Em Nome do ramo, selecione o nome do ramo que contém a última atualização do código.

   Depois de selecionar o nome e a ramificação do repositório, a regra do Amazon CloudWatch Events a ser criada para esse pipeline é exibida.

   Escolha Próximo.

9. Em Etapa 4: Adicionar etapa de compilação, escolha Ignorar etapa de compilação e aceite a mensagem de aviso escolhendo Ignorar novamente.

Escolha Próximo.

10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

11. Na Etapa 6: Adicionar estágio de implantação:

    a. Em Nome da ação, escolha Implantar. Em Deploy provider (Fornecedor de implantação), escolha CloudFormation.

    b. No Modo de ação, escolha Criar ou atualizar uma pilha.

    c. Em Nome da pilha, insira um nome para a pilha. Esse é o nome da pilha que será criada pelo modelo.

    d. Em Nome do arquivo de saída, insira um nome para o arquivo de saída, como **`outputs`**. Esse é o nome do arquivo que será criado pela ação após a criação da pilha.

    e. Expanda Advanced. Em Substituições de parâmetros, insira suas substituições de modelo como pares de chave/valor. Por exemplo, este modelo requer as substituições a seguir.

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

    Se você não inserir substituições, o modelo criará uma pilha com valores padrão.

    f. Escolha Próximo.

    g. Na Etapa 7: Revisão, escolha Criar funil. Você deve ver um diagrama que mostra os estágios do pipeline. Deixe que o pipeline seja executado. O pipeline de dois estágios está completo e pronto para os estágios adicionais a serem incluídos.

Etapa 3: adicionar um CloudFormation ação de implantação para criar o conjunto de alterações

Crie uma próxima ação em seu funil que permitirá CloudFormation criar o conjunto de alterações antes da ação de aprovação manual.

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   Em Pipelines, selecione o pipeline e clique em View (Visualizar). O diagrama mostra os estágios de implantação e a origem do pipeline.
2. Escolha editar o pipeline ou continue exibindo o pipeline no modo Editar.
3. Opte por editar o estágio de implantação.
4. Adicione uma ação de implantação que criará um conjunto de alterações para a pilha criada na ação anterior. Você deve adicionar essa ação após a ação existente no estágio.

   a. Em Nome da ação, insira Change_Set. Em Provedor de ação, selecione AWS CloudFormation .

   b. Em Artefato de entrada, escolha SourceArtifact.

   c. Em Action mode (Modo de ação), escolha Create or replace a change set (Criar ou substitui um conjunto de alterações).

   d. Em Nome da pilha, insira a sintaxe da variável como mostrado. Esse é o nome da pilha para a qual o conjunto de alterações foi criado, em que o namespace padrão `DeployVariables` é atribuído à ação.

   ```
   #{DeployVariables.StackName}
   ```

   e. Em Alterar nome do conjunto, insira o nome do conjunto de alterações.

   ```
   my-changeset
   ```

   f. Em Substituições de parâmetros, altere o parâmetro `Purpose` de `testing` para `production`.

   ```
   {
   "DBClusterName": "MyDBCluster",
   "DBInstanceName": "MyDBInstance",
   "MasterUser": "UserName",
   ```

```
"MasterPassword": "Password",
"DBInstanceClass": "db.r4.large",
"Purpose": "production"}
```

g. Escolha Concluído para salvar a ação.

Etapa 4: Adicionar uma ação de aprovação manual

Crie uma ação de aprovação manual no pipeline.

1. Escolha editar o pipeline ou continue exibindo o pipeline no modo Editar.
2. Opte por editar o estágio de implantação.
3. Adicione uma ação de aprovação manual após a ação de implantação que cria o conjunto de alterações. Essa ação permite que você verifique a alteração de recurso criada CloudFormation antes que o pipeline execute o conjunto de alterações.

Etapa 5: adicionar uma ação CloudFormation de implantação para executar o conjunto de alterações

Crie uma próxima ação em seu funil que permita CloudFormation executar o conjunto de alterações após a ação de aprovação manual.

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   Em Pipelines, selecione o pipeline e clique em View (Visualizar). O diagrama mostra os estágios de implantação e a origem do pipeline.
2. Escolha editar o pipeline ou continue exibindo o pipeline no modo Editar.
3. Opte por editar o estágio de implantação.
4. Adicione uma ação de implantação que executará o conjunto de alterações aprovado na ação manual anterior:

   a. Em Nome da ação, insira Execute_Change_Set. Em Provedor de ação, selecione AWS CloudFormation.

   b. Em Artefato de entrada, escolha SourceArtifact.

c. Em Action mode (Modo de ação), escolha Execute a change set (Executar um conjunto de alterações).

d. Em Nome da pilha, insira a sintaxe da variável como mostrado. Esse é o nome da pilha para a qual o conjunto de alterações é criado.

```
#{DeployVariables.StackName}
```

e. Em Nome do conjunto de alterações, insira o nome do conjunto de alterações criado na ação anterior.

```
my-changeset
```

f. Escolha Concluído para salvar a ação.

g. Continue a execução do pipeline.

Etapa 6: Adicionar uma ação CloudFormation de implantação para excluir a pilha

Crie uma ação final em seu pipeline que permita CloudFormation obter o nome da pilha da variável no arquivo de saídas e excluir a pilha.

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   Em Pipelines, selecione o pipeline e clique em View (Visualizar). O diagrama mostra os estágios de implantação e a origem do pipeline.

2. Escolha editar o pipeline.

3. Opte por editar o estágio de implantação.

4. Adicione uma ação de implantação que excluirá a pilha:

   a. Em Nome da ação, escolha DeleteStack. Em Deploy provider (Fornecedor de implantação), escolha CloudFormation.

   b. Em Modo de ação, escolha Excluir uma pilha.

   c. Em Nome da pilha, insira a sintaxe da variável como mostrado. Esse é o nome da pilha que será excluída pela ação.

   d. Escolha Concluído para salvar a ação.

e. Escolha Salvar para salvar o pipeline.

O pipeline é executado quando ele é salvo.

Tutorial: Implantação padrão do Amazon ECS com CodePipeline

Este tutorial ajuda você a criar um pipeline completo de implantação contínua (CD) de ponta a ponta com o Amazon ECS with. CodePipeline

Important

Como parte da criação de um pipeline no console, um bucket de artefatos do S3 será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Note

Este tutorial é sobre a ação de implantação padrão do Amazon ECS para CodePipeline. Para ver um tutorial que usa o Amazon ECS para CodeDeploy blue/green implementar ações em CodePipeline, consulteTutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR.

Note

Este tutorial é para a ação de implantação padrão do Amazon ECS CodePipeline com uma ação de origem. Para ver um tutorial que usa a ação de implantação padrão do Amazon ECS junto com a ação de ECRBuildAndPublish criação CodePipeline para enviar sua imagem, consulteTutorial: crie e envie uma imagem do Docker para o Amazon ECR com CodePipeline (tipo V2).

Pré-requisitos

Para você usar este tutorial para criar seu pipeline de CD, alguns recursos precisam estar em operação. Veja aqui estão os itens que você precisa para começar:

> **Note**
>
> Todos esses recursos devem ser criados na mesma AWS região.

- Um repositório de controle de origem (este tutorial usa CodeCommit) com seu Dockerfile e a fonte do aplicativo. Para obter mais informações, consulte Criar um CodeCommit repositório no Guia do AWS CodeCommit usuário.
- Um repositório de imagens de Docker (este tutorial usa o Amazon ECR) que contém uma imagem criada com o Dockerfile e a origem da aplicação. Para obter mais informações, consulte Criação de um repositório e Envio de uma imagem por push no Guia do usuário do Amazon Elastic Container Registry.
- Uma definição de tarefa do Amazon ECS que faz referência à imagem do Docker hospedada em seu repositório de imagens. Para obter mais informações, consulte Como criar uma definição de tarefa no Guia do desenvolvedor do Amazon Elastic Container Service.

  > **Important**
  >
  > A ação de implantação padrão do Amazon ECS para CodePipeline cria sua própria revisão da definição da tarefa com base na revisão usada pelo serviço Amazon ECS. Se você criar novas revisões para a definição de tarefa sem atualizar o serviço Amazon ECS, a ação de implantação ignorará essas revisões.

  Veja a seguir um exemplo de definição de tarefa usado neste tutorial. O valor que você usa em `name` e `family` será usado na próxima etapa do seu arquivo de especificação de compilação.

```
{
  "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": []
}
```

- Um cluster do Amazon ECS que executa um serviço que usa a definição de tarefa mencionada anteriormente. Para obter mais informações, consulte Criação de um cluster e Criação de um serviço no Guia do desenvolvedor do Amazon Elastic Container Service.

Assim que você atender a esses pré-requisitos, poderá continuar com o tutorial e criar seu pipeline de CD.

Etapa 1: Adicionar um arquivo de especificação de compilação ao repositório de origem

Este tutorial é usado CodeBuild para criar sua imagem do Docker e enviá-la para o Amazon ECR. Adicione um arquivo `buildspec.yml` ao repositório de código-fonte para informar ao CodeBuild como fazer isso. O exemplo de especificação de compilação abaixo faz o seguinte:

- Pre-build estágio:
  - Faça login no Amazon ECR.
  - Defina a URI de repositório de sua imagem ECR e adicione uma tag de imagem com os primeiros sete caracteres do ID de confirmação do Git da origem.
- Estágio de compilação:
  - Crie a imagem de docker e marque-a como `latest` e com o ID de confirmação do Git.
- Post-build estágio:
  - Envie por push a imagem para o repositório do ECR com ambas as tags.
  - Grave um arquivo denominado `imagedefinitions.json` na raiz da compilação que contém o nome de contêiner, a imagem e tag do serviço do Amazon ECS. O estágio de implantação do pipeline de CD usa essas informações para criar uma nova revisão da definição de tarefa do serviço e, em seguida, atualiza o serviço para usar a nova definição de tarefa. O arquivo `imagedefinitions.json` é necessário para o operador de trabalho do ECS.

Cole este texto de exemplo para criar seu arquivo `buildspec.yml` e substitua os valores para definição da imagem e da tarefa. Este texto usa o exemplo de ID de conta 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
```

A especificação de compilação foi escrita para o exemplo de definição de tarefa fornecido em Pré-requisitos, usado pelo serviço do Amazon ECS para este tutorial. O valor `REPOSITORY_URI` corresponde ao repositório `image` (sem nenhuma tag de imagem) e o valor *`hello-world`* próximo do final do arquivo corresponde ao nome do contêiner na definição de tarefa do serviço.

Para adicionar um **`buildspec.yml`** arquivo para seu repositório de origem

1. Abra um editor de texto e, em seguida, copie e cole a especificação de compilação acima em um novo arquivo.
2. Substitua o valor `REPOSITORY_URI` (*`012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world`*) pelo URI do repositório do Amazon ECR (sem nenhuma tag de imagem) da imagem do Docker. Substitua *`hello-world`* pelo nome do contêiner na definição de tarefa do serviço que se refere à imagem de docker.
3. Confirme e envie o arquivo `buildspec.yml` para o repositório de origem.

   a. Adicione o arquivo.

   ```
   git add .
   ```

   b. Confirme a alteração.

   ```
   git commit -m "Adding build specification."
   ```

   c. Envie a confirmação.

```
git push
```

Etapa 2: Criar uma pipeline de implantação contínua

Use o CodePipeline assistente para criar seus estágios de pipeline e conectar seu repositório de origem ao seu serviço ECS.

Para criar o pipeline

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página Bem-vindo, escolha Criar pipeline.

   Se esta for a primeira vez que você usa CodePipeline, uma página introdutória será exibida em vez de Bem-vindo. Escolha Get Started Now (Começar agora).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher configurações de pipeline, em Nome do pipeline, digite o nome do pipeline. Para este tutorial, o nome do pipeline é hello-world.
5. Em Tipo de pipeline, mantenha a seleção padrão em V2. Os tipos de pipeline diferem em características e preços. Para obter mais informações, consulte Tipos de pipeline. Escolha Próximo.
6. Na página Etapa 3: Adicionar estágio de origem, em Provedor de origem, escolha AWS CodeCommit.

   a. Em Repository name (Nome do repositório), escolha o nome do repositório do CodeCommit para usar como o local de origem do pipeline.

   b. Em Branch name (Nome da ramificação), escolha a ramificação a ser usada e selecione Next (Próximo).
7. Na página Etapa 4: Adicionar estágio de construção, em Provedor de compilação AWS CodeBuild, escolha e, em seguida, escolha Criar projeto.

   a. Em Project name, escolha um nome exclusivo para seu projeto de compilação. Para este tutorial, o nome do projeto é hello-world.

   b. Em Imagem do ambiente, escolha Imagem gerenciada.

   c. Em Sistema operacional, escolha Amazon Linux 2.

d. Em Runtime(s), selecione Padrão.

e. Em Imagem, escolha **`aws/codebuild/amazonlinux2-x86_64-standard:3.0`**.

f. Em Image version (Versão da imagem) e Environment type (Tipo de ambiente), use os valores padrão.

g. Selecione Enable this flag if you want to build Docker images or want your builds to get elevated privileges (Ativar este sinalizador se quiser criar imagens do Docker ou desejar que suas compilações obtenham privilégios elevados).

h. Desmarque os registros CloudWatch . Talvez seja necessário expandir Avançado.

i. Escolha Continuar para CodePipeline.

j. Escolha Próximo.

> **Note**
>
> O assistente cria uma função CodeBuild de serviço para seu projeto de construção, chamada codebuild- ***`build-project-name`*** -service-role. Observe esse nome de perfil ao adicionar as permissões do Amazon ECR a ele posteriormente.

8. Na página Etapa 5: Adicionar estágio de implantação, para provedor de implantação, escolha Amazon ECS.

   a. Em Nome do cluster, escolha o cluster do Amazon ECS no qual o serviço está em execução. Para este tutorial, o cluster é default.

   b. Em Service name (Nome do serviço), escolha o serviço a ser atualizado e selecione Next (Próximo). Para este tutorial, o nome do serviço é hello-world.

9. Na página Step 6: Review, revise a configuração do pipeline e escolha Create pipeline para criá-lo.

> **Note**
>
> Agora que o pipeline foi criado, ele tentará passar pelos diferentes estágios de pipeline. No entanto, a CodeBuild função padrão criada pelo assistente não tem permissões para executar todos os comandos contidos no `buildspec.yml` arquivo, portanto, o estágio de criação falha. A próxima seção adiciona as permissões para o estágio de compilação.

Etapa 3: adicionar permissões do Amazon ECR à função CodeBuild

O CodePipeline assistente criou uma função do IAM para o projeto de construção, chamada CodeBuild codebuild- ***build-project-name*** -service-role. Para este tutorial, o nome será codebuild-hello-world-service-role. Como o arquivo `buildspec.yml` faz chamadas para operações de API do Amazon ECR, o perfil deve ter uma política que conceda permissões para a realização dessas chamadas ao Amazon ECR. O procedimento a seguir ajuda você a anexar as permissões adequadas à função.

Para adicionar permissões do Amazon ECR à função CodeBuild

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, selecione Perfis.
3. Na caixa de pesquisa, digite codebuild- e escolha a função que foi criada pelo CodePipeline assistente. Para este tutorial, o nome da função é codebuild-hello-world-service-role.
4. Na página Summary (Resumo), escolha Attach policies (Associar políticas).
5. Selecione a caixa à esquerda da AmazonEC2ContainerRegistryPowerUser política e escolha Anexar política.

Etapa 4: Testar o pipeline

Seu pipeline deve ter tudo para executar uma implantação AWS contínua nativa de ponta a ponta. Agora, teste a funcionalidade enviando uma alteração de código ao repositório de origem.

Para testar o pipeline

1. Faça uma alteração no código no repositório de origem configurado, confirme e envie a alteração.
2. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
3. Escolha o pipeline na lista.
4. Observe a evolução do pipeline pelos respectivos estágios. O pipeline será concluído e o serviço do Amazon ECS executará a imagem do Docker criada a partir da alteração do código.

Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR

Neste tutorial, você configura um pipeline AWS CodePipeline que implanta aplicativos de contêiner usando uma blue/green implantação que suporta imagens do Docker. Em uma blue/green implantação, você pode iniciar a nova versão do seu aplicativo junto com a versão antiga e testar a nova versão antes de redirecionar o tráfego. Você também poderá monitorar o processo de implantação e realizar uma reversão rapidamente se houver algum problema.

Important

Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Note

Este tutorial é para a ação de CodeDeploy blue/green implantação do Amazon ECS. CodePipeline Para ver um tutorial que usa a ação de implantação padrão do Amazon ECS em CodePipeline, consulteTutorial: Implantação padrão do Amazon ECS com CodePipeline.

O pipeline concluído detecta alterações em sua imagem, que é armazenada em um repositório de imagens, como o Amazon ECR, e é usada CodeDeploy para rotear e implantar tráfego em um cluster e balanceador de carga do Amazon ECS. CodeDeploy usa um ouvinte para redirecionar o tráfego para a porta do contêiner atualizado especificado no AppSpec arquivo. Para obter informações sobre como o balanceador de carga, o ouvinte de produção, os grupos-alvo e seu aplicativo Amazon ECS são usados em uma blue/green implantação, consulte Tutorial: Implantar um serviço do Amazon ECS.

O pipeline também é configurado para usar um local de origem, como, por exemplo CodeCommit, onde sua definição de tarefa do Amazon ECS está armazenada. Neste tutorial, você configura cada um desses AWS recursos e, em seguida, cria seu funil com estágios que contêm ações para cada recurso.

O pipeline de entrega contínua compilará e implantará automaticamente as imagens de contêiner sempre que o código-fonte for alterado ou for feito upload de uma nova imagem de base no Amazon ECR.

Esse fluxo usa os seguintes artefatos:

- Um arquivo de imagem do Docker que especifica o nome do contêiner e a URI do repositório de imagens do Amazon ECR.
- Uma definição de tarefa do Amazon ECS que lista o nome da imagem do Docker, o nome do contêiner, o nome do serviço do Amazon ECS e a configuração do balanceador de carga.
- Um CodeDeploy AppSpec arquivo que especifica o nome do arquivo de definição de tarefas do Amazon ECS, o nome do contêiner do aplicativo atualizado e a porta do contêiner para a qual CodeDeploy redireciona o tráfego de produção. Também é capaz de especificar a configuração de rede opcional e as funções Lambda que podem ser executadas durante os ganchos do evento de ciclo de vida da implantação.

> **Note**
>
> Quando você confirmar uma alteração no repositório de imagens do Amazon ECR, a ação de origem do pipeline criará um arquivo `imageDetail.json` para essa confirmação. Para mais informações sobre o arquivo `imageDetail.json`, consulte Detail.json arquivo de imagem para ações de blue/green implantação do Amazon ECS.

Ao criar ou editar seu pipeline e atualizar ou especificar artefatos de origem para o estágio de implantação, certifique-se de apontar para os artefatos de origem com o nome e a versão mais recente que deseja utilizar. Depois de configurar seu pipeline, à medida que forem feitas alterações em sua imagem ou definição de tarefa, pode ser necessário atualizar os arquivos do artefato de origem em seus repositórios e editar o estágio de implantação em seu pipeline.

Tópicos

- Pré-requisitos
- Etapa 1: Criar uma imagem e enviá-la a um repositório do Amazon ECR
- Etapa 2: Crie arquivos de definição de tarefas e de AppSpec origem e envie para um CodeCommit repositório
- Etapa 3: Criar o Application Load Balancer e os grupos de destino

- Etapa 4: Criar um cluster e um serviço do Amazon ECS
- Etapa 5: Crie seu CodeDeploy aplicativo e grupo de implantação (plataforma de computação ECS)
- Etapa 6: Criar o pipeline
- Etapa 7: Realizar uma alteração no pipeline e verificar a implantação

Pré-requisitos

É necessário que já tenham sido criados os recursos a seguir:

- Um CodeCommit repositório. Você pode usar o AWS CodeCommit repositório no Tutorial: criar um pipeline simples (CodeCommit repositório) qual você criou.
- Inicie uma instância do Linux do Amazon EC2 e instale o Docker para criar uma imagem, conforme mostrado neste tutorial. Caso já exista uma imagem que deseja usar, ignore esse pré-requisito.

Etapa 1: Criar uma imagem e enviá-la a um repositório do Amazon ECR

Nesta seção, você usa o Docker para criar uma imagem e, em seguida, usa o AWS CLI para criar um repositório Amazon ECR e enviar a imagem para o repositório.

> **Note**
>
> Caso já exista uma imagem que deseja usar, pule esta etapa.

Criar uma imagem

1. Conecte-se à sua instância do Linux na qual o Docker esteja instalado.

   Expanda uma imagem para `nginx`. Esse comando fornece a imagem `nginx:latest`:

   ```
   docker pull nginx
   ```

2. Executar docker images. A imagem deve estar presente na lista.

   ```
   docker images
   ```

Para criar um repositório do Amazon ECR e enviar a imagem

1. Crie um repositório do Amazon ECR para armazenar sua imagem . Anote o `repositoryUri` apresentado na saída.

   ```
   aws ecr create-repository --repository-name nginx
   ```

   Saída:

   ```
   {
       "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. Marque a imagem com o valor `repositoryUri` da etapa anterior.

   ```
   docker tag nginx:latest aws_account_id.dkr.ecr.us-east-1.amazonaws.com/nginx:latest
   ```

3. Execute o comando aws ecr get-login-password, como mostrado neste exemplo para a região `us-west-2` e o ID da conta 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. Envie a imagem ao Amazon ECR utilizando o `repositoryUri` da etapa anterior.

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

Etapa 2: Crie arquivos de definição de tarefas e de AppSpec origem e envie para um CodeCommit repositório

Nesta seção, você criará um arquivo JSON de definição da tarefa, registrando-o no Amazon ECS. Em seguida, você cria um AppSpec arquivo CodeDeploy e usa seu cliente Git para enviar os arquivos para o seu CodeCommit repositório.

Criar uma definição de tarefa para sua imagem

1. Crie um arquivo denominado `taskdef.json` com os conteúdos a seguir. Para `image`, insira o nome da imagem, como nginx. Esse valor é atualizado quando o pipeline é executado.

> **Note**
>
> Certifique-se de que a função de execução especificada na definição de tarefas contenha a `AmazonECSTaskExecutionRolePolicy`. Para obter mais informações, consulte Perfil do IAM de execução de tarefas do Amazon ECS no Guia do desenvolvedor do 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 sua definição de tarefas com o arquivo `taskdef.json`.

```
aws ecs register-task-definition --cli-input-json file://taskdef.json
```

3. Após o registro da definição de tarefas, edite o arquivo para remover o nome da imagem e incluir o texto do espaço reservado <IMAGE1_NAME> no campo de imagem.

```
{
    "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 criar um AppSpec arquivo

- O AppSpec arquivo é usado para CodeDeploy implantações. O arquivo, que inclui campos opcionais, usa o seguinte 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 obter mais informações sobre o AppSpec arquivo, incluindo exemplos, consulte Referência CodeDeploy AppSpec do arquivo.

Crie um arquivo denominado `appspec.yaml` com os conteúdos a seguir. Para `TaskDefinition`, não altere o texto do espaço reservado `<TASK_DEFINITION>`. Esse valor é atualizado quando o pipeline é executado.

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: <TASK_DEFINITION>
        LoadBalancerInfo:
          ContainerName: "sample-website"
          ContainerPort: 80
```

Para enviar arquivos para o seu CodeCommit repositório

1. Envie ou faça o upload dos arquivos para o seu CodeCommit repositório. Esses arquivos são o artefato de origem gerado pelo assistente Create pipeline (Criar pipeline) para a ação de implantação no CodePipeline. Os arquivos devem ter a seguinte aparência em seu diretório local:

```
/tmp
```

```
|my-demo-repo
  |-- appspec.yaml
  |-- taskdef.json
```

2. Selecione o método que deseja utilizar para carregar seus arquivos:

   a. Usar a linha de comando git a partir de um repositório clonado no computador local:

      i. Altere diretórios para o repositório local:

      ```
      (For Linux, macOS, or Unix) cd /tmp/my-demo-repo
      (For Windows) cd c:\temp\my-demo-repo
      ```

      ii. Execute o seguinte comando para organizar todos os seus arquivos de uma só vez:

      ```
      git add -A
      ```

      iii. Execute o seguinte comando para confirmar os arquivos com uma mensagem de confirmação:

      ```
      git commit -m "Added task definition files"
      ```

      iv. Execute o comando a seguir para enviar os arquivos do seu repositório local para o seu CodeCommit repositório:

      ```
      git push
      ```

   b. Para usar o CodeCommit console para fazer upload de seus arquivos:

      i. Abra o CodeCommit console e escolha seu repositório na lista Repositórios.

      ii. Selecione Add file (Adicionar arquivo) e clique em Upload file (Carregar arquivo).

      iii. Clique em Choose file (Selecionar arquivo) e localize o arquivo. Informe seu nome de usuário e endereço de e-mail para confirmar a alteração. Escolha Commit changes (Confirmar alterações).

      iv. Repita essa etapa para cada arquivo que deseja carregar.

Etapa 3: Criar o Application Load Balancer e os grupos de destino

Nesta seção, você criará um Application Load Balancer do Amazon EC2. É possível utilizar os nomes de sub-rede e valores do grupo de destino gerados com o balanceador de carga

posteriormente, ao criar o serviço do Amazon ECS. É possível utilizar um Application Load Balancer ou Network Load Balancer. O load balancer deve usar uma VPC com duas sub-redes públicas em diferentes zonas de disponibilidade. Nessas etapas, confirme sua VPC padrão, crie um load balancer e crie dois grupos de destino para seu load balancer. Para obter mais informações, consulte Grupos de destino para seus load balancers de rede.

Verificar sua VPC padrão e sub-redes públicas

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon VPC em https://console.aws.amazon.com/vpc/.
2. Verifique a VPC padrão a ser usada. No painel de navegação, escolha Your VPCs (Suas VPCs). Observe qual VPC exibe Sim na coluna de VPC padrão. Essa é a VPC padrão. Contém sub-redes padrão a serem selecionadas.
3. Selecione Subnets (Sub-redes). Selecione duas sub-redes que exibam Yes (Sim) na coluna de Default subnet (Sub-rede padrão).

   > **Note**
   >
   > Anote seus IDs de sub-rede. Você precisará deles posteriormente neste tutorial.

4. Selecione as sub-redes e escolha a guia Description (Descrição). Verifique se as sub-redes que você deseja utilizar se encontram em diferentes zonas de disponibilidade.
5. Selecione as sub-redes e escolha a guia Route Table (Tabela de rotas). Para verificar se cada sub-rede que deseja utilizar é uma sub-rede pública, confirme se uma linha de gateway está presente na tabela de rotas.

Para criar um Application Load Balancer do Amazon EC2

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. No painel de navegação, selecione Load Balancers.
3. Selecione Criar load balancer.
4. Selecione Application Load Balancer e clique em Create (Criar).
5. Em Name (Nome), informe o nome do load balancer.
6. Em Scheme (Esquema), selecione Internet-facing (Voltado para a Internet).
7. Em IP address type (Tipo de endereço IP), selecione ipv4.

8. Configure duas portas do listener para seu load balancer:

    a. Em Load Balancer Protocol (Protocolo do load balancer), selecione HTTP. Em Load Balancer Port (Porta do load balancer), insira **80**.

    b. Escolha Add listener.

    c. Em Load Balancer Protocol (Protocolo do load balancer) para o segundo listener, selecione HTTP. Em Load Balancer Port (Porta do load balancer), insira **8080**.

9. Em Availability Zones (Zonas de disponibilidade), em VPC, selecione a VPC padrão. A seguir, escolha as duas sub-redes padrão que deseja utilizar.
10. Selecione Next: Configure Security Settings (Próximo: Definir configurações de segurança).
11. Selecione Next: Configure Security Groups (Próximo: Configurar grupos de segurança).
12. Clique em Select an existing security group (Selecionar um grupo de segurança existente) e anote o ID do grupo de segurança.
13. Selecione Next: Configure Routing (Próximo: Configurar roteamento).
14. Em Target group (Grupo de destino), selecione New target group (Novo grupo de destino) e configure o primeiro grupo de destino:

    a. Em Nome, informe o nome do grupo de destino (por exemplo, **target-group-1**).

    b. Em Tipo de destino, selecione IP.

    c. Em Protocolo: Selecione HTTP. Em Porta, insira **80**.

    d. Selecione Next: Register Targets (Próximo: Registrar destinos).

15. Selecione Next: Review (Próximo: Análise) e Create (Criar).

Criar um grupo de destino para seu load balancer.

1. Após a provisão do balanceador de carga, abra o console do Amazon EC2. No painel de navegação, selecione Grupos de destino.
2. Selecione Criar grupo de destino.
3. Em Nome, informe o nome do grupo de destino (por exemplo, **target-group-2**).
4. Em Tipo de destino, selecione IP.
5. Em Protocolo: Selecione HTTP. Em Porta, insira **8080**.
6. Em VPC, escolha a VPC padrão.
7. Escolha Criar.

> **Note**
>
> Para que sua implantação seja executada, é necessário ter dois grupos de destino criados para seu load balancer. Somente é necessário anotar o ARN do seu primeiro grupo de destino. O ARN é usado no arquivo JSON `create-service` na próxima etapa.

Atualizar o load balancer para incluir o segundo grupo de destino

1. Abra o console do Amazon EC2. No painel de navegação, selecione Load Balancers.
2. Selecione seu load balancer e clique na guia Listeners. Escolha o listener com a porta 8080 e selecione Edit (Editar).
3. Selecione o ícone de lápis próximo a Forward to (Avançar para) Escolha o segundo grupo de destino e selecione a marca de verificação. Selecione Update (Atualizar) para salvar as atualizações.

Etapa 4: Criar um cluster e um serviço do Amazon ECS

Nesta seção, você cria um cluster e um serviço do Amazon ECS que direciona o CodeDeploy tráfego durante a implantação (para um cluster do Amazon ECS em vez de instâncias do EC2). Para criar o serviço do Amazon ECS, é necessário utilizar os nomes de sub-redes, o grupo de segurança e o valor do grupo de destino gerados com o balanceador de carga para criar o serviço.

> **Note**
>
> Ao usar essas etapas para criar seu cluster do Amazon ECS, você usa o modelo de cluster Networking only, que provisiona contêineres AWS Fargate. AWS O Fargate é uma tecnologia que gerencia sua infraestrutura de instância de contêiner para você. Não é necessário selecionar nem criar manualmente instâncias do Amazon EC2 para o cluster do Amazon ECS.

Para criar um cluster do Amazon ECS

1. Abra o console clássico do Amazon ECS em https://console.aws.amazon.com/ecs/.

2. No painel de navegação, escolha Clusters.
3. Selecione Criar cluster.
4. Selecione o modelo de cluster Somente redes que utiliza o AWS Fargate e, depois, selecione Próxima etapa.
5. Insira um nome de cluster na página Configure cluster (Configurar cluster). Você pode adicionar uma tag opcional para o seu recurso. Escolha Criar.

Para criar um serviço do Amazon ECS

Use o AWS CLI para criar seu serviço no Amazon ECS.

1. Crie um arquivo JSON e o nomeie como `create-service.json`. Cole a seguinte informação no arquivo JSON.

   No campo `taskDefinition`, ao registrar uma definição de tarefa no Amazon ECS, você atribui uma família a ela. Isso é semelhante a um nome para várias versões da definição da tarefa, especificado com um número de revisão. Neste exemplo, use "`ecs-demo:1`" para a família e o número de revisão no seu arquivo. Use os nomes de sub-rede, o grupo de segurança e valor do grupo de destino que você criou com seu load balancer em Etapa 3: Criar o Application Load Balancer e os grupos de destino.

   > **Note**
   >
   > É necessário incluir o ARN do grupo de destino nesse arquivo. Abra o console do Amazon EC2 e, no painel de navegação, em BALANCEAMENTO DE CARGA, selecione Grupos de destino. Escolha o primeiro grupo de destino. Copie o ARN da guia Description (Descrição).

```
{
    "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. Execute o comando create-service especificando o arquivo JSON:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

Este exemplo cria um serviço denominado `my-service`.

> **Note**
>
> Este comando de exemplo cria um serviço denominado my-service. Se você já tem um serviço com esse nome, o comando retornará um erro.

```
aws ecs create-service --service-name my-service --cli-input-json file://create-service.json
```

A saída retorna os campos de descrição para seu serviço.

3. Execute o comando describe-services para verificar se o serviço foi criado corretamente.

```
aws ecs describe-services --cluster cluster-name --services service-name
```

Etapa 5: Crie seu CodeDeploy aplicativo e grupo de implantação (plataforma de computação ECS)

Quando você cria um CodeDeploy aplicativo e um grupo de implantação para a plataforma computacional Amazon ECS, o aplicativo é usado durante uma implantação para referenciar o grupo de implantação correto, os grupos-alvo, os ouvintes e o comportamento de redirecionamento de tráfego.

Para criar um CodeDeploy aplicativo

1. Abra o CodeDeploy console e escolha Criar aplicativo.
2. Em Application name (Nome do aplicativo), informe o nome que deseja utilizar.
3. Em Compute platform (Plataforma de computação), selecione Amazon ECS.
4. Escolha Criar aplicativo.

Para criar um grupo CodeDeploy de implantação

1. Na guia da página do aplicativo Deployment groups (Grupos de implantação), selecione Create deployment group (Criar grupo de implantação).
2. Em Nome do grupo de implantação digite um nome que descreva o grupo de implantação.
3. Em Função de serviço, escolha uma função de serviço que conceda CodeDeploy acesso ao Amazon ECS. Para criar uma nova função de serviço, siga estas etapas:

   a. Abra o console do IAM em https://console.aws.amazon.com/iam/.
   b. No painel do console, escolha Roles (Funções).
   c. Selecione Criar perfil.
   d. Em Selecionar tipo de entidade confiável, selecione AWS service (Serviço da AWS). Em Escolha um caso de uso, selecione CodeDeploy. Em Selecione seu caso de uso, selecione CodeDeploy - ECS. Escolha Próximo: Permissões. A política gerenciada `AWSCodeDeployRoleForECS` já está anexada à função.
   e. Selecione Next: Tags (Próximo: tags) e Next: Review (Próximo: revisar).

f. Insira um nome para a função (por exemplo, **`CodeDeployECSRole`**) e escolha Create role (Criar função).

4. Em Configuração de ambiente, selecione os nomes de cluster e serviço do Amazon ECS.
5. Em Balanceadores de carga, escolha o nome do balanceador de carga que distribui o tráfego para seu serviço do Amazon ECS.
6. Em Porta do receptor de produção, escolha a porta e o protocolo para o receptor que fornece o tráfego de produção para seu serviço do Amazon ECS. Em Test listener port (Porta do listener de teste), escolha a porta e o protocolo para o listener de teste.
7. Em Nome do grupo de destino 1 e Nome do grupo de destino 2, escolha os grupos de destino utilizados para rotear o tráfego durante a implantação. Certifique-se de que esses são os grupos de destino criados para o load balancer.
8. Selecione Redirecionar o tráfego imediatamente para determinar por quanto tempo após uma implantação bem-sucedida será possível redirecionar o tráfego à tarefa atualizada do Amazon ECS.
9. Selecione Criar grupo de implantação.

Etapa 6: Criar o pipeline

Nesta seção, você criará um pipeline com as seguintes ações:

- Uma CodeCommit ação em que os artefatos de origem são a definição da tarefa e o AppSpec arquivo.
- Um estágio de origem com uma ação de origem do Amazon ECR na qual o artefato de origem é o arquivo de imagem.
- Um estágio de implantação com uma ação de implantação do Amazon ECS em que a implantação é executada com um CodeDeploy aplicativo e um grupo de implantação.

Criar um pipeline de dois estágios com o assistente

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).

3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyImagePipeline`**.
5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.
6. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.
7. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
8. Em Etapa 3: Adicionar etapa de origem, em Provedor de origem, escolha AWS CodeCommit. Em Repository name (Nome do repositório), selecione o nome do repositório do CodeCommit que você criou em Etapa 1: criar um CodeCommit repositório. Em Nome do ramo, selecione o nome do ramo que contém a última atualização do código.

   Escolha Próximo.
9. Em Etapa 4: Adicionar etapa de compilação, escolha Ignorar etapa de compilação e aceite a mensagem de aviso escolhendo Ignorar novamente. Escolha Próximo.
10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.
11. Na Etapa 6: Adicionar estágio de implantação:

    a. Em Implantar provedor, escolha Amazon ECS (Blue/Green). Em Application name (Nome do aplicativo), informe ou selecione o nome de um aplicativo da lista, como `codedeployapp`. Em Deployment group (Grupo de implantação), informe ou selecione o nome de um grupo de implantação da lista, como `codedeploydeplgroup`.

> **Note**
>
> O nome "Deploy" é o nome padrão dado ao estágio criado em Step 4: Deploy (Etapa 4: implantar), assim como "Source" ("Origem) é o nome dado ao primeiro estágio do pipeline.

b. Em Definição de tarefas do Amazon ECS, escolha SourceArtifact. No campo, insira **`taskdef.json`**.

c. Em AWS CodeDeploy AppSpec Arquivo, escolha SourceArtifact. No campo, insira **`appspec.yaml`**.

> **Note**
>
> Nesse momento, não forneça informações em Dynamically update task definition image (Atualização dinâmica da imagem de definição de tarefas).

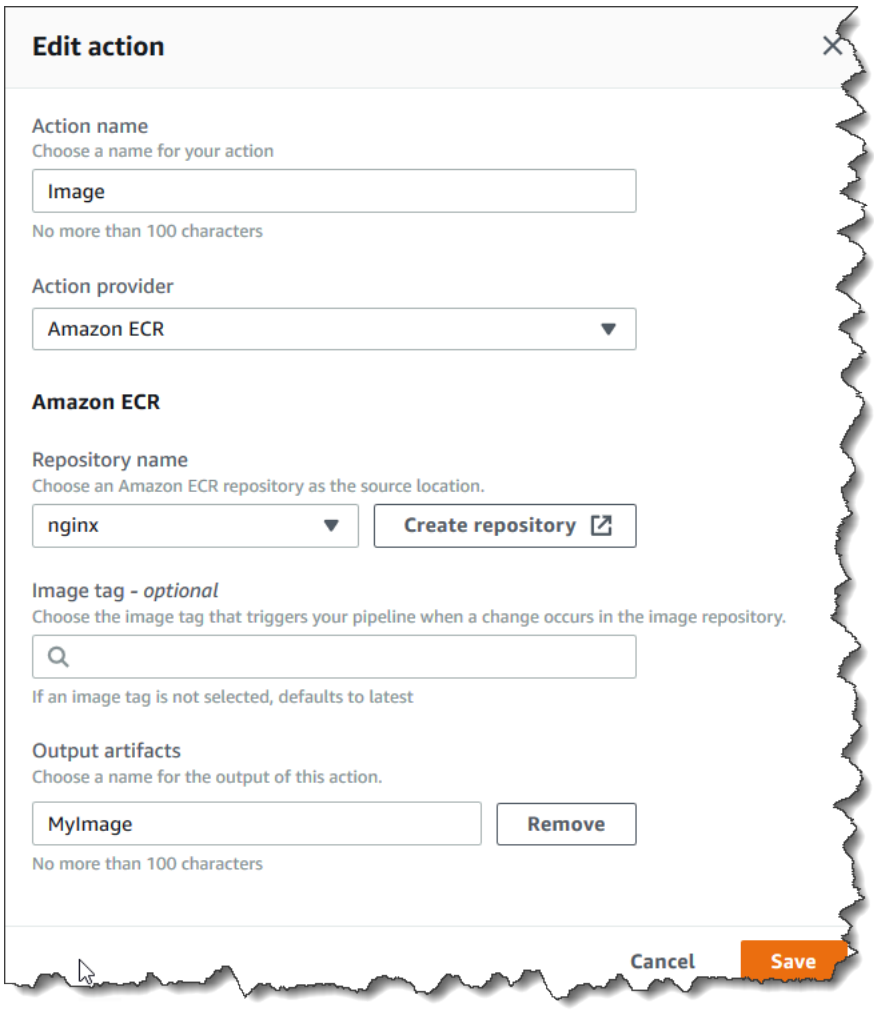
d. Escolha Próximo.

12. Na Etapa 7: Revisar, revise as informações e escolha Criar funil.

Como adicionar uma ação de origem do Amazon ECR ao pipeline

Visualize o pipeline e adicione uma ação de origem do Amazon ECR ao pipeline.

1. Selecione seu pipeline. No canto superior esquerdo, selecione Edit (Editar).
2. No estágio de origem, clique em Edit stage (Editar estágio).
3. Adicione uma ação paralela escolhendo + Adicionar ação ao lado da ação CodeCommit de origem.
4. Em Action name (Nome da ação), informe um nome (por exemplo, **`Image`**).
5. Em Action provider (Provedor de ação), selecione Amazon ECR.



6. Em Nome do repositório, selecione o nome do repositório do Amazon ECR.
7. Em Image tag (Tag da imagem), especifique o nome da imagem e a versão, caso seja diferente da última.
8. Em Output artifacts (Artefatos de saída), escolha o artefato de saída padrão (por exemplo, `MyImage`) que contém o nome da imagem e as informações de URI do repositório que o próximo estágio deve utilizar.
9. Escolha Save (Salvar) na tela de ação. Escolha Done (Concluído) na tela de estágio. Escolha Save (Salvar) no pipeline. Uma mensagem mostra a regra do Amazon CloudWatch Events a ser criada para a ação de origem do Amazon ECR.

Como conectar os artefatos de origem à ação de implantação

1. Escolha Editar em seu estágio de implantação e escolha o ícone para editar a ação do Amazon ECS (Blue/Green).

2. Role até a parte inferior do painel. Em Input artifacts (Artefatos de entrada), selecione Add (Adicionar). Adicione o artefato de origem do novo repositório do Amazon ECR (por exemplo, `MyImage`).
3. Em Definição de Tarefa, escolha e SourceArtifact, em seguida, verifique **`taskdef.json`** se foi inserido.
4. Em AWS CodeDeploy AppSpec Arquivo, escolha e SourceArtifact, em seguida, verifique **`appspec.yaml`** se foi inserido.
5. Em Atualizar dinamicamente a imagem de definição de tarefa, em Inserir artefato com URI de imagem MyImage, escolha e insira o texto do espaço reservado usado no `taskdef.json` arquivo:. **`IMAGE1_NAME`** Escolha Salvar.
6. No AWS CodePipeline painel, escolha Salvar alteração do pipeline e, em seguida, selecione Salvar alteração. Visualize seu pipeline atualizado.

   Depois que esse pipeline de exemplo é criado, a configuração da ação para as entradas do console é exibida na estrutura do pipeline da seguinte forma:

```
"configuration": {
  "AppSpecTemplateArtifact": "SourceArtifact",
  "AppSpecTemplatePath": "appspec.yaml",
  "TaskDefinitionTemplateArtifact": "SourceArtifact",
  "TaskDefinitionTemplatePath": "taskdef.json",
  "ApplicationName": "codedeployapp",
  "DeploymentGroupName": "codedeploydeplgroup",
  "Image1ArtifactName": "MyImage",
  "Image1ContainerName": "IMAGE1_NAME"
},
```

7. Para enviar suas alterações e iniciar uma compilação do pipeline, selecione Liberar alteração e, depois, Liberar.
8. Escolha a ação de implantação para visualizá-la CodeDeploy e ver o progresso da mudança de tráfego.

> **Note**
>
> Você pode ver uma etapa de implantação que mostra um tempo de espera opcional. Por padrão, CodeDeploy espera uma hora após uma implantação bem-sucedida antes de encerrar o conjunto de tarefas original. Você pode usar esse tempo para reverter

ou encerrar a tarefa, mas a sua implantação só será concluída quando o conjunto de tarefas for encerrado.

Etapa 7: Realizar uma alteração no pipeline e verificar a implantação

Faça uma alteração na imagem e, depois, envie-a ao repositório do Amazon ECR. Deste modo, a execução de seu pipeline é acionada. Verifique se a alteração da imagem de origem foi implantada.

Tutorial: Criar um pipeline que implanta uma skill do Amazon Alexa

Nesse tutorial, você configura um pipeline que fornece continuamente sua skill da Alexa usando o Alexa Skills Kit como o provedor de implantação em seu estágio de implantação. O pipeline concluído detecta alterações em sua skill quando você faz uma alteração nos arquivos de origem em seu repositório de origem. O pipeline usa o Alexa Skills Kit para implantar no estágio de desenvolvimento de skills da Alexa.

Important

Como parte da criação de um pipeline, um bucket de artefatos S3 fornecido pelo cliente será usado CodePipeline por for artefacts. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 Contas da AWS seja de propriedade de quem é seguro e confiável.

Note

Este atributo não está disponível na região Ásia-Pacífico (Hong Kong) ou Europa (Milão). Para usar outras ações de implantação disponíveis nessa região, consulte Implantar integrações de ações.

Para criar sua habilidade personalizada como uma função Lambda, consulte Hospedar uma habilidade personalizada como uma função Lambda AWS. Você também pode criar um pipeline que usa arquivos de origem do Lambda e um CodeBuild projeto para implantar alterações no Lambda para sua habilidade.

Pré-requisitos

Você já deve ter o seguinte:

- Um CodeCommit repositório. Você pode usar o AWS CodeCommit repositório em Tutorial: criar um pipeline simples (CodeCommit repositório) que você criou.
- Uma conta de desenvolvedor da Amazon. Essa é a conta que possui suas skills na Alexa. Você pode criar uma conta gratuitamente no Alexa Skills Kit.
- Uma skill da Alexa. Você pode criar um exemplo de skill usando o tutorial Obter código de exemplo da skill personalizada.
- Instale a CLI do ASK e configure-a usando `ask init` com as credenciais da AWS . Consulte Instalar e inicializar a CLI do ASK.

Etapa 1: Criar um perfil de segurança do LWA dos serviços de desenvolvedor da Alexa

Nessa seção, crie um perfil de segurança para usar com o Login with Amazon (LWA). Se você já tiver um perfil, ignore esta etapa.

- Use as etapas em generate-lwa-tokens para criar um perfil de segurança.
- Depois de criar o perfil, anote o Client ID (ID do cliente) e o Client Secret (Segredo do cliente).
- Certifique-se de inserir a devolução permitida URLs conforme fornecido nas instruções. Eles URLs permitem que o comando ASK CLI redirecione as solicitações de token de atualização.

Etapa 2: Crie arquivos de origem de habilidades da Alexa e envie para seu repositório CodeCommit

Nessa seção, você cria e envia seus arquivos de origem de skills da Alexa para o repositório que o pipeline usa para o estágio de origem. Para a skill que você criou no console do desenvolvedor da Amazon, você produz e envia o seguinte:

- Um arquivo `skill.json`.
- Uma pasta `interactionModel/custom`.

> **Note**
>
> Essa estrutura de diretório está em conformidade com os requisitos de formato do pacote de skills do Alexa Skills Kit, conforme descrito em Formato do pacote de skills. Se a estrutura do diretório não usar o formato correto do pacote de skills, as alterações não serão implantadas com êxito no console do Alexa Skills Kit.

Como criar arquivos de origem para sua skill

1. Recupere o ID da skill do console do desenvolvedor do Alexa Skills Kit. Use este comando:

   ```
   ask api list-skills
   ```

   Localize a skill por nome e copie o ID associado no campo `skillId`.

2. Gere um arquivo `skill.json` que contém os detalhes da skill. Use este comando:

   ```
   ask api get-skill -s skill-ID > skill.json
   ```

3. (Opcional) Crie uma pasta `interactionModel/custom`.

   Use esse comando para gerar o arquivo de modelo de interação dentro da pasta. Para localidade, esse tutorial usa en-US como a localidade no nome do arquivo.

   ```
   ask api get-model --skill-id skill-ID --locale locale >
       ./interactionModel/custom/locale.json
   ```

Para enviar arquivos para o seu CodeCommit repositório

1. Envie ou envie os arquivos para o seu CodeCommit repositório. Esses arquivos são o artefato de origem gerado pelo assistente Create Pipeline (Criar pipeline) para a ação de implantação no AWS CodePipeline. Os arquivos devem ter a seguinte aparência em seu diretório local:

   ```
   skill.json
   /interactionModel
     /custom
       |en-US.json
   ```

2. Selecione o método que deseja utilizar para carregar seus arquivos:

   a. Para usar a linha de comando Git a partir de um repositório clonado no computador local:

      i. Execute o seguinte comando para organizar todos os seus arquivos de uma só vez:

      ```
      git add -A
      ```

      ii. Execute o seguinte comando para confirmar os arquivos com uma mensagem de confirmação:

      ```
      git commit -m "Added Alexa skill files"
      ```

      iii. Execute o comando a seguir para enviar os arquivos do seu repositório local para o seu CodeCommit repositório:

      ```
      git push
      ```

   b. Para usar o CodeCommit console para carregar seus arquivos:

      i. Abra o CodeCommit console e escolha seu repositório na lista Repositórios.

      ii. Selecione Add file (Adicionar arquivo) e clique em Upload file (Carregar arquivo).

      iii. Clique em Choose file (Selecionar arquivo) e localize o arquivo. Informe seu nome de usuário e endereço de e-mail para confirmar a alteração. Escolha Commit changes (Confirmar alterações).

      iv. Repita essa etapa para cada arquivo que deseja carregar.

Etapa 3: Usar os comandos da CLI do ASK para criar um token de atualização

CodePipeline usa um token de atualização com base no ID do cliente e no segredo da sua conta de desenvolvedor da Amazon para autorizar ações que ele executa em seu nome. Nessa seção, você usa a CLI do ASK para criar o token. Você pode usar essas credenciais quando usar o assistente Create Pipeline (Criar pipeline).

Para criar um token de atualização com as credenciais da conta de desenvolvedor da Amazon

1. Use o seguinte comando:

```
ask util generate-lwa-tokens
```

2. Quando solicitado, insira o ID de cliente e o segredo, conforme mostrado neste exemplo:

```
? Please type in the client ID:
amzn1.application-client.example112233445566
? Please type in the client secret:
example112233445566
```

3. A página de login do navegador será exibida. Faça login com as credenciais da conta de desenvolvedor da Amazon.
4. Volte para a tela de linha de comando. O token de acesso e o token de atualização serão gerados na saída. Copie o token de atualização retornado na saída.

Etapa 4: Criar o pipeline

Nesta seção, você criará um pipeline com as seguintes ações:

- Um estágio de origem com uma CodeCommit ação em que os artefatos de origem são os arquivos de habilidades da Alexa que dão suporte à sua habilidade.
- Um estágio de implantação com uma ação de implantação do Alexa Skills Kit.

Criar um pipeline com o assistente

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http:// console.aws.amazon. com/codesuite/codepipeline/home.
2. Escolha a AWS região em que você deseja criar o projeto e seus recursos. O tempo de execução da skill da Alexa está disponível somente nas seguintes regiões:
   - Ásia-Pacífico (Tóquio)
   - Europa (Irlanda)
   - Leste dos EUA (N. da Virgínia)
   - Oeste dos EUA (Oregon)
3. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).

4. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
5. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyAlexaPipeline`**.
6. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços de CodePipeline, consulte Preços.
7. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.
8. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
9. Em Etapa 3: Adicionar etapa de origem, em Provedor de origem, escolha AWS CodeCommit. Em Nome do repositório, escolha o nome do CodeCommit repositório em que você criou. Etapa 1: criar um CodeCommit repositório Em Nome do ramo, selecione o nome do ramo que contém a última atualização do código.

   Depois de selecionar o nome do repositório e a ramificação, uma mensagem mostra a regra Amazon CloudWatch Events a ser criada para esse pipeline.

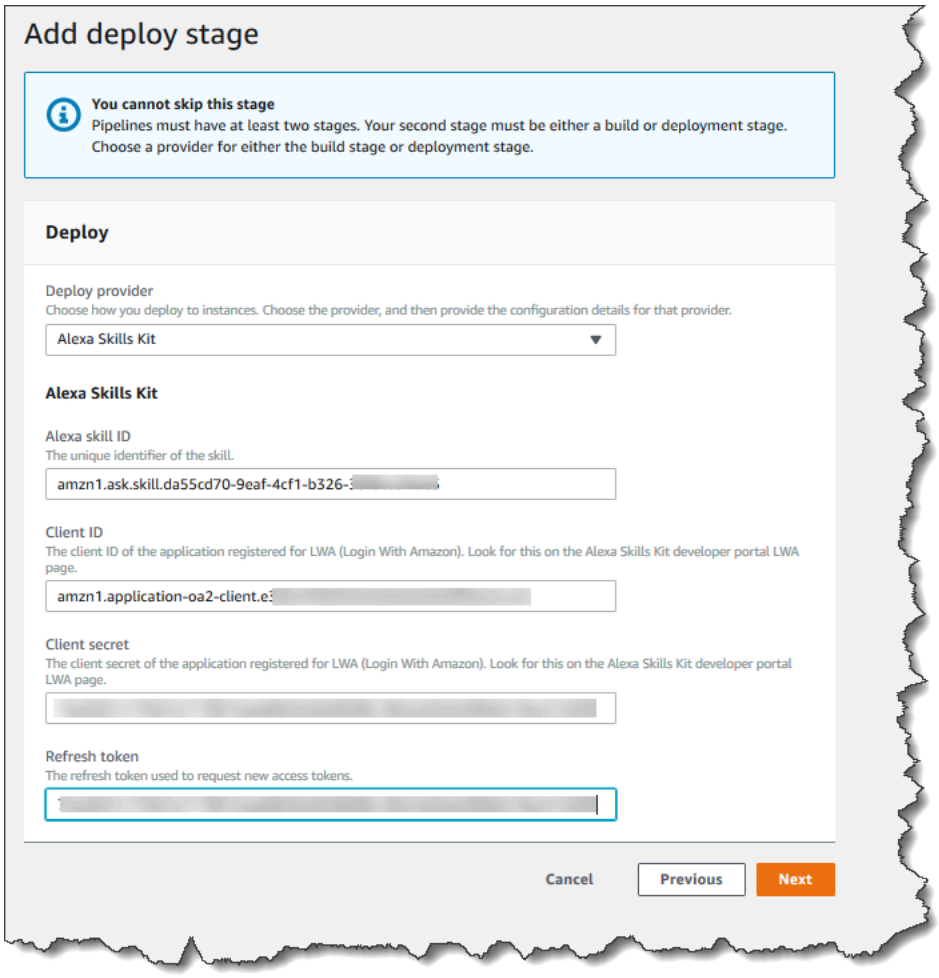
   Escolha Próximo.
10. Em Etapa 4: Adicionar etapa de compilação, escolha Ignorar etapa de compilação e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.
11. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.
12. Na Etapa 6: Adicionar estágio de implantação:

    a. Em Deploy provider (Provedor de implantação), escolha Alexa Skills Kit.
    b. Em Alexa skill ID (ID da skill da Alexa), insira o ID da skill atribuído à sua skill no console do desenvolvedor do Alexa Skills Kit.
    c. Em Client ID (ID do cliente), insira o ID do aplicativo que você registrou.
    d. Em Client secret (Segredo do cliente), insira o segredo que você escolheu quando se registrou.

e. Em Refresh token (Token de atualização), insira o token que você gerou na etapa 3.



f. Escolha Próximo.

13. Na Etapa 7: Revisar, revise as informações e escolha Criar funil.

Etapa 5: Realizar uma alteração em qualquer arquivo de origem e verificar a implantação

Inclua uma alteração em sua skill e envie-a ao repositório. Deste modo, a execução de seu pipeline é acionada. Verifique se sua skill está atualizada no console do desenvolvedor do Alexa Skills Kit.

Tutorial: Criar um pipeline que usa o Amazon S3 como um provedor de implantação

Nesse tutorial, você configurará um pipeline que fornece continuamente arquivos usando o Amazon S3 como o provedor de ação de implantação no estágio de implantação. O pipeline concluído detecta alterações quando você faz uma alteração nos arquivos de origem em seu repositório de origem. O pipeline usa o Amazon S3 para implantar os arquivos no bucket. Sempre que você modifica ou adiciona os arquivos do site no local de origem, a implantação cria o site com os arquivos mais recentes.

Important

Como parte da criação de um pipeline, um bucket de artefatos S3 fornecido pelo cliente será usado CodePipeline por for artefacts. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 Contas da AWS seja de propriedade de quem é seguro e confiável.

Note

Mesmo que você exclua arquivos do repositório de origem, a ação de implantação do S3 não exclui objetos do S3 correspondentes aos arquivos excluídos.

Esse tutorial fornece duas opções:

- Crie um pipeline que implante um site estático no bucket público do S3. Este exemplo cria um pipeline com uma ação de AWS CodeCommit origem e uma ação de implantação do Amazon S3. Consulte Opção 1: Implantar arquivos estáticos de sites no Amazon S3.
- Crie um pipeline que compile o TypeScript código de amostra JavaScript e implante o artefato de CodeBuild saída em seu bucket do S3 para arquivamento. Este exemplo cria um pipeline com uma ação de origem do Amazon S3, uma ação de CodeBuild construção e uma ação de implantação do Amazon S3. Consulte Opção 2: Implantar arquivos de arquivamento compilados no Amazon S3 a partir de um bucket de origem do S3.

> **Important**
>
> Muitas das ações que você adiciona ao pipeline nesse procedimento envolvem AWS recursos que você precisa criar antes de criar o pipeline. AWS os recursos para suas ações de origem sempre devem ser criados na mesma AWS região em que você cria seu pipeline. Por exemplo, se você criar seu pipeline na região Leste dos EUA (Ohio), seu CodeCommit repositório deverá estar na região Leste dos EUA (Ohio).
> Você pode adicionar ações entre regiões ao criar seu pipeline. AWS os recursos para ações entre regiões devem estar na mesma AWS região em que você planeja executar a ação. Para obter mais informações, consulte Adicione uma ação entre regiões em CodePipeline.

Opção 1: Implantar arquivos estáticos de sites no Amazon S3

Neste exemplo, você baixa o arquivo de modelo de site estático de amostra, carrega os arquivos no seu AWS CodeCommit repositório, cria seu bucket e o configura para hospedagem. Em seguida, você usa o AWS CodePipeline console para criar seu pipeline e especificar uma configuração de implantação do Amazon S3.

Pré-requisitos

Você já deve ter o seguinte:

- Um CodeCommit repositório. Você pode usar o AWS CodeCommit repositório em Tutorial: criar um pipeline simples (CodeCommit repositório) que você criou.
- Os arquivos de origem para o site estático. Use este link para fazer download de um exemplo de site estático. O download do sample-website.zip apresenta os seguintes arquivos:
  - Um arquivo `index.html`;
  - Um arquivo `main.css`
  - Um arquivo `graphic.jpg`
- Um bucket do S3 configurado para hospedagem de site. Consulte Hospedagem de um site estático no Amazon S3. Crie o bucket na mesma região do pipeline.

> **Note**
>
> Para hospedar um site, seu bucket deve ter acesso público de leitura, o que dá acesso de leitura a todos. Com exceção da hospedagem de sites, é necessário manter as configurações de acesso padrão que bloqueiam o acesso público aos buckets do S3.

Etapa 1: Envie os arquivos de origem para o seu CodeCommit repositório

Nessa seção, envie seus arquivos de origem para o repositório que o pipeline usa para o estágio de origem.

Para enviar arquivos para o seu CodeCommit repositório

1. Extraia os exemplos de arquivos obtidos por download. Não faça upload do arquivo ZIP para o repositório.
2. Envie ou envie os arquivos para o seu CodeCommit repositório. Esses arquivos são o artefato de origem gerado pelo assistente Create Pipeline (Criar pipeline) para a ação de implantação no CodePipeline. Os arquivos devem ter a seguinte aparência em seu diretório local:

   ```
   index.html
   main.css
   graphic.jpg
   ```

3. Você pode usar o Git ou o CodeCommit console para carregar seus arquivos:

   a. Para usar a linha de comando Git a partir de um repositório clonado no computador local:

      i. Execute o seguinte comando para organizar todos os seus arquivos de uma só vez:

      ```
      git add -A
      ```

      ii. Execute o seguinte comando para confirmar os arquivos com uma mensagem de confirmação:

      ```
      git commit -m "Added static website files"
      ```

      iii. Execute o comando a seguir para enviar os arquivos do seu repositório local para o seu CodeCommit repositório:

```
git push
```

b. Para usar o CodeCommit console para carregar seus arquivos:

i. Abra o CodeCommit console e escolha seu repositório na lista Repositórios.

ii. Selecione Add file (Adicionar arquivo) e clique em Upload file (Carregar arquivo).

iii. Selecione Choose file (Escolher arquivo) e procure o arquivo. Informe seu nome de usuário e endereço de e-mail para confirmar a alteração. Escolha Commit changes (Confirmar alterações).

iv. Repita essa etapa para cada arquivo que deseja carregar.

Etapa 2: Criar o pipeline

Nesta seção, você criará um pipeline com as seguintes ações:

- Um estágio de origem com uma CodeCommit ação em que os artefatos de origem são os arquivos do seu site.
- Um estágio de implantação com uma ação de implantação do Amazon S3.

Criar um pipeline com o assistente

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http:// console.aws.amazon. com/codesuite/codepipeline/home.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyS3DeployPipeline`**.
5. Em Tipo de pipeline, escolha V2. Para obter mais informações, consulte Tipos de pipeline. Escolha Próximo.
6. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.
7. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).

8. Em Etapa 3: Adicionar etapa de origem, em Provedor de origem, escolha AWS CodeCommit. Em Repository name (Nome do repositório), selecione o nome do repositório do CodeCommit que você criou em Etapa 1: criar um CodeCommit repositório. Em Nome do ramo, selecione o nome do ramo que contém a última atualização do código. A não ser que você tenha criado outro ramo, apenas `main` esta disponível.

   Depois de selecionar o nome do repositório e a filial, a regra Amazon CloudWatch Events a ser criada para esse pipeline é exibida.

   Escolha Próximo.

9. Em Etapa 4: Adicionar etapa de compilação, escolha Ignorar etapa de compilação e aceite a mensagem de aviso escolhendo Ignorar novamente.

   Escolha Próximo.

10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

11. Na Etapa 6: Adicionar estágio de implantação:

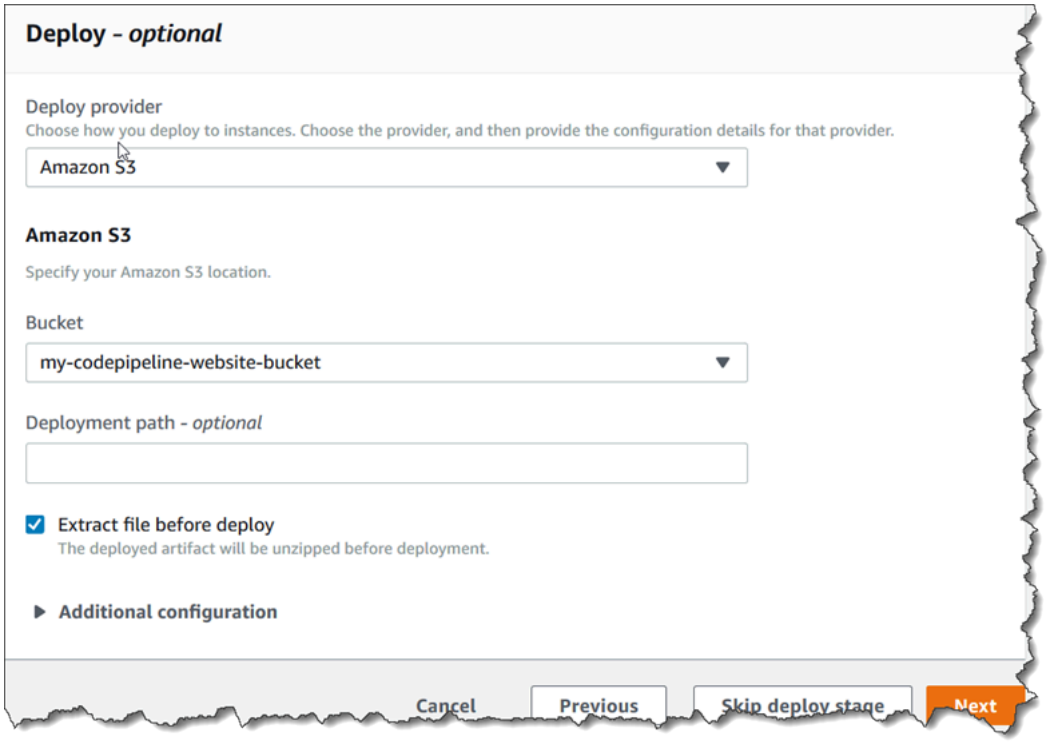
    a. Em Deploy provider (Provedor de implantação), escolha Amazon S3.

    b. Em Bucket, insira o nome do seu bucket público.

    c. Selecione Extract file before deploy (Extrair arquivo antes de implantar).

    > **Note**
    >
    > A implantação falhará se você não selecionar Extrair arquivo antes da implantação. Isso ocorre porque a AWS CodeCommit ação em seu pipeline compacta os artefatos de origem e seu arquivo é um arquivo ZIP.

    Quando Extract file before deploy (Extrair arquivo antes de implantar) for selecionado, o Deployment path (Caminho de implantação) será exibido. Insira no nome do caminho que você deseja usar. Isso cria uma estrutura de pastas no Amazon S3 para a qual os arquivos são extraídos. Para esse tutorial, deixe esse campo em branco.



d. (Opcional) Em Canned ACL (ACL pré-configurada), você pode aplicar um conjunto de concessões predefinidas, conhecido como ACL pré-configurada, aos artefatos carregados.

e. (Opcional) Em Cache control (Controle de cache), insira os parâmetros de armazenamento em cache. Você pode definir isso para controlar o comportamento do armazenamento em cache para solicitações/respostas. Para obter os valores válidos, consulte o campo de cabeçalho `Cache-Control` para operações HTTP.

f. Escolha Próximo.

12. Na Etapa 7: Revisar, revise as informações e escolha Criar funil.

13. Depois que o pipeline for executado com êxito, abra o console do Amazon S3 e verifique se os arquivos aparecem no bucket público, conforme mostrado:

```
index.html
main.css
graphic.jpg
```

14. Acesse o endpoint para testar o site. O endpoint segue este formato: `http://`*`bucket-name`*`.s3-website-`*`region`*`.amazonaws.com/`.

    Exemplo do endpoint: `http://my-bucket.s3-website-us-west-2.amazonaws.com/`.

    A página da web a seguir é exibida.

Etapa 3: Realizar uma alteração em qualquer arquivo de origem e verificar a implantação

Inclua uma alteração nos arquivos de origem e envie-a ao repositório. Deste modo, a execução de seu pipeline é acionada. Verifique se o site está atualizado.

Opção 2: Implantar arquivos de arquivamento compilados no Amazon S3 a partir de um bucket de origem do S3

Nessa opção, os comandos de compilação em seu estágio de compilação compilam o TypeScript código em JavaScript código e implantam a saída no bucket de destino do S3 em uma pasta separada com carimbo de data e hora. Primeiro, você cria o TypeScript código e um arquivo buildspec.yml. Depois de combinar os arquivos de origem em um arquivo ZIP, você carrega o arquivo ZIP de origem no bucket de origem do S3 e usa um CodeBuild estágio para implantar um arquivo ZIP do aplicativo criado no bucket de destino do S3. O código compilado é mantido como um arquivo no bucket de destino.

Pré-requisitos

Você já deve ter o seguinte:

- Um bucket de origem do S3. Você pode usar o bucket que criou em Tutorial: Criar um pipeline simples (bucket do S3).
- Um bucket de destino do S3. Consulte Hospedagem de um site estático no Amazon S3. Certifique-se de criar seu bucket da Região da AWS mesma forma que o pipeline que você deseja criar.

> **Note**
>
> Este exemplo demonstra a implantação de arquivos em um bucket privado. Não ative o bucket de destino para hospedagem de sites nem anexe políticas que tornem o bucket público.

Etapa 1: Criar e fazer upload de arquivos de origem para o bucket de origem do S3

Nessa seção, você cria e faz upload dos arquivos de origem para o bucket que o pipeline usa para seu estágio de origem. Essa seção fornece instruções para criar os seguintes arquivos de origem:

- Um `buildspec.yml` arquivo usado para CodeBuild criar projetos.

- Um arquivo `index.ts`.

criar um arquivo buildspec.yml

- Crie um arquivo denominado `buildspec.yml` com os conteúdos a seguir. Esses comandos de compilação instalam TypeScript e usam o TypeScript compilador para reescrever o código no `index.ts` código. JavaScript

```
version: 0.2

phases:
  install:
    commands:
      - npm install -g typescript
  build:
    commands:
      - tsc index.ts
artifacts:
  files:
    - index.js
```

Criar um arquivo index.ts

- Crie um arquivo denominado `index.ts` com os conteúdos a seguir.

```
interface Greeting {
    message: string;
}

class HelloGreeting implements Greeting {
    message = "Hello!";
}

function greet(greeting: Greeting) {
    console.log(greeting.message);
}

let greeting = new HelloGreeting();

greet(greeting);
```

Como fazer upload de arquivos para o bucket de origem do S3

1. Os arquivos devem ter a seguinte aparência em seu diretório local:

   ```
   buildspec.yml
   index.ts
   ```

   Compacte os arquivos e nomeie o arquivo `source.zip`.
2. No console do Amazon S3, para o bucket de origem, selecione Carregar. Escolha Add files (Adicionar arquivos) e procure o arquivo ZIP que você criou.
3. Escolha Carregar. Esses arquivos são o artefato de origem gerado pelo assistente Create Pipeline (Criar pipeline) para a ação de implantação no CodePipeline. O arquivo deve ter a seguinte aparência no bucket:

   ```
   source.zip
   ```

Etapa 2: Criar o pipeline

Nesta seção, você criará um pipeline com as seguintes ações:

- Um estágio de origem com uma ação do Amazon S3 em que os artefatos de origem são os arquivos da aplicação para download.
- Um estágio de implantação com uma ação de implantação do Amazon S3.

Criar um pipeline com o assistente

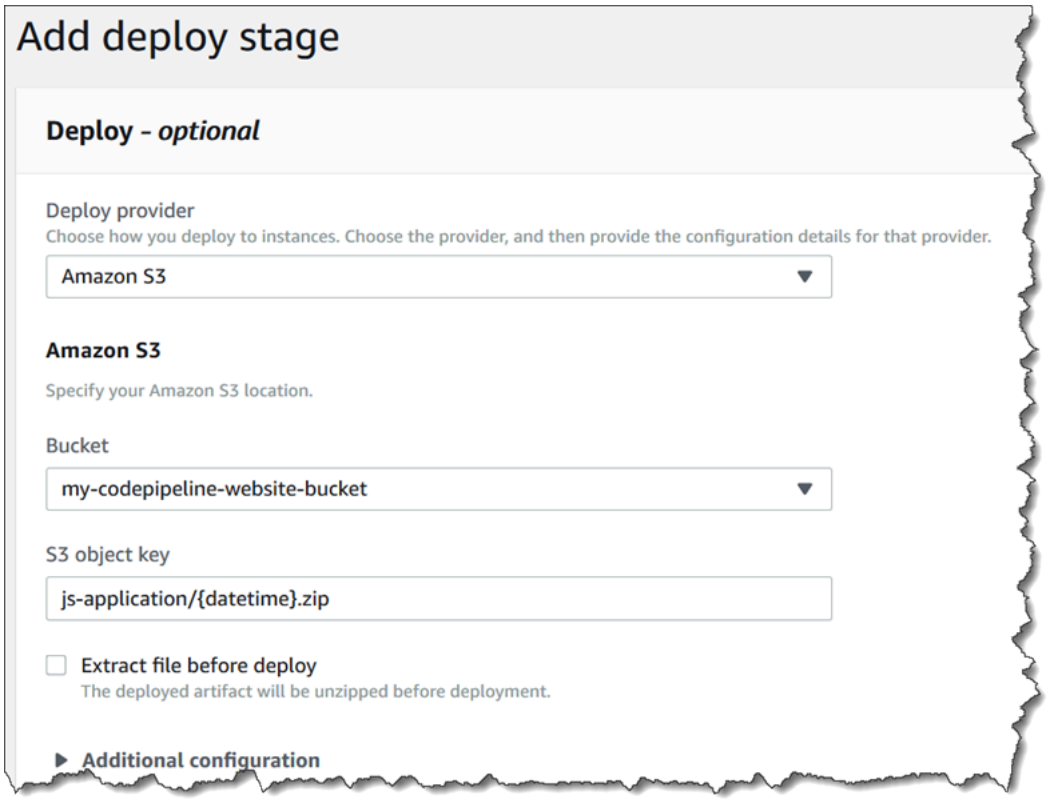
1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyS3DeployPipeline`**.
5. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.

6. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
7. Em Etapa 3: Adicionar etapa de origem, em Provedor de origem, escolha Amazon S3. Em Bucket, escolha o nome do bucket de origem. Na chave de objeto do S3, insira o nome do arquivo ZIP de origem. Certifique-se de incluir a extensão de arquivo .zip.

   Escolha Próximo.
8. Em Etapa 4: adicionar estágio de compilação:
   a. Em Build provider (Provedor de compilação), escolha CodeBuild.
   b. Selecione Create build project (Criar projeto de compilação). Na página Create project (Criar projeto):
   c. Em Nome do projeto, insira um nome para esse projeto de compilação.
   d. Em Environment (Ambiente), escolha Managed image (Imagem gerenciada). Para Operating system, selecione Ubuntu.
   e. Em Runtime (Tempo de execução), selecione Standard (Padrão). Para a versão Runtime, escolha:1.0aws/codebuild/standard.
   f. Na Image version (Versão da imagem), escolha Always use the latest image for this runtime version (Sempre usar a imagem mais recente para esta versão de tempo de execução).
   g. Em Função de serviço, escolha sua função de CodeBuild serviço ou crie uma.
   h. Para Build specifications (Especificações da compilação), escolha Use a buildspec file (Usar um arquivo buildspec).
   i. Escolha Continuar para CodePipeline. Uma mensagem será exibida se o projeto foi criado com sucesso.
   j. Escolha Próximo.
9. Em Etapa 5: Adicionar etapa de implantação:
   a. Em Deploy provider (Provedor de implantação), escolha Amazon S3.
   b. Em Bucket, insira o nome do bucket de destino do S3.
   c. Certifique-se de que Extract file before deploy (Extrair arquivo antes de implantar) esteja desmarcada.

      Quando Extract file before deploy (Extrair arquivo antes de implantar) estiver desmarcado, a S3 object key (chave de objeto do S3) será exibida. Insira no nome do caminho que você deseja usar: `js-application/{datetime}.zip`

Isso cria uma pasta `js-application` no Amazon S3 para a qual os arquivos são extraídos. Nessa pasta, a variável `{datetime}` cria um timestamp em cada arquivo de saída quando o pipeline é executado.



d. (Opcional) Em Canned ACL (ACL pré-configurada), você pode aplicar um conjunto de concessões predefinidas, conhecido como ACL pré-configurada, aos artefatos carregados.

e. (Opcional) Em Cache control (Controle de cache), insira os parâmetros de armazenamento em cache. Você pode definir isso para controlar o comportamento do armazenamento em cache para solicitações/respostas. Para obter os valores válidos, consulte o campo de cabeçalho `Cache-Control` para operações HTTP.

f. Escolha Próximo.

10. Em Etapa 6: Revisar, revise as informações e, então selecione Criar pipeline.

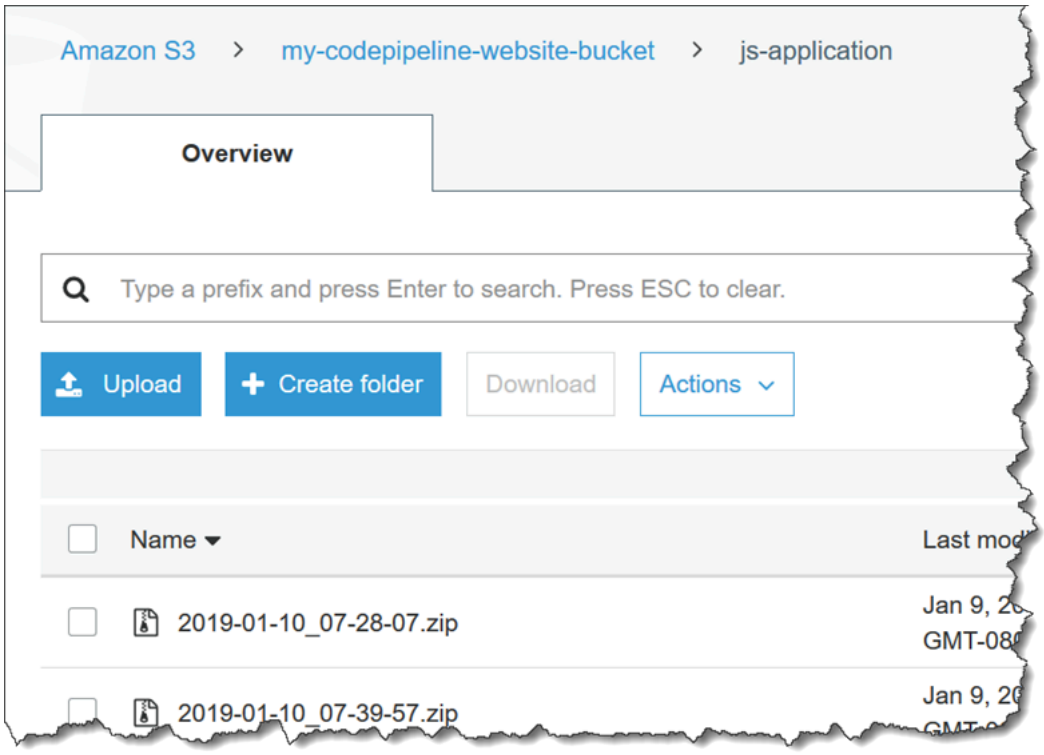
11. Depois que o pipeline for executado com êxito, visualize o bucket no console do Amazon S3. Verifique se o arquivo ZIP implantado é exibido no seu bucket de destino na pasta `js-application`. O JavaScript arquivo contido no arquivo ZIP deve ser`index.js`. O arquivo `index.js` contém a seguinte saída:

```
var HelloGreeting = /** @class */ (function () {
    function HelloGreeting() {
```

```
        this.message = "Hello!";
    }
    return HelloGreeting;
}());
function greet(greeting) {
    console.log(greeting.message);
}
var greeting = new HelloGreeting();
greet(greeting);
```

Etapa 3: Realizar uma alteração em qualquer arquivo de origem e verificar a implantação

Faça uma alteração nos seus arquivos de origem e faça upload deles para o bucket de origem. Deste modo, a execução de seu pipeline é acionada. Visualize seu bucket de destino e verifique se os arquivos de saída implantados estão disponíveis na pasta `js-application`, conforme mostrado:



Tutorial: Crie um pipeline que publique seu aplicativo sem servidor no AWS Serverless Application Repository

Você pode usar AWS CodePipeline para entregar continuamente seu aplicativo AWS SAM sem servidor ao. AWS Serverless Application Repository

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos S3 fornecido pelo cliente será usado CodePipeline por for artefacts. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 Contas da AWS seja de propriedade de quem é seguro e confiável.

Este tutorial mostra como criar e configurar um pipeline para criar seu aplicativo sem servidor hospedado GitHub e publicá-lo automaticamente. AWS Serverless Application Repository O pipeline é usado GitHub como provedor de origem e CodeBuild como provedor de compilação. Para publicar seu aplicativo sem servidor no AWS Serverless Application Repository, você implanta um aplicativo (do AWS Serverless Application Repository) e associa a função Lambda criada por esse aplicativo como um provedor de ação Invoke em seu pipeline. Em seguida, você pode fornecer continuamente atualizações de aplicativos para o AWS Serverless Application Repository, sem escrever nenhum código.

> **Important**
>
> Muitas das ações que você adiciona ao pipeline nesse procedimento envolvem AWS recursos que você precisa criar antes de criar o pipeline. AWS os recursos para suas ações de origem sempre devem ser criados na mesma AWS região em que você cria seu pipeline. Por exemplo, se você criar seu pipeline na região Leste dos EUA (Ohio), seu CodeCommit repositório deverá estar na região Leste dos EUA (Ohio).
>
> Você pode adicionar ações entre regiões ao criar seu pipeline. AWS os recursos para ações entre regiões devem estar na mesma AWS região em que você planeja executar a ação. Para obter mais informações, consulte Adicione uma ação entre regiões em CodePipeline.

Antes de começar

Neste tutorial, assumimos o seguinte.

- Você está familiarizado com o AWS Serverless Application Model (AWS SAM) e o AWS Serverless Application Repository.
- Você tem um aplicativo sem servidor hospedado no GitHub qual você publicou no AWS Serverless Application Repository usando a CLI AWS SAM . Para publicar um aplicativo de exemplo no AWS Serverless Application Repository, consulte Início rápido: publicação de aplicativos no Guia do AWS Serverless Application Repository desenvolvedor. Para publicar seu próprio aplicativo no AWS Serverless Application Repository, consulte Publicação de aplicativos usando a AWS SAM CLI no Guia do AWS Serverless Application Model desenvolvedor.

Etapa 1: Criar um arquivo buildspec.yml

Crie um `buildspec.yml` arquivo com o conteúdo a seguir e adicione-o ao repositório do GitHub seu aplicativo sem servidor. *`template.yml`*Substitua pelo AWS SAM modelo do seu aplicativo e *`bucketname`* pelo bucket do S3 em que o aplicativo empacotado está armazenado.

```
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
```

Etapa 2: Criar e configurar o pipeline

Siga estas etapas para criar seu pipeline no Região da AWS local em que você deseja publicar seu aplicativo sem servidor.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

2. Se necessário, mude para o Região da AWS local em que você deseja publicar seu aplicativo sem servidor.
3. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
4. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
5. Selecione Criar pipeline. Em Etapa 2: escolher a página de configurações do pipeline, em Nome do pipeline, insira o nome do seu pipeline.
6. Em Tipo de pipeline, escolha V2. Para obter mais informações, consulte Tipos de pipeline. Escolha Próximo.
7. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.
8. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
9. Na página Etapa 3: Adicionar estágio de origem, em Provedor de origem, escolha GitHub.
10. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulteGitHub conexões.
11. Em Repositório, escolha seu repositório GitHub de origem.
12. Em Filial, escolha sua GitHub filial.
13. Deixe os padrões restantes para a ação de origem. Escolha Próximo.
14. Na página Etapa 4: Adicionar etapa de compilação, adicione um estágio de compilação:
    a. Em Build provider (Provedor de compilação), escolha AWS CodeBuild. Em Region (Região), use a região do pipeline.
    b. Escolha Criar projeto.
    c. Em Nome do projeto, insira um nome para esse projeto de compilação.
    d. Em Environment image (Imagem do ambiente), escolha Managed image (Imagem gerenciada). Para Operating system, selecione Ubuntu.
    e. Em Runtime (Tempo de execução) e Runtime version (Versão do tempo de execução), escolha o tempo de execução e a versão necessários para o seu aplicativo sem servidor.
    f. Em Service role (Função de serviço), selecione New service role (Nova função de serviço).
    g. Para Build specifications (Especificações da compilação), escolha Use a buildspec file (Usar um arquivo buildspec).

h. Escolha Continuar para CodePipeline. Isso abre o CodePipeline console e cria um CodeBuild projeto que usa o `buildspec.yml` em seu repositório para configuração. O projeto de criação usa um perfil de serviço para gerenciar permissões de AWS service (Serviço da AWS) . Essa etapa pode levar alguns minutos.

i. Escolha Próximo.

15. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

16. Na página Etapa 6: Adicionar estágio de implantação, escolha Ignorar estágio de implantação e, em seguida, aceite a mensagem de aviso escolhendo Ignorar novamente. Escolha Próximo.

17. Na Etapa 7: Revisão, escolha Criar funil. Você deve ver um diagrama que mostra os estágios.

18. Conceda à função CodeBuild de serviço permissão para acessar o bucket do S3 em que seu aplicativo empacotado está armazenado.

    a. No estágio Build (Compilação) de seu novo pipeline, escolha CodeBuild.

    b. Selecione a guia Build details (Detalhes de compilação).

    c. Em Ambiente, escolha a função CodeBuild de serviço para abrir o console do IAM.

    d. Expanda a seleção para `CodeBuildBasePolicy` e escolha Edit policy (Editar política).

    e. Escolha JSON.

    f. Adicione uma nova declaração de política com o seguinte conteúdo. A instrução permite CodeBuild colocar objetos no bucket do S3 onde seu aplicativo empacotado está armazenado. *`bucketname`*Substitua pelo nome do seu bucket do S3.

```
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::bucketname/*"
            ],
            "Action": [
                "s3:PutObject"
            ]
        }
```

    g. Selecione Revisar política.

    h. Escolha Salvar alterações.

Etapa 3: Implantar o aplicativo de publicação

Siga estas etapas para implantar o aplicativo que contém a função do Lambda que realiza a publicação no AWS Serverless Application Repository. Este aplicativo é aws-serverless-codepipeline-serverlessrepo-publish.

> **Note**
>
> Você deve implantar o aplicativo da Região da AWS mesma forma que seu pipeline.

1. Vá até a página do aplicativo e escolha Deploy (Implantar).
2. Selecione I acknowledge that this app creates custom IAM roles (Eu reconheço que este aplicativo cria funções personalizadas do IAM).
3. Escolha Implantar.
4. Escolha View CloudFormation Stack para abrir o CloudFormation console.
5. Expanda a seção Resources (Recursos). Veja ServerlessRepoPublish, o que é do tipo AWS::Lambda::Function. Anote o ID físico desse recurso para a próxima etapa. Esse ID físico será usado ao criar a nova ação de publicação no CodePipeline.

Etapa 4: Criar a ação de publicação

Siga essas etapas para criar a ação de publicação em seu pipeline.

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na seção de navegação à esquerda, escolha o pipeline que deseja editar.
3. Escolha Editar.
4. Após o último estágio do pipeline atual, escolha + Add stage (+ Adicionar estágio). Em Stage name (Nome do estágio) insira um nome, como **`Publish`**, e escolha Add stage (Adicionar estágio).
5. No novo estágio, escolha + Add action group (+ Adicionar grupo de ação).
6. Insira um nome de ação. Em Action provider (Provedor de ação), em Invoke (Invocação), escolha AWS Lambda.
7. Em Artefatos de entrada, escolha BuildArtifact.

8. Em Nome da função, escolha o ID físico da função do Lambda anotado na etapa anterior.
9. Escolha Save (Salvar) para a ação.
10. Escolha Done (Concluído) para o estágio.
11. No canto superior direito, escolha Save (Salvar).
12. Para verificar seu pipeline, faça uma alteração em seu aplicativo em GitHub. Por exemplo, altere a descrição do aplicativo na `Metadata` seção do seu arquivo de AWS SAM modelo. Confirme a alteração e envie-a para sua GitHub filial. Deste modo, a execução de seu pipeline é acionada. Quando o pipeline estiver concluído, verifique se o aplicativo foi atualizado com a alteração no AWS Serverless Application Repository.

Tutorial: implantações da função Lambda com CodePipeline

Este tutorial ajuda você a criar uma ação de implantação CodePipeline que implanta seu código na função que você configurou no Lambda. Você criará uma função do Lambda de amostra na qual vai criar um alias e uma versão, adicionar a função do Lambda compactada ao local de origem e executar a ação do Lambda no pipeline.

> **Note**
>
> Como parte da criação de um pipeline no console, um bucket de artefatos S3 será usado CodePipeline por for artefacts. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 Contas da AWS seja de propriedade de quem é seguro e confiável.

> **Note**
>
> A ação de implantação `Lambda` está disponível somente para pipelines do tipo V2.

Pré-requisitos

Para você usar este tutorial para criar seu pipeline de CD, alguns recursos precisam estar em operação. Veja aqui estão os itens que você precisa para começar:

> **Note**
>
> Todos esses recursos devem ser criados na mesma AWS região.

- Um repositório de controle de origem, como GitHub, ou um bucket S3 de origem (este tutorial usa o S3), onde você armazenará um `.zip` arquivo criado para sua função Lambda.
- Você deve usar uma função CodePipeline de serviço existente que tenha sido atualizada com as permissões para essa ação. Para atualizar o perfil de serviço, consulte Permissões de política do perfil de serviço para a ação de implantação do Lambda.

Assim que você atender a esses pré-requisitos, poderá continuar com o tutorial e criar seu pipeline de CD.

Etapa 1: Criar a função do Lambda de amostra

Nesta etapa, você cria a função do Lambda na qual implantará.

Como criar sua função do Lambda

1. Acesse o console do Lambda e siga as etapas no tutorial a seguir para criar uma função do Lambda de amostra: link.
2. Na navegação superior, escolha Criar e selecione Iniciar do zero na parte superior da página.
3. Em Nome, insira **`MyLambdaFunction`**.
4. Publique uma nova versão. Esta será a versão para a qual o alias direcionará.
   a. Selecione a função.
   b. Escolha o menu suspenso Ações.
   c. Escolha Publicar nova versão.
   d. (Opcional) Adicione à descrição em Descrição.
   e. Selecione Publish.
5. Crie um alias para a função, como `aliasV1`.
6. Verifique se o alias está direcionando para a versão que você recém criou (como 1).

> **Note**
>
> Se escolher $LATEST, você não conseguirá usar os recursos de mudança de tráfego porque o Lambda não oferece suporte a $LATEST para um alias que direcione para mais de 1 versão.

Etapa 2: Carregar o arquivo de função no repositório

Baixe a função a e salve como um arquivo zip. Carregue o arquivo compactado no bucket do S3 usando as etapas a seguir.

Para adicionar um arquivo **`.zip`** ao repositório de origem

1. Abra o bucket do S3.
2. Escolha Carregar.
3. Carregue o arquivo zip que contém o arquivo `sample_lambda_source.zip` no bucket de origem.

   Anote o caminho.

   ```
   object key
   ```

Etapa 3: Criação do pipeline

Use o CodePipeline assistente para criar seus estágios de funil e conectar seu repositório de origem.

Para criar o pipeline

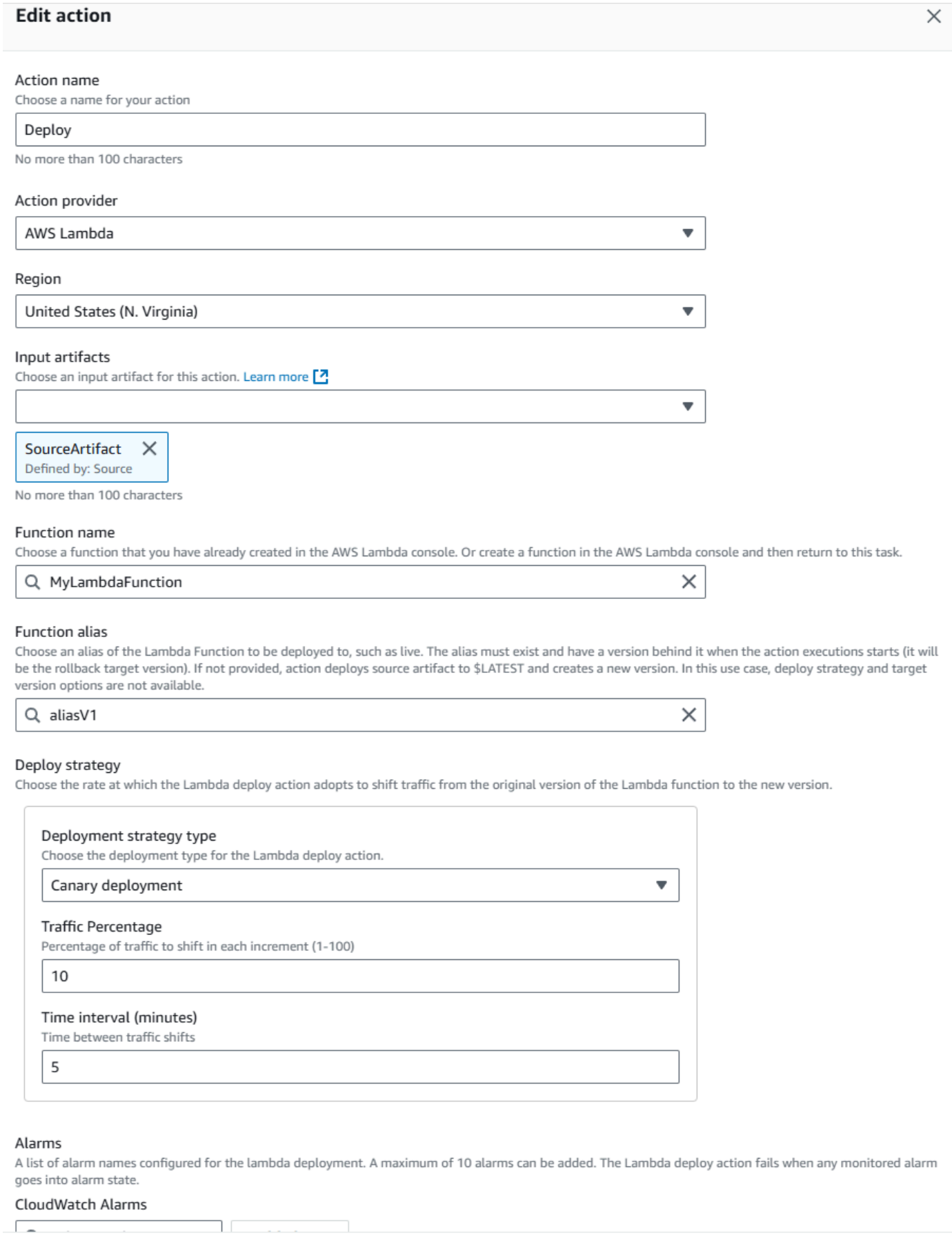
1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyPipeline`**.

5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços de CodePipeline, consulte Preços.
6. Em Função de serviço, escolha Usar função de serviço existente e, em seguida, escolha a função de CodePipeline serviço que foi atualizada com as permissões necessárias para essa ação. Para configurar sua função CodePipeline de serviço para essa ação, consulte Permissões de política do perfil de serviço para a ação de implantação do Lambda.
7. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).
8. Na página Etapa 3: Adicionar etapa de origem, adicione uma etapa de origem:

   a. Em Source provider (Provedor de origem), selecione Amazon S3.

   b. Em Chave de objeto, adicione o nome do arquivo .zip, inclusive a extensão do arquivo, como `sample_lambda_source.zip`.

   Escolha Próximo.
9. Na página Etapa 4: Adicionar etapa de compilação, escolha Ignorar.
10. Na página Etapa 5: Adicionar etapa de teste, escolha Ignorar.
11. Na página Etapa 6: Adicionar etapa de implantação, escolha Lambda.

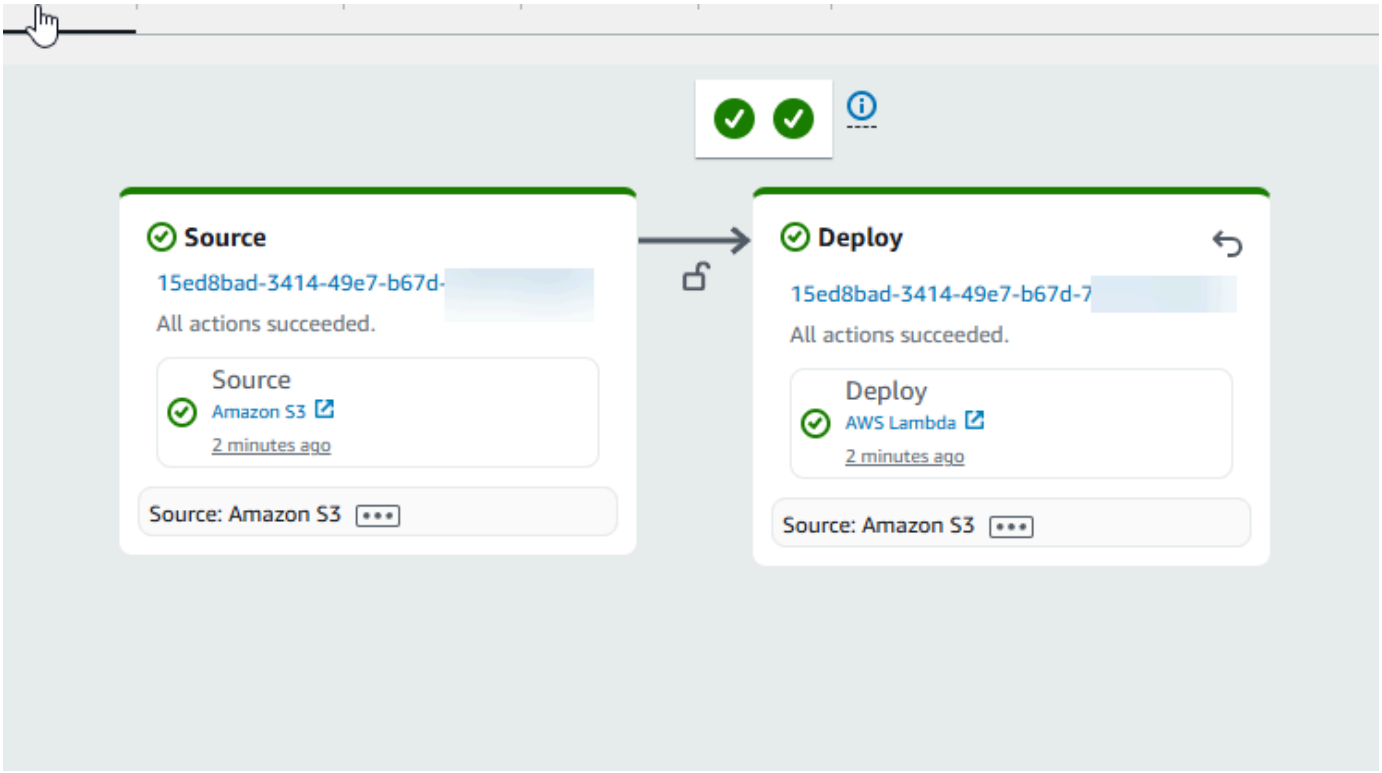

Edit action
Action name
Choose a name for your action
Deploy
No more than 100 characters
Action provider
AWS Lambda
Region
United States (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
Defined by: Source
No more than 100 characters
Function name
Choose 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. Adicione o nome e o alias da função.

b. Escolha a estratégia de implantação.

c. Escolha Próximo.

12. Na página Etapa 7: Revisão, revise a configuração do pipeline e escolha Create pipeline para criar o pipeline.



Etapa 4: Testar o pipeline

Seu pipeline deve ter tudo para executar uma implantação AWS contínua end-to-end nativa. Agora, teste a funcionalidade enviando uma alteração de código ao repositório de origem.

Para testar o pipeline

1. Faça uma alteração no código no repositório de origem configurado, confirme e envie a alteração.
2. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
3. Escolha o pipeline na lista.
4. Observe a evolução do pipeline pelos respectivos estágios. O pipeline deve ser concluído, e a ação é implantada na função do Lambda.

Saiba mais

A ação de implantação do Lambda permite dois métodos de implantação. Um método é a mudança de tráfego sozinho, sem um artefato de entrada da ação de origem. O outro método é atualizar o código da função usando um artefato de entrada da ação de origem e, em seguida, publicar uma nova versão com base no código atualizado. Para o segundo método, se o alias for fornecido, CodePipeline também fará a mudança de tráfego. Este tutorial da ação de implantação do Lambda demonstra como atualizar a função usando um artefato de origem.

Para saber mais sobre a ação, consulte a página de referência da ação em AWS Lambda Referência da ação de implantação do.

Tutorial: Usar variáveis com ações de invocação do Lambda

Uma ação de invocação do Lambda pode usar variáveis de outra ação como parte de sua entrada e retornar novas variáveis juntamente com sua saída. Para obter informações sobre variáveis para ações em CodePipeline, consulteReferência de variáveis.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

No final deste tutorial, você terá:

- Uma ação de invocação do Lambda que:
  - Consome a `CommitId` variável de uma ação de CodeCommit origem
  - Produz três novas variáveis: `dateTime`, `testRunId` e `region`
- Uma ação de aprovação manual que consome as novas variáveis da ação de invocação do Lambda para fornecer um URL de teste e um ID de execução de teste
- Um pipeline atualizado com as novas ações

Tópicos

- Pré-requisitos
- Etapa 1: criar uma função do Lambda
- Etapa 2: Adicionar uma ação de invocação do Lambda e uma ação de aprovação manual ao pipeline

Pré-requisitos

Antes de começar, você deve ter o seguinte:

- Você pode criar ou usar o pipeline com a CodeCommit fonte inseridaTutorial: criar um pipeline simples (CodeCommit repositório).
- Edite seu pipeline existente para que a ação CodeCommit de origem tenha um namespace. Atribua o namespace `SourceVariables` à ação.

Etapa 1: criar uma função do Lambda

Use as etapas a seguir para criar uma função do Lambda e uma função de execução do Lambda. Adicione a ação do Lambda ao pipeline após criar a função do Lambda.

Para criar uma função do Lambda e um perfil de execução

1. Faça login no Console de gerenciamento da AWS e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Escolha a opção Criar função. Deixe Author from scratch (Criar do zero) selecionado.
3. Em Function Name (Nome da função), insira o nome da função, como **`myInvokeFunction`**. Em Runtime (Tempo de Execução), deixe a opção padrão marcada.
4. Expanda Choose or create an execution role (Escolher ou criar uma função de execução). Escolha Create a new role with basic Lambda permissions (Criar uma nova função com permissões básicas do Lambda).
5. Escolha a opção Criar função.
6. Para usar uma variável de outra ação, ela precisará ser passada para `UserParameters` na configuração de ação de invocação do Lambda. Você configurará a ação em seu pipeline posteriormente no tutorial, mas adicionará o código presumindo que a variável será passada.

Para produzir novas variáveis, defina uma propriedade chamada `outputVariables` na entrada como `putJobSuccessResult`. Não é possível produzir variáveis como parte de um `putJobFailureResult`.

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

Na nova função, na guia Código, cole o seguinte código de exemplo em `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. Permitir que a função salve automaticamente.

8. Copie o nome do recurso da Amazon (ARN) contido no campo Função ARN na parte superior da tela.
9. Como última etapa, abra o console AWS Identity and Access Management (IAM) em https://console.aws.amazon.com/iam/. Modifique a função de execução do Lambda para adicionar a seguinte política: AWSCodePipelineCustomActionAccess. Para ver as etapas para criar uma função de execução do Lambda ou modificar a política de função, consulte Etapa 2: Criar a função do Lambda.

Etapa 2: Adicionar uma ação de invocação do Lambda e uma ação de aprovação manual ao pipeline

Nesta etapa, você adiciona uma ação de invocação do Lambda ao pipeline. Você adiciona a ação como parte de um estágio chamado Test (Teste). O tipo de ação é uma ação de chamada. Em seguida, você adiciona uma ação de aprovação manual após a ação de chamada.

Para adicionar uma ação do Lambda e uma ação de aprovação manual ao pipeline

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   Os nomes de todos os funis associados à sua AWS conta são exibidos. Escolha o pipeline onde deseja adicionar a ação.
2. Adicione uma ação de teste do Lambda ao pipeline.
   a. Para editar o pipeline, escolha Edit (Editar). Adicione um estágio após a ação de origem no pipeline existente. Insira um nome para o estágio, como **`Test`**.
   b. No novo estágio, escolha Adicionar grupo de ação para adicionar uma ação. Em Action Name (Nome da ação), insira o nome da ação de chamada, como **`Test_Commit`**.
   c. Em Provedor de ação, selecione AWS Lambda.
   d. Em Input artifacts (Artefatos de entrada), escolha o nome do artefato de saída da ação de origem, como `SourceArtifact`.
   e. Em FunctionName, adicione o ARN da função Lambda que você criou.
   f. Em Variable namespace (Namespace de variável), adicione o nome do namespace, como **`TestVariables`**.
   g. Em Artefatos de saída, adicione o nome do artefato de saída, como **`LambdaArtifact`**.
   h. Selecione Concluído.

3. Adicionar a ação de aprovação manual ao pipeline.

   a. Com o pipeline ainda no modo de edição, adicione um estágio após a ação de chamada. Insira um nome para o estágio, como **`Approval`**.

   b. No novo estágio, escolha o ícone para adicionar uma ação. Em Action name (Nome da ação), insira o nome da ação de aprovação, como **`Change_Approval`**.

   c. Em Action provider (Provedor de ação), escolha Manual approval (Aprovação manual).

   d. No URL for review (URL para revisão), construa o URL adicionando a sintaxe da variável para a variável `region` e a variável `CommitId`. Certifique-se de usar os namespaces atribuídos às ações que fornecem as variáveis de saída.

      Neste exemplo, o URL com a sintaxe variável de uma CodeCommit ação tem o `SourceVariables` namespace padrão. A variável de saída da Região do Lambda tem o namespace `TestVariables`. O URL se parece ao seguinte:

      ```
      https://#{TestVariables.region}.console.aws.amazon.com/codesuite/codecommit/
      repositories/MyDemoRepo/commit/#{SourceVariables.CommitId}
      ```

      Em Comments (Comentários), construa o texto da mensagem de aprovação adicionando a sintaxe da variável para a variável `testRunId`. Para este exemplo, o URL com a sintaxe variável para a variável de saída do Lambda `testRunId` tem o namespace `TestVariables`. Insira a mensagem a seguir.

      ```
      Make sure to review the code before approving this action. Test Run ID:
       #{TestVariables.testRunId}
      ```

4. Escolha Done (Concluído) para fechar a tela de edição da ação e Done (Concluído) para fechar a tela de edição para o estágio. Para salvar o pipeline, escolha Done (Concluído). O pipeline concluído contém agora uma estrutura com estágios de origem, teste, aprovação e implantação.

   Escolha Relase Change (Alteração de versão), para executar a alteração mais recente através da estrutura do pipeline.

5. Quando o pipeline atingir o estágio de aprovação manual, selecione Review (Revisar). As variáveis resolvidas aparecem como o URL para o ID de confirmação. O aprovador pode escolher o URL para exibir a confirmação.

6. Depois que o pipeline for executado com êxito, você também poderá visualizar os valores das variáveis na página do histórico de execução da ação.

Tutorial: use uma ação de AWS Step Functions invocação em um pipeline

Você pode usar AWS Step Functions para criar e configurar máquinas de estado. Este tutorial mostra como adicionar uma ação de invocação a um pipeline que ativa execuções de máquina de estado do pipeline.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos S3 fornecido pelo cliente será usado CodePipeline por for artefacts. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 Contas da AWS seja de propriedade de quem é seguro e confiável.

Neste tutorial, você executará as seguintes tarefas:

- Crie uma máquina de estado padrão em AWS Step Functions.
- Insira o JSON de entrada da máquina de estado diretamente. Você também pode fazer upload do arquivo de entrada de máquina de estado para um bucket do Amazon Simple Storage Service (Amazon S3).
- Atualize o pipeline adicionando a ação da máquina de estado.

Tópicos

- Pré-requisito: criar ou escolher um pipeline simples
- Etapa 1: Criar a máquina de estado de exemplo
- Etapa 2: Adicionar uma ação de invocação do Step Functions ao pipeline

Pré-requisito: criar ou escolher um pipeline simples

Neste tutorial, você adicionará uma ação de invocação a um pipeline existente. Você pode usar o pipeline criado em Tutorial: Criar um pipeline simples (bucket do S3) ou Tutorial: criar um pipeline simples (CodeCommit repositório).

Use um pipeline existente com uma ação de origem e pelo menos uma estrutura de duas etapas, mas não use artefatos de origem para este exemplo.

> **Note**
>
> Talvez seja necessário atualizar a função de serviço usada pelo pipeline com permissões adicionais necessárias para executar essa ação. Para fazer isso, abra o console AWS Identity and Access Management (IAM), encontre a função e adicione as permissões à política da função. Para obter mais informações, consulte Adicionar permissões à função CodePipeline de serviço.

Etapa 1: Criar a máquina de estado de exemplo

No console do Step Functions, crie uma máquina de estado usando o modelo de exemplo `HelloWorld`. Para obter instruções, consulte Criar uma máquina de estado no Guia do desenvolvedor do AWS Step Functions .

Etapa 2: Adicionar uma ação de invocação do Step Functions ao pipeline

Adicione uma ação de invocação do Step Functions ao pipeline da seguinte maneira:

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon. com/codesuite/codepipeline/home.

   Os nomes de todos os pipelines associados à sua AWS conta são exibidos.
2. Em Nome, selecione o nome do pipeline que você deseja editar. Isso abrirá um visão detalhada do pipeline, incluindo o estado de cada uma das ações em cada estágio do pipeline.
3. Na página de detalhes do pipeline, selecione Editar.
4. Na segunda etapa do pipeline simples, escolha Editar etapa. Escolha Excluir. Isso exclui a segunda etapa agora que você não precisa mais dela.
5. Na parte inferior do diagrama, escolha + Add stage (+ Adicionar estágio).
6. Em Nome da etapa, insira um nome para a etapa, como **`Invoke`**, e escolha Adicionar etapa.
7. Escolha + Add action group (Adicionar grupo de ação).
8. Em Nome da ação, insira um nome, como **`Invoke`**.
9. Em Provedor de ação, selecione AWS Step Functions. Permita que Region (Região) seja definida para a região do pipeline.

10. Em Artefatos de entrada, selecione `SourceArtifact`.
11. No ARN da máquina de estado, escolha o Nome de recurso da Amazon (ARN) para a máquina de estado criada anteriormente.
12. (Opcional) Em Prefixo de nome de execução, insira um prefixo a ser adicionado ao ID de execução da máquina de estado.
13. Em Tipo de entrada, escolha Literal.
14. Em Entrada, insira o JSON de entrada que é esperado pela máquina de estado de exemplo `HelloWorld`.

    > **Note**
    >
    > A entrada para a execução da máquina de estado é diferente do termo usado CodePipeline para descrever artefatos de entrada para ações.

    Neste exemplo, insira o seguinte JSON:

    ```
    {"IsHelloWorldExample": true}
    ```

15. Selecione Concluído.
16. Na etapa que está sendo editada, escolha Concluído. No painel do AWS CodePipeline , escolha Save (Salvar) e selecione Save (Salvar) na mensagem de aviso.
17. Para enviar as alterações e iniciar uma execução de pipeline, selecione Release change (Liberar alteração) e Release (Liberar).
18. No pipeline concluído, escolha AWS Step Functions na ação de invocação. No AWS Step Functions console, visualize seu ID de execução da máquina de estado. O ID mostra o nome da máquina de estado `HelloWorld` e o ID de execução da máquina de estado com o prefixo `my-prefix`.

    ```
    arn:aws:states:us-west-2:account-ID:execution:HelloWorld:my-
    prefix-0d9a0900-3609-4ebc-925e-83d9618fcca1
    ```

Tutorial: Crie um pipeline que use AWS AppConfig como provedor de implantação

Neste tutorial, você configura um pipeline que entrega continuamente arquivos de configuração usando AWS AppConfig como provedor de ação de implantação em seu estágio de implantação.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos S3 fornecido pelo cliente será usado CodePipeline por for artefacts. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 Contas da AWS seja de propriedade de quem é seguro e confiável.

Tópicos

- Pré-requisitos
- Etapa 1: Crie seus AWS AppConfig recursos
- Etapa 2|: Fazer upload de arquivos para o bucket de origem do S3
- Etapa 3: Criar o pipeline
- Etapa 4: Realizar uma alteração em qualquer arquivo de origem e verificar a implantação

Pré-requisitos

Antes de começar, é necessário concluir as seguintes etapas:

- Este exemplo usa uma origem do S3 para o pipeline. Crie ou use um bucket do Amazon S3 com o versionamento habilitado. Você pode seguir as instruções em Etapa 1: criar um bucket de origem do S3 para a aplicação para criar um bucket do S3.

Etapa 1: Crie seus AWS AppConfig recursos

Nesta seção, você criará os seguintes recursos:

- Um aplicativo em AWS AppConfig é uma unidade lógica de código que fornece recursos para seus clientes.

- Um ambiente em AWS AppConfig é um grupo lógico de AppConfig destinos de implantação, como aplicativos em um ambiente beta ou de produção.
- Um perfil de configuração é um conjunto de configurações que influenciam o comportamento do aplicativo. O perfil de configuração AWS AppConfig permite acessar sua configuração em seu local armazenado.
- (Opcional) Uma estratégia de implantação AWS AppConfig define o comportamento de uma implantação de configuração, como a porcentagem de clientes que devem receber a nova configuração implantada a qualquer momento durante a implantação.

Como criar uma aplicação, um ambiente, um perfil de configuração e uma estratégia de implantação

1. Faça login no Console de gerenciamento da AWS.
2. Use as etapas nos tópicos a seguir para criar seus recursos em AWS AppConfig.

   - Criar um aplicativo.
   - Crie um ambiente.
   - Crie um perfil AWS CodePipeline de configuração.
   - (Opcional) Escolha uma estratégia de implantação predefinida ou crie sua própria.

Etapa 2|: Fazer upload de arquivos para o bucket de origem do S3

Nesta seção, crie um ou mais arquivos de configuração. Em seguida, compacte e envie seus arquivos de origem para o bucket que o pipeline usa na etapa de origem.

Para criar arquivos de configuração

1. Crie um arquivo `configuration.json` para cada configuração em cada região. O conteúdo inclui o seguinte:

   ```
   Hello World!
   ```

2. Use as etapas a seguir para compactar e fazer upload dos arquivos de configuração.

Para compactar e fazer upload de arquivos de origem

1. Crie um arquivo .zip com os arquivos e nomeie-o como `configuration-files.zip`. Por exemplo, o arquivo .zip pode usar a seguinte estrutura:

```
.
### appconfig-configurations
    ### MyConfigurations
        ### us-east-1
        #   ### configuration.json
        ### us-west-2
            ### configuration.json
```

2. No console do Amazon S3 para o bucket, selecione Carregar e siga as instruções para fazer upload do arquivo .zip.

Etapa 3: Criar o pipeline

Nesta seção, você criará um pipeline com as seguintes ações:

- Uma etapa de origem com uma ação do Amazon S3 em que os artefatos de origem são os arquivos da sua configuração.
- Um estágio de implantação com uma ação AppConfig de implantação.

Criar um pipeline com o assistente

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyAppConfigPipeline`**.
5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços de CodePipeline, consulte Preços.
6. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.
7. Deixe as configurações em Advanced settings (Configurações avançadas) como padrão e escolha Next (Próximo).

8. Em Etapa 3: Adicionar etapa de origem, em Provedor de origem, escolha Amazon S3. Em Bucket, escolha o nome do bucket de origem do S3.

   Em chave de objeto do S3, insira o nome do arquivo .zip: `configuration-files.zip`.

   Escolha Próximo.

9. Em Etapa 4: Adicionar etapa de compilação, escolha Ignorar etapa de compilação e aceite a mensagem de aviso escolhendo Ignorar novamente.

   Escolha Próximo.

10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

11. Na Etapa 6: Adicionar estágio de implantação:

    a. Em Deploy provider, escolha AWS AppConfig.

    b. Em Aplicativo, escolha o nome do aplicativo em que você criou AWS AppConfig. O campo mostra o ID do seu aplicativo.

    c. Em Ambiente, escolha o nome do ambiente em que você criou AWS AppConfig. O campo mostra o ID do ambiente.

    d. Em Perfil de configuração, escolha o nome do perfil de configuração que você criou AWS AppConfig. O campo mostra o ID do perfil de configuração.

    e. Em Estratégia de implantação, selecione o nome da estratégia de implantação. Isso pode ser uma estratégia de implantação que você criou AppConfig ou uma que você escolheu entre as estratégias de implantação predefinidas. AppConfig O campo mostra o ID da estratégia de implantação.

    f. Em Caminho de configuração do artefato de entrada, insira o caminho do arquivo. O caminho de configuração do artefato de entrada deve corresponder à estrutura de diretórios no arquivo .zip do bucket do S3. Neste exemplo, insira o seguinte caminho de arquivo: `appconfig-configurations/MyConfigurations/us-west-2/configuration.json`.

    g. Escolha Próximo.

12. Na Etapa 7: Revisar, revise as informações e escolha Criar funil.

Etapa 4: Realizar uma alteração em qualquer arquivo de origem e verificar a implantação

Faça uma alteração nos seus arquivos de origem e faça upload da alteração efetuada no bucket de origem. Deste modo, a execução de seu pipeline é acionada. Verifique se a configuração está disponível visualizando a versão.

Tutorial: use o clone completo com uma fonte de GitHub pipeline

Você pode escolher a opção de clonagem completa para sua ação GitHub de origem em CodePipeline. Use essa opção para executar CodeBuild comandos para metadados do Git na ação de criação do pipeline.

> **Note**
>
> A opção de clonagem completa descrita aqui se refere à especificação de se CodePipeline deve clonar os metadados do repositório, que só podem ser usados por comandos. CodeBuild Para usar um token de acesso GitHub do usuário para uso com CodeBuild projetos, siga as etapas aqui para instalar o AWS Connector for GitHub app e, em seguida, deixe o campo de instalação do aplicativo vazio. AWS CodeConnections usará o token de acesso do usuário para a conexão.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos S3 fornecido pelo cliente será usado CodePipeline por for artefacts. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 Contas da AWS seja de propriedade de quem é seguro e confiável.

Neste tutorial, você criará um pipeline que se conecta ao seu GitHub repositório, usa a opção de clonagem completa para os dados de origem e executa uma CodeBuild compilação que clona seu repositório e executa comandos Git para o repositório.

> **Note**
>
> Esse recurso não está disponível nas regiões Ásia-Pacífico (Hong Kong), África (Cidade do Cabo), Oriente Médio (Bahrein), Europa (Zurique) ou (). AWS GovCloud US-West Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline. Para considerações sobre essa ação na região Europa (Milão), consulte a nota em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

Tópicos

- Pré-requisitos
- Etapa 1: Criar um arquivo README
- Etapa 2: Criar seu pipeline e criar o projeto
- Etapa 3: atualizar a política CodeBuild de função de serviço para usar conexões
- Etapa 4: Visualizar os comandos do repositório na saída da compilação

Pré-requisitos

Antes de começar, é necessário fazer o seguinte:

- Crie um GitHub repositório com sua GitHub conta.
- Tenha suas GitHub credenciais prontas. Quando você usa o Console de gerenciamento da AWS para configurar uma conexão, você é solicitado a entrar com suas GitHub credenciais.

Etapa 1: Criar um arquivo README

Depois de criar seu GitHub repositório, use essas etapas para adicionar um arquivo README.

1. Faça login no seu GitHub repositório e escolha seu repositório.
2. Para criar um novo arquivo, escolha Adicionar arquivo > Criar novo arquivo. Dê um nome ao arquivo `README.md` e adicione o seguinte texto.

```
This is a GitHub repository!
```

3. Escolha Commit changes (Confirmar alterações).

   Certifique-se de que o arquivo `README.md` esteja no nível raiz do repositório.

Etapa 2: Criar seu pipeline e criar o projeto

Nesta seção, você criará um pipeline com as seguintes ações:

- Um estágio de origem com uma conexão com seu GitHub repositório e ação.
- Um estágio de construção com uma ação de AWS CodeBuild construção.

Criar um pipeline com o assistente

1. Faça login no CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyGitHubPipeline`**.
5. Em Tipo de pipeline, selecione V1 para os fins deste tutorial. Também é possível selecionar V2; no entanto, observe que os tipos de pipeline diferem em características e preços. Para obter mais informações, consulte Tipos de pipeline.
6. Em Service role (Função do serviço), selecione New service role (Nova função de serviço).

   > **Note**
   >
   > Se você optar por usar sua função de CodePipeline serviço existente, certifique-se de ter adicionado a permissão do `codestar-connections:UseConnection` IAM à sua política de função de serviço. Para obter instruções sobre a função de CodePipeline serviço, consulte Adicionar permissões à função CodePipeline de serviço.

7. Em Configurações avançadas mantenha os padrões. Em Artifact store (Armazenamento de artefatos), selecione Default location (Local padrão) para usar o armazenamento de artefatos padrão, como o bucket de artefatos do Amazon S3 designado como padrão, para o pipeline na região que você selecionou.

> **Note**
>
> Este não é o bucket de origem para seu código-fonte. Este é o armazenamento de artefatos para o pipeline. Um armazenamento de artefatos separado, como um bucket do S3, é necessário para cada pipeline.

Escolha Próximo.

8. Na página Etapa 3: Adicionar etapa de origem, adicione uma etapa de origem:

   a. Em Provedor de origem, escolha GitHub (via GitHub aplicativo).

   b. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulteGitHub conexões.

      Você instala uma aplicação para todas as suas conexões com um provedor específico. Se você já instalou o GitHub aplicativo AWS Connector for, escolha-o e pule esta etapa.

      > **Note**
      >
      > Se você quiser criar um token de acesso de usuário, verifique se você já instalou o AWS Connector for GitHub app e deixe o campo de instalação do aplicativo vazio. AWS CodeConnections usará o token de acesso do usuário para a conexão. Para obter mais informações, consulte Acessar seu provedor de origem em CodeBuild.

   c. Em Repository name (Nome do repositório), selecione o nome do seu repositório do GitHub.

   d. Em Nome da ramificação, selecione a ramificação do repositório que você deseja usar.

   e. Assegure-se de que a opção Iniciar o pipeline na alteração do código-fonte esteja selecionada.

   f. Em Formato do artefato de saída, selecione Clone completo para habilitar a opção de clone do Git para o repositório de origem. Somente as ações fornecidas pelo CodeBuild podem usar a opção de clonagem do Git. Você usará Etapa 3: atualizar a política CodeBuild de função de serviço para usar conexões neste tutorial para atualizar as permissões da sua função de serviço CodeBuild do projeto para usar essa opção.

Escolha Próximo.

9. Em Etapa 4: Adicionar etapa de compilação, adicione um estágio de compilação:

   a. Em Build provider (Provedor de compilação), escolha AWS CodeBuild. Permita que Region (Região) seja definida para a região do pipeline.

   b. Escolha Criar projeto.

   c. Em Nome do projeto, insira um nome para esse projeto de compilação.

   d. Em Environment image (Imagem do ambiente), escolha Managed image (Imagem gerenciada). Para Operating system, selecione Ubuntu.

   e. Em Runtime (Tempo de execução), selecione Standard (Padrão). Para Imagem, escolha aws/codebuild/standard:5.0.

   f. Em Service role (Função de serviço), selecione New service role (Nova função de serviço).

      > **Note**
      >
      > Anote o nome da sua função CodeBuild de serviço. Você precisará do nome do perfil para a etapa final deste tutorial.

   g. Em Buildspec, para Build specifications (Especificações da compilação), escolha Insert build commands (Inserir comandos de compilação). Selecione Alternar para o editor e cole o seguinte em Comandos de compilação.

      > **Note**
      >
      > Na seção `env` da especificação de compilação, o assistente de credenciais para comandos do git deve estar habilitado conforme mostrado neste exemplo.

```
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. Escolha Continuar para CodePipeline. Isso retorna ao CodePipeline console e cria um CodeBuild projeto que usa seus comandos de compilação para configuração. O projeto de compilação usa uma função de serviço para gerenciar AWS service (Serviço da AWS) permissões. Essa etapa pode levar alguns minutos.

i. Escolha Próximo.

10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

11. Na página Etapa 6: Adicionar estágio de implantação, escolha Ignorar estágio de implantação e, em seguida, aceite a mensagem de aviso escolhendo Ignorar novamente. Escolha Próximo.

12. Na Etapa 7: Revisão, escolha Criar funil.

Etapa 3: atualizar a política CodeBuild de função de serviço para usar conexões

A execução inicial do pipeline falhará porque a função de CodeBuild serviço deve ser atualizada com permissões para usar conexões. Adicione a permissão do IAM `codestar-connections:UseConnection` à política de perfil de serviço. Para obter instruções sobre como atualizar a política no console do IAM, consulte Adicione CodeBuild GitClone permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com.

Etapa 4: Visualizar os comandos do repositório na saída da compilação

1. Quando sua função de serviço for atualizada com êxito, escolha Tentar novamente no CodeBuild estágio de falha.
2. Depois que o pipeline for executado com êxito, no estágio de criação bem-sucedido, selecione Visualizar detalhes.

   Na página de detalhes, selecione a guia Logs. Veja a saída da CodeBuild compilação. Os comandos geram o valor da variável inserida.

   Os comandos geram o conteúdo do arquivo `README.md`, listam os arquivos no diretório, clonam o repositório, exibem o log e arquivam o repositório como um arquivo ZIP.

Tutorial: use o clone completo com uma fonte de CodeCommit pipeline

Você pode escolher a opção de clonagem completa para sua ação CodeCommit de origem em CodePipeline. Use essa opção para permitir o acesso CodeBuild aos metadados do Git em sua ação de criação de funil.

Neste tutorial, você cria um pipeline que acessa seu CodeCommit repositório, usa a opção de clonagem completa para dados de origem e executa uma CodeBuild compilação que clona seu repositório e executa comandos Git para o repositório.

> **Note**
>
> CodeBuild ações são as únicas ações downstream que suportam o uso de metadados do Git disponíveis com a opção de clone do Git. Além disso, embora seu funil possa conter ações entre contas, a CodeCommit ação e a CodeBuild ação devem estar na mesma conta para que a opção de clonagem completa seja bem-sucedida.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Tópicos

- Pré-requisitos
- Etapa 1: Criar um arquivo README
- Etapa 2: Criar seu pipeline e criar o projeto
- Etapa 3: atualizar a política da função CodeBuild de serviço para clonar o repositório
- Etapa 4: Visualizar os comandos do repositório na saída da compilação

Pré-requisitos

Antes de começar, você deve criar um CodeCommit repositório na mesma AWS conta e região do seu funil.

Etapa 1: Criar um arquivo README

Use estas etapas para adicionar um arquivo README ao repositório de origem. O arquivo README fornece um exemplo de arquivo de origem para leitura da ação CodeBuild downstream.

Para adicionar um arquivo README

1. Faça login no seu repositório e escolha seu repositório.

2. Para criar um arquivo, selecione Adicionar arquivo > Criar arquivo. Dê um nome ao arquivo `README.md` e adicione o seguinte texto.

```
This is a CodeCommit repository!
```

3. Escolha Commit changes (Confirmar alterações).

   Certifique-se de que o arquivo `README.md` esteja no nível raiz do repositório.

Etapa 2: Criar seu pipeline e criar o projeto

Nesta seção, você criará um pipeline com as seguintes ações:

- Um estágio de origem com uma ação CodeCommit de origem.
- Um estágio de construção com uma ação de AWS CodeBuild construção.

Criar um pipeline com o assistente

1. Faça login no CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyCodeCommitPipeline`**.
5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.
6. Em Perfil de serviço, faça um dos seguintes procedimentos:
   - Escolha Existing service role (Função de serviço existente).
   - Escolha sua função CodePipeline de serviço existente. Esse perfil deve ter a permissão do IAM `codecommit:GetRepository` para a política de perfil de serviço. Consulte Adicionar permissões à função CodePipeline de serviço.
7. Em Configurações avançadas mantenha os padrões. Escolha Próximo.
8. Na página Etapa 3: Adicionar etapa de origem, faça o seguinte:

a. Em Source provider (Provedor de código-fonte), selecione CodeCommit.

b. Em Nome do repositório, selecione o nome do repositório.

c. Em Nome da ramificação, selecione o nome da ramificação.

d. Assegure-se de que a opção Iniciar o pipeline na alteração do código-fonte esteja selecionada.

e. Em Formato do artefato de saída, selecione Clone completo para habilitar a opção de clone do Git para o repositório de origem. Somente as ações fornecidas pelo CodeBuild podem usar a opção de clonagem do Git.

Escolha Próximo.

9. Em Etapa 4: Adicionar etapa de compilação, faça o seguinte:

a. Em Build provider (Provedor de compilação), escolha AWS CodeBuild. Permita que Region (Região) seja definida para a região do pipeline.

b. Escolha Criar projeto.

c. Em Nome do projeto, insira um nome para esse projeto de compilação.

d. Em Environment image (Imagem do ambiente), escolha Managed image (Imagem gerenciada). Para Operating system, selecione Ubuntu.

e. Em Runtime (Tempo de execução), selecione Standard (Padrão). Em Imagem, escolha aws/codebuild /standard:5.0.

f. Em Service role (Função de serviço), selecione New service role (Nova função de serviço).

> **Note**
>
> Anote o nome da sua função CodeBuild de serviço. Você precisará do nome do perfil para a etapa final deste tutorial.

g. Em Buildspec, para Build specifications (Especificações da compilação), escolha Insert build commands (Inserir comandos de compilação). Selecione Alternar para editor e cole o seguinte em Comandos de compilação:

```
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. Escolha Continuar para CodePipeline. Isso leva você de volta ao CodePipeline console e cria um CodeBuild projeto que usa seus comandos de construção para configuração. O projeto de criação usa um perfil de serviço para gerenciar permissões de AWS service (Serviço da AWS) . Essa etapa pode levar alguns minutos.

i. Escolha Próximo.

10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

11. Na página Etapa 6: Adicionar estágio de implantação, escolha Ignorar estágio de implantação e aceite a mensagem de aviso escolhendo Ignorar novamente. Escolha Próximo.

12. Na Etapa 7: Revisão, escolha Criar funil.

Etapa 3: atualizar a política da função CodeBuild de serviço para clonar o repositório

A execução inicial do pipeline falhará porque você precisa atualizar a função CodeBuild de serviço com permissões para extrair do seu repositório.

Adicione a permissão do IAM `codecommit:GitPull` à política de perfil de serviço. Para obter instruções sobre como atualizar a política no console do IAM, consulte Adicionar CodeBuild GitClone permissões para ações CodeCommit de origem.

Etapa 4: Visualizar os comandos do repositório na saída da compilação

Para visualizar a saída da compilação

1. Quando sua função de serviço for atualizada com êxito, escolha Tentar novamente no CodeBuild estágio de falha.

2. Depois que o pipeline for executado com êxito, no estágio de criação bem-sucedido, selecione Visualizar detalhes.

   Na página de detalhes, selecione a guia Logs. Veja a saída da CodeBuild compilação. Os comandos geram o valor da variável inserida.

   Os comandos geram o conteúdo do arquivo `README.md`, listam os arquivos no diretório, clonam o repositório, visualizam o log e executam `git describe --all`.

Tutorial: criar um pipeline com AWS CloudFormation StackSets ações de implantação

Neste tutorial, você usa o AWS CodePipeline console para criar um pipeline com ações de implantação para criar um conjunto de pilhas e criar instâncias de pilha. Quando o pipeline é executado, o modelo cria um conjunto de pilhas e também cria e atualiza as instâncias em que o conjunto de pilhas é implantado.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

Há duas maneiras de gerenciar permissões para um conjunto de pilhas: funções de IAM autogerenciadas e AWS gerenciadas. Este tutorial fornece exemplos de permissões autogerenciadas.

Para usá-lo StackSets de forma mais eficaz CodePipeline, você deve ter uma compreensão clara dos conceitos por trás CloudFormation StackSets e de como eles funcionam. Veja StackSets os conceitos no Guia AWS CloudFormation do usuário.

Tópicos

- Pré-requisitos
- Etapa 1: faça o upload da amostra AWS CloudFormation modelo e arquivo de parâmetros
- Etapa 2: Criar o pipeline
- Etapa 3: Visualizar a implantação inicial
- Etapa 4: adicionar uma CloudFormationStackInstances ação
- Etapa 5: Visualizar os recursos do conjunto de pilhas para a implantação
- Etapa 6: Fazer uma atualização no conjunto de pilhas

Pré-requisitos

Para operações de conjunto de pilhas, você vai usar duas contas diferentes: uma conta de administração e uma conta de destino. Você vai criar conjuntos de pilhas na conta de administrador. Na conta de destino, vai criar pilhas individuais pertencentes a um conjunto de pilhas.

Como criar um perfil de administrador com a conta de administrador

- Siga as instruções em Definir permissões básicas para operações de conjuntos de pilhas. O perfil deve ser nomeado **`AWSCloudFormationStackSetAdministrationRole`**.

Como criar um perfil de serviço na conta de destino

- Crie um perfil de serviço na conta de destino que confie na conta do administrador. Siga as instruções em Definir permissões básicas para operações de conjuntos de pilhas. O perfil deve ser nomeado **`AWSCloudFormationStackSetExecutionRole`**.

Etapa 1: faça o upload da amostra AWS CloudFormation modelo e arquivo de parâmetros

Crie um bucket de origem para os arquivos de parâmetros e modelos de conjunto de pilhas. Faça o download do arquivo AWS CloudFormation de modelo de amostra, configure um arquivo de parâmetros e, em seguida, compacte os arquivos antes de fazer o upload para seu bucket de origem do S3.

> **Note**
>
> Compacte os arquivos de origem antes de fazer upload para o bucket de origem do S3, mesmo que o único arquivo de origem seja o modelo.

Como criar um bucket de origem do S3

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Selecione Criar bucket.
3. Em Nome do bucket, insira um nome para o bucket.

Em Região, selecione a região onde você deseja criar o pipeline. Selecione Criar bucket.

4. Depois que o bucket é criado, um banner de sucesso é exibido. Escolha Go to bucket details (Ir para detalhes do bucket).
5. Na guia Properties (Propriedades) escolha Versioning (Versionamento). Escolha Enable versioning (Ativar versionamento) e escolha Save (Salvar).

Para criar o AWS CloudFormation arquivo de modelo

1. Baixe o seguinte arquivo de modelo de exemplo para gerar CloudTrail configuração para conjuntos de pilhas:https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSCloudtrail.yml.
2. Salve o arquivo como `template.yml`.

Como criar o arquivo parameters.txt

1. Crie um arquivo com os parâmetros para a implantação. Os parâmetros são valores que você deseja atualizar na pilha em runtime. O arquivo de exemplo a seguir atualiza os parâmetros do modelo do conjunto de pilhas para permitir a validação do registro em log e eventos globais.

```
[
  {
    "ParameterKey": "EnableLogFileValidation",
    "ParameterValue": "true"
  },
  {
    "ParameterKey": "IncludeGlobalEvents",
    "ParameterValue": "true"
  }
]
```

2. Salve o arquivo como `parameters.txt`.

Como criar o arquivo accounts.txt

1. Crie um arquivo com as contas nas quais você deseja criar instâncias, conforme mostrado no arquivo de exemplo a seguir.

```
[
    "111111222222","333333444444"
]
```

2. Salve o arquivo como `accounts.txt`.

Como criar e fazer upload de arquivos de origem

1. Combine os arquivos em um único arquivo ZIP. Os arquivos devem ter a aparência a seguir no arquivo ZIP.

```
template.yml
parameters.txt
accounts.txt
```

2. Faça upload do arquivo ZIP no bucket do S3. Esse arquivo é o artefato de origem criado pelo assistente Create Pipeline para sua ação de implantação no CodePipeline.

Etapa 2: Criar o pipeline

Nesta seção, você criará um pipeline com as seguintes ações:

- Estágio de origem com uma ação de origem do S3 em que o artefato de origem é o arquivo de modelo e todos os arquivos de origem de apoio.
- Um estágio de implantação com uma ação de implantação do conjunto de CloudFormation pilhas que cria o conjunto de pilhas.
- Um estágio de implantação com uma ação de implantação de instâncias de CloudFormation pilha que cria as pilhas e instâncias nas contas de destino.

Para criar um funil com uma CloudFormationStackSet ação

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).

3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyStackSetsPipeline`**.
5. Em Tipo de pipeline, selecione V1 para os fins deste tutorial. Também é possível selecionar V2; no entanto, observe que os tipos de pipeline diferem em características e preços. Para obter mais informações, consulte Tipos de pipeline.
6. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma função de serviço no IAM.
7. No Armazenamento de artefatos, deixe os padrões.

   > **Note**
   >
   > Este não é o bucket de origem para seu código-fonte. Este é o armazenamento de artefatos para o pipeline. Um armazenamento de artefatos separado, como um bucket do S3, é necessário para cada pipeline. Ao criar ou editar um pipeline, você deve ter um bucket de artefatos na região do pipeline e um bucket de artefatos por AWS região em que você está executando uma ação.
   > Para obter mais informações, consulte Artefatos de entrada e saída e CodePipeline referência de estrutura de tubulação.

   Escolha Próximo.
8. Na página Etapa 3: Adicionar etapa de origem, em Provedor de origem, escolha Amazon S3.
9. Em Bucket, insira o bucket de origem do S3 que você criou para este tutorial, como `BucketName`. Em Chave de objeto do S3, insira o caminho e o nome do arquivo ZIP, como `MyFiles.zip`.
10. Escolha Próximo.
11. Em Etapa 4: Adicionar etapa de compilação, escolha Ignorar etapa de compilação e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.
12. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

13. Na Etapa 6: Adicionar estágio de implantação:

    a. No Provedor de implantação, selecione Conjunto de pilhas do AWS CloudFormation .

    b. Em Nome do conjunto de pilhas, insira um nome para o conjunto de pilhas. Esse é o nome do conjunto de pilhas que será criado pelo modelo.

    > **Note**
    >
    > Anote o nome do conjunto de pilhas. Você o usará ao adicionar a segunda ação de StackSets implantação ao seu funil.

    c. Em Caminho do modelo, insira o nome do artefato e o caminho do arquivo onde você fez upload do arquivo de modelo. Por exemplo, insira o comando a seguir usando o nome `SourceArtifact` do artefato de origem padrão.

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

    d. Em Destinos de implantação, insira o nome do artefato e o caminho do arquivo onde você fez upload do arquivo da conta. Por exemplo, insira o comando a seguir usando o nome `SourceArtifact` do artefato de origem padrão.

    ```
    SourceArtifact::accounts.txt
    ```

    e. Em Destino de implantação Regiões da AWS, insira uma região para implantação de sua instância de pilha inicial, como`us-east-1`.

    f. Expanda Opções de implantação. Em Parâmetros, insira o nome do artefato e o caminho do arquivo onde você fez upload do arquivo de parâmetros. Por exemplo, insira o comando a seguir usando o nome `SourceArtifact` do artefato de origem padrão.

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

    Para inserir os parâmetros como uma entrada literal em vez de um caminho de arquivo, digite o seguinte:

    ```
    ParameterKey=EnableLogFileValidation,ParameterValue=true
    ParameterKey=IncludeGlobalEvents,ParameterValue=true
    ```

    g. Em Recursos, escolha CAPABILITY_IAM e CAPABILITY_NAMED_IAM.

h. Em Modelo de permissão, selecione SELF_MANAGED.

i. Em Porcentagem de tolerância a falhas, insira 20.

j. Em Porcentagem máxima simultânea, insira 25.

k. Escolha Próximo.

l. Em Etapa 7: Revisar, escolha Criar pipeline. O pipeline é exibido.

m. Deixe que o pipeline seja executado.

Etapa 3: Visualizar a implantação inicial

Visualize os recursos e o status da implantação inicial. Depois de verificar se a implantação criou com êxito o conjunto de pilhas, é possível adicionar a segunda ação ao estágio de implantação.

Como visualizar os recursos

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Em Pipelines, selecione o pipeline e clique em View (Visualizar). O diagrama mostra os estágios de implantação e a origem do pipeline.
3. Escolha a CloudFormation ação na CloudFormationStackSet ação em seu funil. O modelo, os recursos e os eventos do seu conjunto de pilhas são mostrados no CloudFormation console.
4. No painel de navegação esquerdo, escolha StackSets. Na lista, selecione o novo conjunto de pilhas.
5. Escolha a guia Instâncias de pilha. Verifique se foi criada uma instância da pilha para cada conta fornecida na região us-east-1. Verifique se o status de cada instância da pilha é CURRENT.

Etapa 4: adicionar uma CloudFormationStackInstances ação

Crie uma próxima ação em seu funil que permitirá CloudFormation StackSets criar as instâncias de pilha restantes.

Como criar uma próxima ação no pipeline

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.

   Em Pipelines, selecione o pipeline e clique em View (Visualizar). O diagrama mostra os estágios de implantação e a origem do pipeline.
2. Escolha editar o pipeline. O pipeline é exibido no modo de edição.

3. No estágio de implantação, selecione Editar.
4. Na ação de implantação do Conjunto de pilhas do AWS CloudFormation , selecione Adicionar grupo de ações.
5. Na página Editar ação, adicione os detalhes da ação:

   a. Em Nome da ação, insira um nome para a ação.

   b. Em Provedor de ação, selecione Instâncias de pilhas do AWS CloudFormation .

   c. Em Artefatos de entrada, escolha SourceArtifact.

   d. Em Nome do conjunto de pilhas, insira o nome do conjunto de pilhas. Esse é o nome do conjunto de pilhas que você forneceu na primeira ação.

   e. Em Destinos de implantação, insira o nome do artefato e o caminho do arquivo onde você fez upload do arquivo da conta. Por exemplo, insira o comando a seguir usando o nome `SourceArtifact` do artefato de origem padrão.

   ```
   SourceArtifact::accounts.txt
   ```

   f. Em Destino de implantação Regiões da AWS, insira as regiões para implantação de suas instâncias de pilha restantes, `eu-central-1` como `us-east-2` e da seguinte forma:

   ```
   us-east2, eu-central-1
   ```

   g. Em Porcentagem de tolerância a falhas, insira `20`.

   h. Em Porcentagem máxima simultânea, insira `25`.

   i. Escolha Salvar.

   j. Libere manualmente uma alteração. O pipeline atualizado é exibido com duas ações no estágio de implantação.

Etapa 5: Visualizar os recursos do conjunto de pilhas para a implantação

É possível visualizar os recursos e o status da implantação do conjunto de pilhas.

Como visualizar os recursos

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Em Pipelines, selecione o pipeline e, depois, Visualizar. O diagrama mostra os estágios de implantação e a origem do pipeline.

3. Escolha a CloudFormation ação na **`AWS CloudFormation Stack Instances`** ação em seu funil. O modelo, os recursos e os eventos do seu conjunto de pilhas são mostrados no CloudFormation console.
4. No painel de navegação esquerdo, escolha StackSets. Na lista, selecione o conjunto de pilhas.
5. Escolha a guia Instâncias de pilha. Verifique se todas as instâncias de pilha restantes de cada conta que você forneceu foram criadas ou atualizadas nas regiões esperadas. Verifique se o status de cada instância da pilha é `CURRENT`.

Etapa 6: Fazer uma atualização no conjunto de pilhas

Faça uma atualização no conjunto de pilhas e implante a atualização nas instâncias. Neste exemplo, você também vai fazer uma alteração nos destinos de implantação que deseja designar para atualização. As instâncias que não fazem parte da atualização passam para um status desatualizado.

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Em Pipelines, selecione o pipeline e, em seguida, escolha Visualizar. No estágio de implantação, selecione Editar.
3. Opte por editar a ação Conjunto de pilhas do AWS CloudFormation no pipeline. Em Descrição, substitua a descrição existente do conjunto de pilhas.
4. Opte por editar a ação Instâncias de pilha do AWS CloudFormation no pipeline. Em Destino de implantação Regiões da AWS, exclua o `us-east-2` valor que foi inserido quando a ação foi criada.
5. Salve as alterações. Selecione Lançar alteração para executar o pipeline.
6. Abra a ação no CloudFormation. Escolha a guia de StackSet informações. Na StackSet descrição, verifique se a nova descrição é mostrada.
7. Escolha a guia Instâncias de pilha. Em Status, verifique se o status das instâncias de pilhas em us-east-2 é `OUTDATED`.

Tutorial: criar uma regra de verificação de variável para um pipeline como uma condição de entrada

Neste tutorial, você configura um pipeline que entrega arquivos continuamente usando GitHub como provedor de ação de origem em seu estágio de origem. O pipeline concluído detecta alterações

quando você faz uma alteração nos arquivos de origem em seu repositório de origem. O pipeline é executado e, em seguida, verifica as variáveis de saída em relação ao nome do repositório de origem e ao nome da ramificação fornecida na condição para entrada no estágio de compilação.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos do S3 fornecido pelo cliente será usado pelo CodePipeline for artefatos. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 seja de propriedade de Contas da AWS pessoas seguras e confiáveis.

> **Important**
>
> Muitas das ações que você adiciona ao seu funil neste procedimento envolvem AWS recursos que você precisa criar antes de criar o funil. AWS os recursos para suas ações de origem sempre devem ser criados na mesma AWS região em que você cria seu funil. Por exemplo, se você criar seu funil na região Leste dos EUA (Ohio), seu CodeCommit repositório deverá estar na região Leste dos EUA (Ohio).
>
> Você pode adicionar ações entre regiões ao criar seu funil. AWS os recursos para ações entre regiões devem estar na mesma AWS região em que você planeja executar a ação. Para obter mais informações, consulte Adicione uma ação entre regiões em CodePipeline.

Este exemplo usa o exemplo de pipeline com uma ação de origem GitHub (Versão 2) e uma ação de CodeBuild construção em que a condição de entrada para o estágio de construção verificará as variáveis.

Pré-requisitos

Antes de começar, é necessário fazer o seguinte:

- Crie um GitHub repositório com sua GitHub conta.
- Tenha suas GitHub credenciais prontas. Quando você usa o Console de gerenciamento da AWS para configurar uma conexão, é solicitado que você faça login com suas GitHub credenciais.

- Uma conexão com seu repositório para configurar GitHub (via GitHub aplicativo) como a ação de origem do seu funil. Para criar uma conexão com seu GitHub repositório, consulteGitHub conexões.

Etapa 1: Crie um arquivo de origem de amostra e adicione-o ao seu GitHub repositório

Nesta seção, você cria e adiciona seu arquivo de origem de exemplo ao repositório que o pipeline usa para o estágio de origem. Neste exemplo, você produz e adiciona o seguinte:

- Um arquivo `README.md`.

Depois de criar seu GitHub repositório, use estas etapas para adicionar seu arquivo README.

1. Faça login no seu GitHub repositório e escolha seu repositório.
2. Para criar um arquivo, escolha Adicionar arquivo e, em seguida, escolha Criar novo arquivo. Dê um nome ao arquivo `README.md` e adicione o seguinte texto.

   ```
   This is a GitHub repository!
   ```

3. Escolha Commit changes (Confirmar alterações). Para os propósitos deste tutorial, adicione uma mensagem de confirmação que contenha a palavra “Update” em maiúsculas, como no exemplo a seguir:

   ```
   Update to source files
   ```

   > **Note**
   >
   > A verificação de regras para strings diferencia maiúsculas de minúsculas.

   Certifique-se de que o arquivo `README.md` esteja no nível raiz do repositório.

Etapa 2: Criar o pipeline

Nesta seção, você criará um pipeline com as seguintes ações:

- Um estágio de origem com uma conexão com seu GitHub repositório e ação.
- Um estágio de CodeBuild construção em que o estágio tem uma condição On Entry configurada para a regra de verificação de variáveis.

Criar um pipeline com o assistente

1. Faça login no CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página Welcome (Bem-vindo), Getting started (Conceitos básicos) ou Pipelines, selecione Create pipeline (Criar pipeline).
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: Escolher as configurações do pipeline, em Nome do pipeline, insira **`MyVarCheckPipeline`**.
5. CodePipeline fornece tubulações do tipo V1 e V2, que diferem em características e preço. O tipo V2 é o único tipo que você pode escolher no console. Para ter mais informações, consulte Pipeline types. Para obter informações sobre preços para CodePipeline, consulte Preços.
6. Em Service role (Função do serviço), selecione New service role (Nova função de serviço).

   > **Note**
   >
   > Se, em vez disso, você optar por usar sua função de CodePipeline serviço existente, certifique-se de ter adicionado a permissão `codeconnections:UseConnection` do IAM à sua política de função de serviço. Para obter instruções sobre a função CodePipeline de serviço, consulte Adicionar permissões à função CodePipeline de serviço.

7. Em Configurações avançadas mantenha os padrões.

   Escolha Próximo.

8. Na página Etapa 3: Adicionar etapa de origem, adicione uma etapa de origem:

   a. Em Provedor de origem, escolha GitHub (via GitHub aplicativo).

   b. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulteGitHub conexões.

   c. Em Repository name (Nome do repositório), selecione o nome do seu repositório do GitHub .

d. Em Nome da ramificação, selecione a ramificação do repositório que você deseja usar.

e. Verifique se a opção Sem gatilho está selecionada.

Escolha Próximo.

9. Em Etapa 4: Adicionar etapa de compilação, adicione um estágio de compilação:

a. Em Build provider (Provedor de compilação), escolha AWS CodeBuild. Permita que Region (Região) seja definida para a região do pipeline.

b. Escolha Criar projeto.

c. Em Nome do projeto, insira um nome para esse projeto de compilação.

d. Em Environment image (Imagem do ambiente), escolha Managed image (Imagem gerenciada). Para Operating system, selecione Ubuntu.

e. Em Runtime (Tempo de execução), selecione Standard (Padrão). Em Imagem, escolha aws/codebuild /standard:5.0.

f. Em Service role (Função de serviço), selecione New service role (Nova função de serviço).

> **Note**
>
> Anote o nome da sua função CodeBuild de serviço. Você precisará do nome do perfil para a etapa final deste tutorial.

g. Em Buildspec, para Build specifications (Especificações da compilação), escolha Insert build commands (Inserir comandos de compilação). Selecione Alternar para o editor e cole o seguinte em Comandos de compilação.

```
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. Escolha Continuar para CodePipeline. Isso retorna ao CodePipeline console e cria um CodeBuild projeto que usa seus comandos de construção para configuração. O projeto de construção usa uma função de serviço para gerenciar AWS service (Serviço da AWS) permissões. Essa etapa pode levar alguns minutos.

i. Escolha Próximo.

10. Em Etapa 5: Adicionar etapa de teste, escolha Ignorar etapa de teste e aceite a mensagem de aviso escolhendo Ignorar novamente.

    Escolha Próximo.

11. Na página Etapa 6: Adicionar estágio de implantação, escolha Ignorar estágio de implantação e aceite a mensagem de aviso escolhendo Ignorar novamente. Escolha Próximo.

12. Na Etapa 7: Revisão, escolha Criar funil.

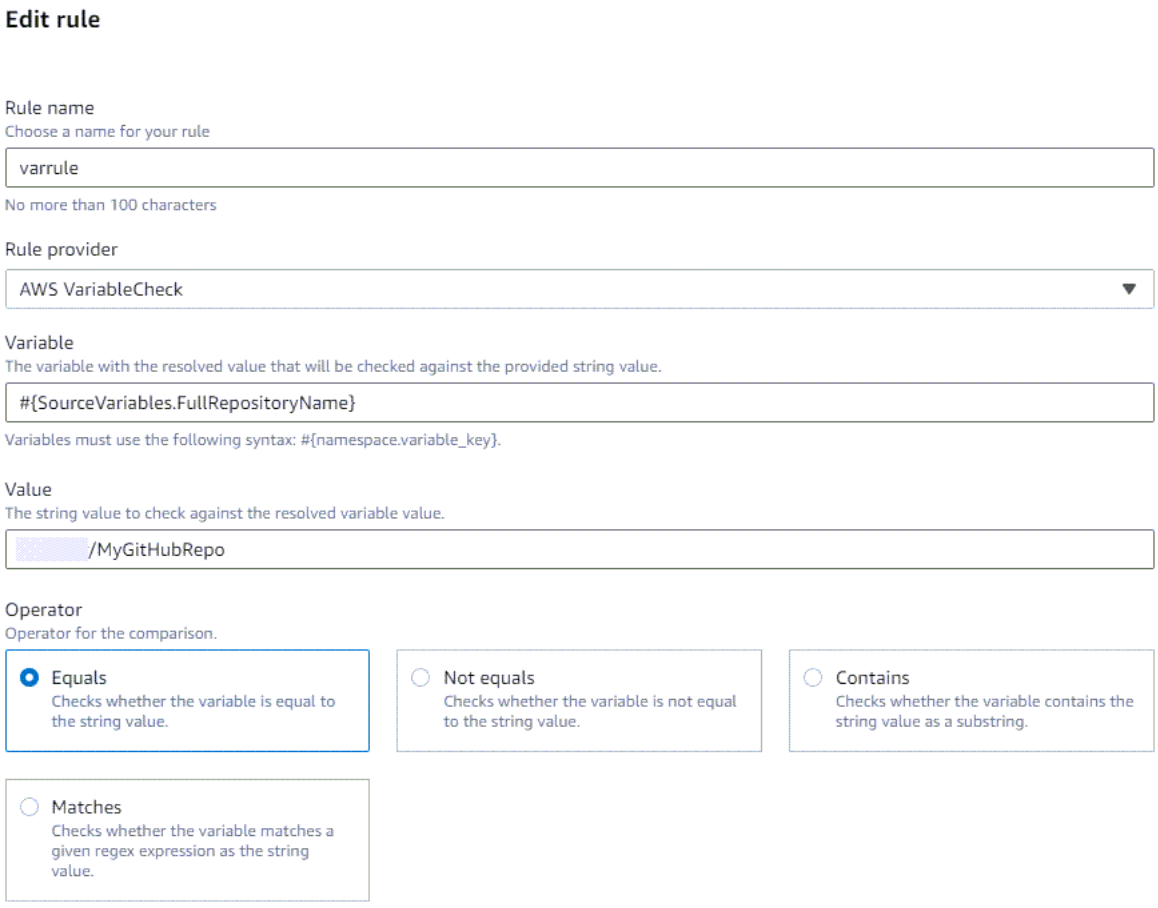
Etapa 2: editar o estágio de compilação para adicionar a condição e a regra

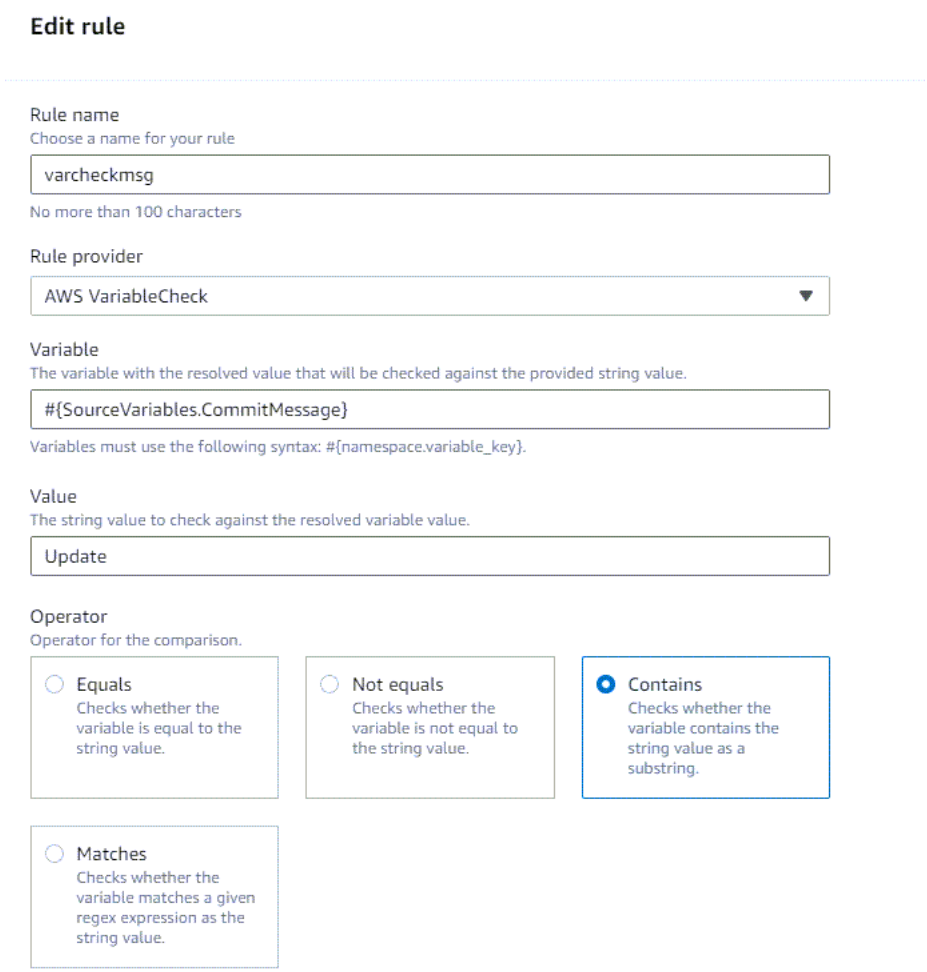
Nesta etapa, você edita o estágio para adicionar uma condição na estrada para a regra de verificação de variável.

1. Selecione o pipeline e escolha Editar. Selecione adicionar uma regra de entrada no estágio de compilação.

   Em Provedor de regras, escolha VariableCheck.

2. Em Variável, insira a variável ou variáveis que você deseja verificar. Em Valor, insira o valor da string para verificar em relação à variável resolvida. Nas telas de exemplo a seguir, uma regra é criada para uma verificação “equals” e outra é criada para uma verificação “contains”.





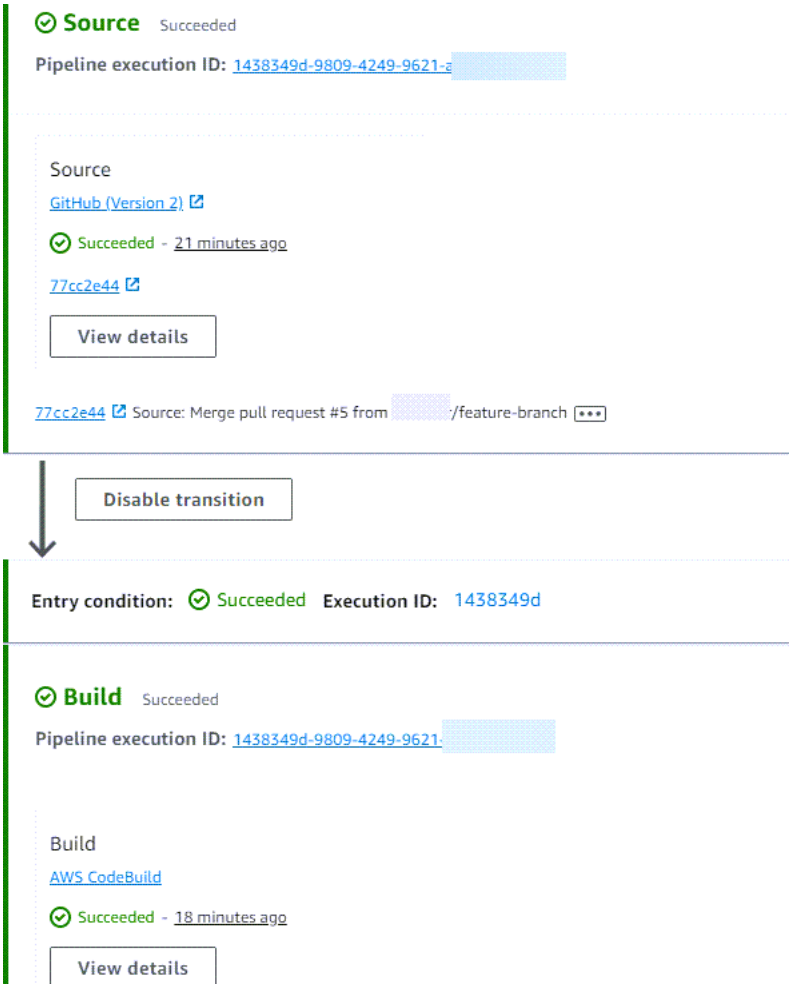
3. Escolha Salvar.

   Selecione Concluído.

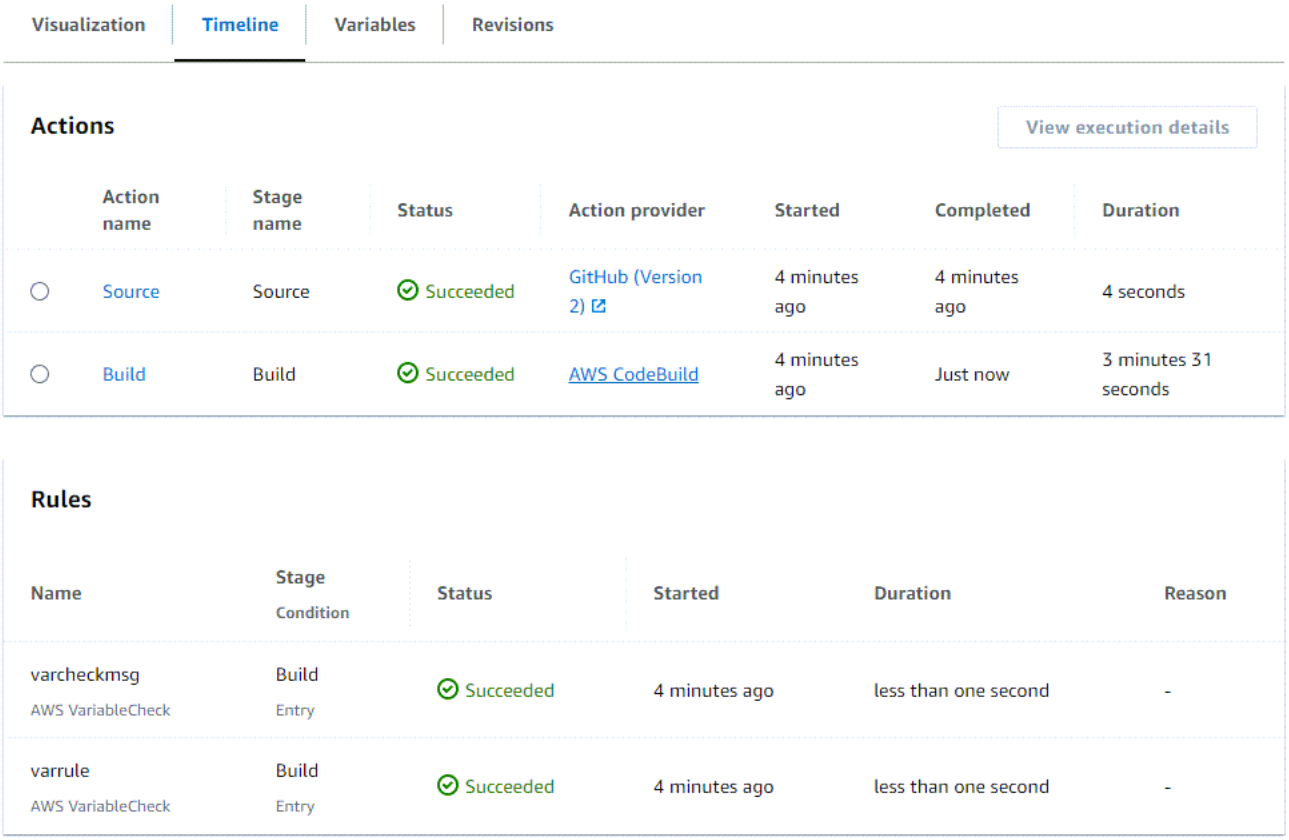
Etapa 3: executar o pipeline e visualizar as variáveis resolvidas

Nesta etapa, você visualiza os valores resolvidos e os resultados da regra de verificação de variáveis.

1. Visualize a execução resolvida após a verificação da regra ser bem-sucedida, conforme mostrado no exemplo a seguir.



2. Visualize as informações variáveis na guia Linha do tempo.


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

As seções a seguir descrevem os casos de uso do CodePipeline.

Tópicos

- Casos de uso para CodePipeline

Casos de uso para CodePipeline

Você pode criar pipelines que se integram a outros Serviços da AWS. Eles podem ser Serviços da AWS, como o Amazon S3, ou produtos de terceiros, como. GitHub Esta seção fornece exemplos para CodePipeline automatizar seus lançamentos de código usando diferentes integrações de produtos. Para obter uma lista completa de integrações CodePipeline organizadas por tipo de ação, consulteCodePipeline referência de estrutura de tubulação.

Tópicos

- Use CodePipeline com o Amazon S3, AWS CodeCommit e AWS CodeDeploy
- Use CodePipeline com provedores de ação terceirizados (GitHub e Jenkins)
- Use CodePipeline para compilar, criar e testar código com CodeBuild
- Use CodePipeline com o Amazon ECS para entrega contínua de aplicativos baseados em contêineres para a nuvem
- Use CodePipeline com o Elastic Beanstalk para entrega contínua de aplicativos web para a nuvem
- Use CodePipeline com AWS Lambda para entrega contínua Lambda-based e sem servidor de aplicativos
- Use CodePipeline com CloudFormation modelos para entrega contínua na nuvem

Use CodePipeline com o Amazon S3, AWS CodeCommit e AWS CodeDeploy

Quando você cria um pipeline, ele CodePipeline se integra a AWS produtos e serviços que atuam como provedores de ações em cada estágio do seu funil. Ao escolher estágios no assistente, você deve escolher um estágio de origem e pelo menos um estágio de compilação ou de implantação. O assistente cria os estágios para você com nomes padrão que não podem ser alterados. Estes são os nomes dos estágios criados ao configurar um pipeline completo de três estágios no assistente:

- Um estágio de origem de ação com um nome padrão de "Source".
- Um estágio de compilação de ação com um nome padrão de "Build".
- Um estágio de implantação de ação com um nome padrão de "Staging".

Você pode usar os tutoriais deste guia para criar pipelines e especificar estágios:

- As etapas em Tutorial: Criar um pipeline simples (bucket do S3) ajudam você a usar o assistente para criar um pipeline com dois estágios padrão: "Origem" e "Preparação", em que o repositório do Amazon S3 é o provedor de origem. Este tutorial cria um pipeline que usa AWS CodeDeploy para implantar um aplicativo de amostra de um bucket do Amazon S3 para instâncias do Amazon EC2 executando o Amazon Linux.
- As etapas a seguir Tutorial: criar um pipeline simples (CodeCommit repositório) ajudam você a usar o assistente para criar um pipeline com um estágio de “Origem” que usa seu AWS CodeCommit repositório como provedor de origem. Este tutorial cria um pipeline que usa AWS CodeDeploy para implantar um aplicativo de amostra de um AWS CodeCommit repositório em uma instância do Amazon EC2 executando o Amazon Linux.

Use CodePipeline com provedores de ação terceirizados (GitHub e Jenkins)

Você pode criar pipelines que se integram a produtos de terceiros, como GitHub Jenkins. As etapas em Tutorial: Criar um pipeline de quatro estágios mostram como criar um pipeline que:

- Obtém o código-fonte de um GitHub repositório
- Usa o Jenkins para compilar e testar o código-fonte,
- Usa AWS CodeDeploy para implantar o código-fonte criado e testado em instâncias do Amazon EC2 executando Amazon Linux ou Microsoft Windows Server.

Use CodePipeline para compilar, criar e testar código com CodeBuild

CodeBuild é um serviço gerenciado de compilação na nuvem que permite criar e testar seu código sem um servidor ou sistema. Use CodePipeline with CodeBuild para automatizar a execução de revisões no pipeline para entrega contínua de compilações de software sempre que houver uma alteração no código-fonte. Para obter mais informações, consulte Usar CodePipeline com CodeBuild para testar código e executar compilações.

Use CodePipeline com o Amazon ECS para entrega contínua de aplicativos baseados em contêineres para a nuvem

O Amazon ECS é um serviço de gerenciamento de contêineres que permite a você implantar aplicações baseadas em contêiner em instâncias do Amazon ECS na nuvem. Use CodePipeline com o Amazon ECS para automatizar a execução de revisões por meio do pipeline para a implantação contínua de aplicativos baseados em contêineres sempre que houver uma alteração no repositório de imagens de origem. Para obter mais informações, consulte Tutorial: implantação contínua com o CodePipeline.

Use CodePipeline com o Elastic Beanstalk para entrega contínua de aplicativos web para a nuvem

O Elastic Beanstalk é um serviço de computação que permite a você implantar aplicações e serviços web em servidores web. Use CodePipeline com o Elastic Beanstalk para implantação contínua de aplicativos web em seu ambiente de aplicativos. Você também pode usar AWS CodeStar para criar um pipeline com uma ação de implantação do Elastic Beanstalk.

Use CodePipeline com AWS Lambda para entrega contínua Lambda-based e sem servidor de aplicativos

Você pode usar AWS Lambda with CodePipeline para invocar uma AWS Lambda função, conforme descrito em Implantação de aplicativos sem servidor. Você também pode usar AWS Lambda e AWS CodeStar criar um pipeline para implantar aplicativos sem servidor.

Use CodePipeline com CloudFormation modelos para entrega contínua na nuvem

Você pode usar CloudFormation com CodePipeline para entrega e automação contínuas. Para obter mais informações, consulte Entrega contínua com CodePipeline. CloudFormation também é usado para criar os modelos para pipelines criados em AWS CodeStar.

Use CodePipeline com a Amazon Virtual Private Cloud

AWS CodePipeline agora oferece suporte a endpoints da Amazon Virtual Private Cloud (Amazon VPC) alimentados por. AWS PrivateLink Isso significa que você pode se conectar diretamente CodePipeline por meio de um endpoint privado em sua VPC, mantendo todo o tráfego dentro da VPC e da rede. AWS

A Amazon VPC é uma AWS service (Serviço da AWS) que você pode usar para lançar AWS recursos em uma rede virtual que você define. Com uma VPC, você possui controle sobre suas configurações de rede, como:

- Intervalo de endereços IP
- Sub-redes
- Tabelas de rotas
- Gateways de rede

Os endpoints VPC da Interface são alimentados por AWS PrivateLink uma AWS tecnologia que facilita a comunicação privada entre o Serviços da AWS uso de uma interface de rede elástica com endereços IP privados. Para conectar sua VPC a CodePipeline, você define uma interface para a qual VPC endpoint. CodePipeline Esse tipo de endpoint permite que você conecte a VPC aos Serviços da AWS. O endpoint fornece conectividade confiável e escalável CodePipeline sem a necessidade de um gateway de internet, instância de tradução de endereços de rede (NAT) ou conexão VPN. Para obter mais informações sobre como configurar uma VPC, consulte o Guia do usuário da VPC.

Disponibilidade

CodePipeline atualmente oferece suporte a endpoints de VPC no seguinte: Regiões da AWS

- Leste dos EUA (Ohio)
- Leste dos EUA (Norte da Virgínia)
- Oeste dos EUA (N. da Califórnia)
- Oeste dos EUA (Oregon)
- Canadá (Central)

- Europa (Frankfurt)
- Europa (Irlanda)
- Europa (Londres)
- Europa (Milão)*
- Europa (Paris)
- Europa (Estocolmo)
- Ásia-Pacífico (Hong Kong)*
- Ásia-Pacífico (Mumbai)
- Ásia-Pacífico (Tóquio)
- Ásia-Pacífico (Seul)
- Ásia-Pacífico (Singapura)
- Ásia-Pacífico (Sydney)
- América do Sul (São Paulo)
- AWS GovCloud (Oeste dos EUA)

* Você deve habilitar esta região para que possa usá-la.

Crie um VPC endpoint para CodePipeline

Você pode usar o console Amazon VPC para criar o com.amazonaws. ***`region`***Endpoint VPC .codepipeline. No console, *`region`* é o identificador de região de uma região Região da AWS suportada por CodePipeline, como `us-east-2` para a região Leste dos EUA (Ohio). Para obter mais informações, consulte Criar um endpoint de interface no Guia do usuário da Amazon VPC.

O endpoint é preenchido automaticamente com a região especificada quando você se conecta à AWS. Se você se conectar a outra região, o VPC endpoint será atualizado com a nova região.

> **Note**
>
> Outros Serviços da AWS que fornecem suporte à VPC e se integram CodePipeline, como CodeCommit, podem não oferecer suporte ao uso de endpoints da Amazon VPC para essa integração. Por exemplo, o tráfego entre CodePipeline e CodeCommit não pode ser restrito ao intervalo de sub-redes da VPC.

Solucionar problemas da configuração da VPC

Ao solucionar problemas da VPC, use as informações exibidas nas mensagens de erro de conectividade com a Internet para facilitar a identificação, diagnóstico e resolução de problemas.

1. Certifique-se de que o gateway da Internet esteja anexado à sua VPC.
2. Certifique-se de que a tabela de rotas para a sub-rede pública aponte para o gateway da Internet.
3. Certifique-se de que sua rede ACLs permita que o tráfego flua.
4. Certifique-se de que os grupos de segurança permitam o fluxo do tráfego.
5. Certifique-se de que a tabela de rotas para sub-redes privadas aponte para o gateway privado virtual.
6. Certifique-se de que a função de serviço usada por CodePipeline tenha as permissões apropriadas. Por exemplo, se você CodePipeline não tiver as permissões do Amazon EC2 necessárias para trabalhar com uma Amazon VPC, você poderá receber um erro que diz: “Erro inesperado do EC2:”. UnauthorizedOperation

Defina CI/CD pipelines com estágios e ações

Para definir um processo de lançamento automatizado em AWS CodePipeline, você cria um pipeline, que é uma construção de fluxo de trabalho que descreve como as alterações de software passam por um processo de lançamento. Um pipeline é composto de uma combinação de estágios e de ações que você configura.

> **Note**
>
> Ao adicionar estágios de criação, implantação, teste ou invocação, além das opções padrão fornecidas CodePipeline, você pode escolher ações personalizadas que você já criou para uso com seus pipelines. As ações personalizadas podem ser usadas para tarefas como executar um processo de compilação desenvolvido internamente ou um conjunto de testes. Os identificadores de versão estão incluídos para ajudar você a distinguir entre as versões diferentes de uma ação personalizada nas listas de provedores. Para obter mais informações, consulte Crie e adicione uma ação personalizada no CodePipeline.

Antes que possa criar um pipeline, você deve primeiro concluir as etapas em Começando com CodePipeline.

Para obter mais informações sobre pipelines, consulte CodePipeline conceitosCodePipeline tutoriais,, e, se você quiser usar o AWS CLI para criar um pipeline,Criar um pipeline, estágios e ações. Para ver uma lista de pipelines, consulte Veja os pipelines e os detalhes em CodePipeline.

Tópicos

- Criar um pipeline, estágios e ações
- Editar um pipeline no CodePipeline
- Veja os pipelines e os detalhes em CodePipeline
- Excluir um pipeline no CodePipeline.
- Crie um pipeline CodePipeline que use recursos de outro AWS account
- Migrar pipelines de sondagem para usar a detecção de alterações baseada em eventos
- Crie a função CodePipeline de serviço
- Marcando atributos
- Marque um funil em CodePipeline

- Criar uma regra de notificação

Criar um pipeline, estágios e ações

Você pode usar o AWS CodePipeline console ou o AWS CLI para criar um pipeline. Os pipelines devem ter pelo menos dois estágios. O primeiro estágio de um pipeline deve ser um estágio de origem. O pipeline deve ter pelo menos um outro estágio que seja um estágio de compilação ou implantação.

> **Important**
>
> Como parte da criação de um pipeline, um bucket de artefatos S3 fornecido pelo cliente será usado CodePipeline por for artefacts. (Este bucket não é o mesmo utilizado para uma ação de origem do S3.) Se o bucket de artefatos do S3 estiver em uma conta diferente da conta do seu pipeline, certifique-se de que o bucket de artefatos do S3 Contas da AWS seja de propriedade de quem é seguro e confiável.

Você pode adicionar ações ao seu funil que estejam em um lugar Região da AWS diferente do seu funil. Uma ação entre regiões é aquela em que an AWS service (Serviço da AWS) é o provedor de uma ação e o tipo de ação ou tipo de provedor está em uma AWS região diferente do seu funil. Para obter mais informações, consulte Adicione uma ação entre regiões em CodePipeline.

Você também pode criar pipelines que criem e implantem aplicações baseadas em contêiner usando o Amazon ECS como provedor de implantação. Antes de criar um pipeline que implante aplicações baseadas em contêiner com o Amazon ECS, você deverá criar um arquivo de definições de imagem, conforme descrito em Referência de arquivo de definições de imagem.

CodePipeline usa métodos de detecção de alterações para iniciar seu pipeline quando uma alteração no código-fonte é enviada. Esses métodos de detecção são baseados em tipo de origem:

- CodePipeline usa o Amazon CloudWatch Events para detectar alterações em seu repositório e filial de CodeCommit origem ou em seu bucket de origem do S3.

> **Note**
>
> Quando o console é usado para criar ou editar um pipeline, os recursos de detecção de alterações são criados para você. Se você usar a AWS CLI para criar o pipeline, deverá criar

você mesmo os recursos adicionais. Para obter mais informações, consulte CodeCommit ações de origem e EventBridge.

Tópicos

- Criar um pipeline personalizado (console)
- Criar um pipeline (CLI)
- Criar um pipeline usando modelos estáticos

Criar um pipeline personalizado (console)

Para criar um pipeline personalizado no console, você precisa fornecer ao arquivo de origem a localização e informações sobre os provedores que usará para suas ações.

Ao usar o console para criar um pipeline, você deve incluir um estágio de origem e o seguinte item (ou os dois itens):

- Um estágio de compilação.
- Um estágio de implantação.

Quando você usa o assistente de pipeline, CodePipeline cria os nomes dos estágios (origem, construção, preparação). Esses nomes não podem ser alterados. Você pode usar nomes mais específicos (por exemplo, BuildToGamma ou DeployToProd) para os estágios adicionados posteriormente.

Etapa 1: criar e nomear seu pipeline

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Na página Bem-vindo, escolha Criar pipeline.

   Se for a primeira vez que você usa CodePipeline, escolha Começar.
3. Na página Etapa 1: Escolher opção de criação, em Opções de criação, selecione a opção Criar pipeline personalizado. Escolha Próximo.
4. Em Etapa 2: escolher a página de configurações do pipeline, em Nome do pipeline, insira o nome do seu pipeline.

Em uma única AWS conta, cada funil que você cria em uma AWS região deve ter um nome exclusivo. Os nomes podem ser reutilizados para pipelines em regiões diferentes.

> **Note**
>
> Depois de criar um pipeline, não é possível alterar o nome dele. Para obter informações sobre outras limitações, consulte Cotas em AWS CodePipeline.

5. Em Tipo de pipeline, escolha uma das seguintes opções: Os tipos de pipeline diferem em características e preços. Para obter mais informações, consulte Tipos de pipeline.

   - Os pipelines do tipo V1 têm uma estrutura JSON que contém parâmetros padrão no nível do pipeline, do estágio e da ação.
   - Os pipelines do tipo V2 têm a mesma estrutura do tipo V1, além do suporte adicional a parâmetros, como gatilhos em tags Git e variáveis no nível do pipeline.

6. Em Perfil de serviço, faça um dos seguintes procedimentos:

   - Escolha Nova função de serviço para permitir CodePipeline a criação de uma nova função de serviço no IAM.
   - Escolha Existing service role (Função de serviço existente) para usar uma função de serviço já criada no IAM. Na Role ARN (Função ARN), escolha o ARN da função de serviço na lista.

   > **Note**
   >
   > Dependendo de quando sua função de serviço foi criada, talvez seja necessário atualizar suas permissões para oferecer suporte adicional Serviços da AWS. Para mais informações, consulte Adicionar permissões à função CodePipeline de serviço.

   Para mais informações sobre função de serviço e sua declaração de política, consulte Gerenciar a função CodePipeline de serviço.

7. (Opcional) Em Variáveis, escolha Adicionar variável para adicionar variáveis no nível do pipeline.

   Para obter mais informações sobre as variáveis no nível do pipeline, consulte Referência de variáveis. Para assistir a um tutorial com uma variável no nível do pipeline que é passada no momento da execução do pipeline, consulte. Tutorial: Usar variáveis no nível do pipeline

> **Note**
>
> Embora seja opcional adicionar variáveis no nível do pipeline, para um pipeline especificado com variáveis no nível do pipeline em que nenhum valor é fornecido, a execução do pipeline falhará.

8. (Opcional) Expanda Advanced settings (Configurações avançadas).
9. Em Artifact store (Armazenamento de artefatos), siga um dos seguintes procedimentos:

   a. Escolha o local padrão para usar o armazenamento de artefatos padrão, como o bucket de artefatos S3 designado como padrão, para seu pipeline no que Região da AWS você selecionou para seu pipeline.

   b. Selecione Custom location (Local personalizado) se você já tiver um armazenamento de artefato, como um bucket de artefatos do S3 na mesma região que o pipeline. Em Bucket, escolha o nome do bucket.

> **Note**
>
> Este não é o bucket de origem para seu código-fonte. Este é o armazenamento de artefatos para o pipeline. Um armazenamento de artefatos separado, como um bucket do S3, é necessário para cada pipeline. Ao criar ou editar um pipeline, você deve ter um bucket de artefatos na região do pipeline e um bucket de artefatos por AWS região em que você está executando uma ação.
> Para obter mais informações, consulte Artefatos de entrada e saída e CodePipeline referência de estrutura de tubulação.

10. (Opcional) Em Encryption key (Chave de criptografia), faça um dos seguintes procedimentos:

    a. Para usar o CodePipeline padrão AWS KMS key para criptografar os dados no armazenamento de artefatos do pipeline (bucket do S3), escolha Chave gerenciada padrão AWS .

    b. Para usar a chave gerenciada pelo cliente para criptografar os dados no armazenamento de artefatos do pipeline (bucket do S3), escolha Chave gerenciada pelo cliente. Escolha o ID da chave, o ARN da chave ou o ARN do alias.

11. Escolha Próximo.

Etapa 2: criar um estágio de origem

1. Na página Etapa 3: Adicionar etapa de origem, em Provedor de origem, escolha o tipo de repositório em que o seu código-fonte está armazenado e especifique suas opções necessárias. Os campos adicionais são exibidos dependendo do provedor de origem selecionado da seguinte forma.

   - Para Bitbucket Cloud GitHub (via GitHub aplicativo), GitHub Enterprise Server ou GitLab.com GitLab autogerenciado:

     1. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulteGitHub conexões.
     2. Escolha o repositório que deseja utilizar como local de origem do pipeline.

        Selecione a opção para adicionar um gatilho ou filtrar os tipos de gatilho para iniciar o pipeline. Para ter mais informações sobre como trabalhar com gatilhos, consulte Adicionar gatilho com tipos de eventos code push ou pull request. Para obter mais informações sobre a filtragem de padrões glob, consulte Trabalhar com padrões glob na sintaxe.
     3. Em Formato de artefato de saída, escolha o formato dos seus artefatos.
        - Para armazenar artefatos de saída da GitHub ação usando o método padrão, escolha CodePipelinedefault. A ação acessa os arquivos do GitHub repositório e armazena os artefatos em um arquivo ZIP no repositório de artefatos do pipeline.
        - Para armazenar um arquivo JSON que contém uma referência de URL ao repositório para que as ações downstream possam executar comandos Git diretamente, escolha Full clone (Clone completo). Esta opção só pode ser usada por ações downstream do CodeBuild .

          Se você escolher essa opção, precisará atualizar as permissões para sua função de serviço CodeBuild do projeto, conforme mostrado emSolução de problemas CodePipeline. Para assistir a um tutorial que mostre como usar a opção Clone completo, consulte Tutorial: use o clone completo com uma fonte de GitHub pipeline.

   - Para Amazon S3:

     1. Em Amazon S3 location (Localização do Amazon S3), forneça o nome do bucket do S3 e o caminho ao objeto em um bucket com versionamento habilitado. O formato do nome do bucket e o caminho são semelhantes a:

        ```
        s3://bucketName/folderName/objectName
        ```

> **Note**
>
> Quando o Amazon S3 é o provedor de origem do pipeline, é possível compactar o(s) arquivo(s) de origem em um único .zip e fazer upload do .zip para o bucket de origem. Também é possível fazer upload de um único arquivo descompactado; no entanto, ocorrerão falha nas ações downstream que aguardam um arquivo .zip.

2. Depois de escolher o bucket de origem do S3, CodePipeline cria a regra Amazon CloudWatch Events e a AWS CloudTrail trilha a ser criada para esse pipeline. Aceite os padrões em Change detection options (Alterar opções de detecção). Isso permite CodePipeline usar o Amazon CloudWatch Events e AWS CloudTrail detectar alterações em seu novo pipeline. Escolha Próximo.

- Para AWS CodeCommit:
  - Em Nome do repositório, escolha o nome do CodeCommit repositório que você deseja usar como local de origem para seu pipeline. Em Branch name, na lista suspensa, escolha a ramificação que você deseja usar.
  - Em Formato do artefato de saída, escolha o formato dos seus artefatos.
    - Para armazenar artefatos de saída da CodeCommit ação usando o método padrão, escolha CodePipelinedefault. A ação acessa os arquivos do CodeCommit repositório e armazena os artefatos em um arquivo ZIP no repositório de artefatos do pipeline.
    - Para armazenar um arquivo JSON que contém uma referência de URL ao repositório para que as ações downstream possam executar comandos Git diretamente, escolha Full clone (Clone completo). Esta opção só pode ser usada por ações downstream do CodeBuild .

      Se você escolher essa opção, precisará adicionar a `codecommit:GitPull` permissão à sua função de CodeBuild serviço, conforme mostrado emAdicionar CodeBuild GitClone permissões para ações CodeCommit de origem. Você também precisará adicionar `codecommit:GetRepository` as permissões à sua função CodePipeline de serviço, conforme mostrado emAdicionar permissões à função CodePipeline de serviço. Para um tutorial que mostra como usar a opção Clone completo, consulte Tutorial: use o clone completo com uma fonte de GitHub pipeline.
  - Depois de escolher o nome do CodeCommit repositório e a ramificação, uma mensagem é exibida nas opções de detecção de alterações mostrando a regra do Amazon CloudWatch Events a ser criada para esse pipeline. Aceite os padrões em Change detection options

(Alterar opções de detecção). Isso permite CodePipeline usar o Amazon CloudWatch Events para detectar alterações em seu novo pipeline.

- Para Amazon ECR:
  - Em Nome do repositório, selecione o nome do repositório do Amazon ECR.
  - Em Image tag (Tag da imagem), especifique o nome da imagem e a versão, caso seja diferente da ÚLTIMA.
  - Em Artefatos de saída, escolha o artefato de saída padrão, como MyApp, que contém o nome da imagem e as informações de URI do repositório que você deseja usar na próxima etapa.

    Para obter um tutorial sobre a criação de um pipeline para o Amazon ECS com implantações CodeDeploy azul-esverdeadas que inclua um estágio de origem do Amazon ECR, consulte. Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR

  Quando você inclui um estágio de origem do Amazon ECR no pipeline, a ação de origem gera um arquivo `imageDetail.json` como artefato de saída quando você confirma uma alteração. Para mais informações sobre o arquivo `imageDetail.json`, consulte Detail.json arquivo de imagem para ações de blue/green implantação do Amazon ECS.

> **Note**
>
> O objeto e o tipo de arquivo devem ser compatíveis com o sistema de implantação que você planeja usar (por exemplo, Elastic Beanstalk ou). CodeDeploy Os tipos de arquivos compatíveis podem incluir .zip, .tar e .tgz. Para obter mais informações sobre os tipos de contêiner compatíveis com o Elastic Beanstalk, consulte Personalizar e configurar ambientes do Elastic Beanstalk e Plataformas compatíveis. Para obter mais informações sobre como implantar revisões com CodeDeploy, consulte Carregando a revisão do seu aplicativo e preparando uma revisão.

2. Para configurar a repetição automática para um estágio, escolha Habilitar nova tentativa automática em caso de falha na etapa. Para ter mais informações sobre a repetição automática, consulte Configurar a repetição automática em caso de falha para um estágio.
3. Escolha Próximo.

Etapa 4: criar um estágio de compilação

Essa etapa é opcional se você planeja criar um estágio de implantação.

1. Na página Etapa 4: Adicionar etapa de compilação, faça o seguinte e, em seguida, escolha Próximo:

   - Escolha Ignorar etapa de compilação caso você pretenda criar um estágio de teste ou implantação.
   - Para escolher a ação Commands para o estágio de compilação, selecione Comandos.

     > **Note**
     >
     > A execução da ação Commands acarretará cobranças separadas em AWS CodeBuild. Se você planeja inserir comandos de compilação como parte de uma CodeBuild ação, escolha Outros provedores de compilação e selecione CodeBuild.

     Em Comandos, insira os comandos do shell para a ação. Para ter mais informações sobre a ação Commands, consulte Referência da ação Commands.
   - Para escolher outros provedores de compilação CodeBuild, como, escolha Outros provedores. Em Build provider (Provedor de compilação), selecione um provedor de ações personalizado dos serviços de compilação e forneça os detalhes de configuração para esse provedor. Para um exemplo sobre como incluir a Jenkins como um provedor de construção, consulte Tutorial: Criar um pipeline de quatro estágios.
   - Em Build provider (Provedor de compilação), escolha AWS CodeBuild.

     Em Região, escolha a AWS região em que o recurso existe. O campo Região designa onde os AWS recursos são criados para esse tipo de ação e tipo de provedor. Esse campo é exibido apenas para as ações em que o provedor de ação é um AWS service (Serviço da AWS). O campo Região usa como padrão a mesma AWS Região do seu funil.

     Em Project name (Nome do projeto), escolha o projeto de compilação. Se você já criou um projeto de compilação em CodeBuild, escolha-o. Ou você pode criar um projeto de compilação CodeBuild e depois retornar a essa tarefa. Siga as instruções em Criar um pipeline que usa o CodeBuild no Guia do usuário do CodeBuild.

Em Especificações de compilação, o arquivo CodeBuild buildspec é opcional e, em vez disso, você pode inserir comandos. Em Inserir comados de compilação, insira os comandos do shell para a ação. Para obter mais informações sobre considerações de uso dos comandos de compilação, consulte Referência da ação Commands. Escolha Usar um arquivo buildspec se você quiser executar comandos em outras fases ou se tiver uma lista de comandos longa.

Em Variáveis de ambiente, para adicionar variáveis de CodeBuild ambiente à sua ação de criação, escolha Adicionar variável de ambiente. Cada variável é composta por três entradas:

- Em Name (Nome), insira o nome ou chave da variável de ambiente.
- Em Value (Valor), insira o valor da variável de ambiente. Se você escolher Parâmetro para o tipo de variável, verifique se esse valor é o nome de um parâmetro que você já armazenou no AWS Systems Manager Parameter Store.

> **Note**
>
> Nós desencorajamos fortemente o uso de variáveis de ambiente para armazenar valores confidenciais, especialmente AWS credenciais. Quando você usa o CodeBuild console ou a AWS CLI, as variáveis de ambiente são exibidas em texto sem formatação. Para valores confidenciais, recomendamos que você use o tipo de Parameter (Parâmetro).

- (Opcional) Em Type (Tipo), insira o tipo de variável de ambiente. Os valores válidos são Plaintext (Texto sem formatação) ou Parameter (Parâmetro). O padrão é Plaintext (Texto sem formatação).

(Opcional) Em Tipo de compilação, escolha uma das alternativas a seguir:

- Para executar cada compilação em uma única execução de ação de compilação, escolha Compilação única.
- Para executar várias compilações na mesma execução de ação de compilação, escolha Compilação em lote.

(Opcional) Se você optar por executar compilações em lote, poderá escolher Combinar todos os artefatos do lote em um único local para colocar todos os artefatos de compilação em um único artefato de saída.

2. Para configurar a repetição automática para um estágio, escolha Habilitar nova tentativa automática em caso de falha na etapa. Para ter mais informações sobre a repetição automática, consulte Configurar a repetição automática em caso de falha para um estágio.
3. Escolha Próximo.

Etapa 5: Criar um estágio de teste

Esta etapa é opcional caso você pretenda criar um estágio de implantação.

1. Na página Etapa 5: Adicionar etapa de teste, faça o seguinte e, em seguida, escolha Próximo:
   - Escolha Ignorar etapa de teste caso você pretenda criar um estágio de compilação ou implantação.
   - Em Provedor de testes, escolha o provedor da ação de teste e preencha os campos indicados.
2. Escolha Próximo.

Etapa 6: criar um estágio de implantação

Essa etapa é opcional se você já tiver criado um estágio de compilação.

1. Na página Etapa 6: Adicionar etapa de implantação, realize o seguinte e, em seguida, escolha Próximo:
   - Escolha Ignorar etapa de implantação se você tiver criado um estágio de compilação ou teste nas etapas anteriores.

     > **Note**
     >
     > Essa opção não será exibida se você já tiver ignorado o estágio de compilação ou teste.

   - Em Deploy provider (Provedor de implantação), selecione uma ação personalizada que você criou para um provedor de implantação.

     Em Região, somente para ações entre regiões, escolha a AWS região em que o recurso foi criado. O campo Região designa o local em que os recursos da AWS serão criados para esse tipo de ação e de provedor. Esse campo exibe apenas as ações em que o provedor de ação

é um AWS service (Serviço da AWS). O campo Região usa como padrão a mesma AWS Região do seu funil.

- Em Deploy provider (Provedor de implantação), estão disponível campos para provedores padrão da seguinte forma:
  - CodeDeploy

    Em Nome do aplicativo, insira ou escolha o nome de um CodeDeploy aplicativo existente. Em Grupo de implantação, insira o nome de um grupo de implantação do aplicativo. Escolha Próximo. Você também pode criar um aplicativo, grupo de implantação ou os dois no console do CodeDeploy.
  - AWS Elastic Beanstalk

    Em Nome da aplicação, insira ou escolha o nome de uma aplicação existente do Elastic Beanstalk. Em Nome do ambiente, insira um ambiente para o aplicativo. Escolha Próximo. Você também pode criar uma aplicação, um ambiente ou os dois no console do Elastic Beanstalk.
  - AWS OpsWorks Stacks

    Em Pilha, insira ou escolha o nome da pilha que deseja usar. Em Camada, escolha a camada a que suas instâncias de destino pertencem. Em App, selecione o aplicativo que deseja atualizar e implantar. Caso precise criar um aplicativo, selecione Create a new one in (Criar outro no AWS OpsWorks).

    Para obter informações sobre como adicionar um aplicativo a uma pilha e uma camada AWS OpsWorks, consulte Adicionando aplicativos no Guia do AWS OpsWorks usuário.

    Para obter um exemplo completo de como usar um pipeline simples CodePipeline como fonte para o código que você executa em AWS OpsWorks camadas, consulte Usando com CodePipeline . AWS OpsWorks Stacks
  - AWS CloudFormation

    Execute um destes procedimentos:

    - No Modo de ação, escolha Criar ou atualizar uma pilha, insira o nome da pilha e o nome do arquivo de modelo e, em seguida, escolha o nome da função AWS CloudFormation a ser assumida. Se desejar, digite o nome de um arquivo de configuração e escolha uma opção de capacidade do IAM.

- No Modo de ação, escolha Criar ou substituir um conjunto de alterações, insira o nome da pilha e o nome do conjunto de alterações e, em seguida, escolha o nome de uma função AWS CloudFormation a ser assumida. Se desejar, digite o nome de um arquivo de configuração e escolha uma opção de capacidade do IAM.

Para obter informações sobre a integração de AWS CloudFormation recursos em um pipeline em CodePipeline, consulte Entrega contínua CodePipeline no Guia do AWS CloudFormation usuário.

- Amazon ECS

Em Nome do cluster, insira ou escolha o nome de um cluster existente do Amazon ECS.I Em Service name, insira ou escolha o nome do serviço que está executando o cluster. Você também pode criar um novo cluster e serviço. Em Image filename, insira o nome do arquivo de definições de imagem que descreve o contêiner e a imagem do serviço.

> **Note**
>
> A ação de implantação do Amazon ECS requer um arquivo `imagedefinitions.json` como entrada para a ação de implantação. O nome padrão do arquivo é imagedefinitions.json. Se você optar por usar um nome de arquivo diferente, você deve fornecê-lo ao criar o estágio de implantação do pipeline. Para obter mais informações, consulte Arquivo imagedefinitions.json para ações de implantação padrão do Amazon ECS.

Escolha Próximo.

> **Note**
>
> Certifique-se de que o cluster do Amazon ECS esteja configurado com duas ou mais instâncias. Os clusters do Amazon ECS devem conter pelo menos duas instâncias para que uma seja mantida como primária e a outra seja usada para acomodar novas implantações.

Para obter um tutorial sobre implantação de aplicativos baseados em contêiner com o pipeline, consulte Tutorial: implantação contínua com o CodePipeline.

- Amazon ECS () Blue/Green

  Insira o CodeDeploy aplicativo e o grupo de implantação, a definição da tarefa do Amazon ECS e as informações AppSpec do arquivo e, em seguida, escolha Avançar.

  > **Note**
  >
  > A ação do Amazon ECS (Blue/Green) exige um Detail.json arquivo de imagem como artefato de entrada para a ação de implantação. Como a ação de origem do Amazon ECR cria esse arquivo, os pipelines com uma ação de origem do Amazon ECR não precisam fornecer um arquivo `imageDetail.json`. Para obter mais informações, consulte Detail.json arquivo de imagem para ações de blue/green implantação do Amazon ECS.

  Para obter um tutorial sobre a criação de um pipeline para implantações azul-esverdeadas em um cluster do Amazon ECS com, consulte. CodeDeploy Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR

- AWS Service Catalog

  Escolha Enter deployment configuration (Inserir configuração de implantação) se você quiser usar campos no console para especificar sua configuração ou escolha Configuration file (Arquivo de configuração) se tiver um arquivo de configuração separado. Insira as informações do produto e da configuração e escolha Next (Próximo).

  Para obter um tutorial sobre a implantação de alterações de produtos no Service Catalog com seu pipeline, consulte Tutorial: Criar um pipeline que realiza a implantação no Service Catalog.

- Alexa Skills Kit

  No Alexa Skill ID (ID da skill da Alexa), insira o ID de skill da sua skill da Alexa. Em Client ID (ID do cliente) e Client secret (Segredo do cliente), insira as credenciais geradas usando um perfil de segurança de Login with Amazon (LWA). Em Refresh token (Token de atualização), insira o token de atualização que você gerou usando o comando da CLI do ASK para recuperar um token de atualização. Escolha Próximo.

Para obter um tutorial sobre como implantar as skills da Alexa com seu pipeline e gerar as credenciais do LWA, consulte Tutorial: Criar um pipeline que implanta uma skill do Amazon Alexa.

- Amazon S3

  Em Bucket, insira o nome do bucket do S3 que você deseja usar. Escolha Extract file before deploy (Extrair arquivo antes de implantar) se o artefato de entrada para o estágio de implantação for um arquivo ZIP. Se Extract file before deploy (Extrair arquivo antes de implantar) for selecionado, você pode, opcionalmente, inserir um valor para o Deployment path (Caminho de implantação) no qual seu arquivo ZIP será descompactado. Se não estiver selecionado, você deverá inserir um valor na chave de objeto do S3.

  > **Note**
  >
  > A maioria dos artefatos de saída do estágio de origem e de compilação é compactada. Todos os provedores de origem de pipeline, exceto o Amazon S3, compactam os arquivos de origem antes de fornecê-los como o artefato de entrada da próxima ação.

  (Opcional) Em ACL pré-configurada, insira a ACL pré-configurada a ser aplicada ao objeto implantado no Amazon S3.

  > **Note**
  >
  > A aplicação de uma ACL pré-configurada substitui todas as ACLs existentes aplicadas ao objeto.

  (Opcional) Em Cache control (Controle de cache), especifique os parâmetros de controle de cache para solicitações de download de objetos do bucket. Para obter uma lista de valores válidos, consulte o campo de cabeçalho `Cache-Control` para operações HTTP. Para inserir vários valores em Cache control (Controle de cache), use uma vírgula entre cada valor. É possível adicionar um espaço após cada vírgula (opcional), conforme mostrado neste exemplo.



A entrada de exemplo anterior é exibida na CLI da seguinte forma:

```
"CacheControl": "public, max-age=0, no-transform"
```

Escolha Próximo.

Para obter um tutorial sobre a criação de um pipeline com um provedor de ação de implantação do Amazon S3, consulte Tutorial: Criar um pipeline que usa o Amazon S3 como um provedor de implantação.

2. Para configurar a repetição automática para um estágio, escolha Habilitar nova tentativa automática em caso de falha na etapa. Para ter mais informações sobre a repetição automática, consulte Configurar a repetição automática em caso de falha para um estágio.
3. Para configurar o estágio para reversão automática, escolha Configurar reversão automática em caso de falha do estágio. Para ter mais informações sobre a reversão automática, consulte Configurar um estágio para reversão automática.
4. Escolha Próxima etapa.

Etapa 7: revisar o pipeline

- Na página Etapa 7: Revisão, revise a configuração do funil e escolha Criar funil para criar o funil ou Anterior para voltar e editar suas escolhas. Para sair do assistente sem criar um pipeline, selecione Cancelar.

Agora que criou o pipeline, você pode vê-lo no console. O pipeline começa a ser executado depois de ser criado. Para obter mais informações, consulte Veja os pipelines e os detalhes em CodePipeline. Para obter mais informações sobre como fazer alterações em seu pipeline, acesse Editar um pipeline no CodePipeline.

Criar um pipeline (CLI)

Para usar o AWS CLI para criar um pipeline, você cria um arquivo JSON para definir a estrutura do pipeline e, em seguida, executa o create-pipeline comando com o `--cli-input-json` parâmetro.

> **Important**
>
> Você não pode usar o AWS CLI para criar um pipeline que inclua ações de parceiros. Em vez disso, você deve usar o CodePipeline console.

Para obter mais informações sobre a estrutura do pipeline, consulte CodePipeline referência de estrutura de tubulação e create-pipeline na Referência da CodePipeline API.

Para criar um arquivo JSON, use o arquivo JSON do pipeline de exemplo, edite-o e, em seguida, chame esse arquivo quando executar o comando create-pipeline.

Pré-requisitos:

Você precisa do ARN da função de serviço para CodePipeline a qual você criou. Começando com CodePipeline Você usa a função CodePipeline de serviço ARN no arquivo JSON do pipeline ao executar o comando. create-pipeline Para obter mais informações sobre como criar um perfil de serviço, consulte Crie a função CodePipeline de serviço. Ao contrário do console, executar o create-pipeline comando no AWS CLI não tem a opção de criar a função CodePipeline de serviço para você. A função de serviço já deve existir.

É necessário o nome de um bucket do S3 para armazenar artefatos do pipeline. Esse bucket deve estar na mesma região que o pipeline. Use o nome do bucket no arquivo JSON do pipeline ao executar o comando create-pipeline. Ao contrário do console, a execução do create-pipeline comando no AWS CLI não cria um bucket S3 para armazenar artefatos. O bucket já deve existir.

> **Note**
>
> Você também pode usar o comando get-pipeline para obter uma cópia da estrutura JSON do pipeline e modificar a estrutura em um editor de texto plano.

Para criar o arquivo JSON

1. Em um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows), crie um novo arquivo de texto em um diretório local.
2. (Opcional) Você pode adicionar uma ou mais variáveis no nível do pipeline. Você pode referenciar esse valor na configuração de CodePipeline ações. Você pode adicionar os nomes

e valores das variáveis ao criar o pipeline. Além disso, você pode optar por atribuir valores ao iniciar o pipeline no console.

> **Note**
>
> Embora seja opcional adicionar variáveis no nível do pipeline, para um pipeline especificado com variáveis no nível do pipeline em que nenhum valor é fornecido, a execução do pipeline falhará.

Uma variável no nível do pipeline é resolvida no tempo de execução do pipeline. Todas as variáveis são imutáveis, o que significa que elas não podem ser atualizadas após a atribuição de um valor. As variáveis no nível do pipeline com valores resolvidos serão exibidas no histórico de cada execução.

Forneça variáveis no nível do pipeline usando o atributo variables na estrutura do pipeline. No exemplo a seguir, o valor da variável `Variable1` é `Value1`.

```
"variables": [
     {
         "name": "Timeout",
         "defaultValue": "1000",
         "description": "description"
     }
 ]
```

Adicione essa estrutura ao arquivo JSON do pipeline ou ao arquivo JSON de exemplo na etapa a seguir. Para obter mais informações sobre variáveis, incluindo informações de namespace, consulte Referência de variáveis.

3. Abra o arquivo em um editor de texto plano e edite os valores para refletir a estrutura que deseja criar. No mínimo, você deve alterar o nome do pipeline. Você também deve considerar se deseja alterar:

   - O bucket do S3 onde os artefatos para este pipeline são armazenados.
   - O local de origem para o código.
   - O provedor de implantação.
   - Como você deseja que o código seja implantado.

- As tags para o pipeline.

O seguinte modelo de estrutura de pipeline de dois estágios destaca os valores que você deve considerar alterar para o seu pipeline. O pipeline provavelmente contém mais de dois estágios:

```
{
    "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 exemplo adiciona tags ao pipeline por meio da inclusão da chave de tag `Project` e do valor `ProjectA` no pipeline. Para obter mais informações sobre a marcação de recursos em CodePipeline, consulteMarcando atributos.

Verifique se o parâmetro `PollForSourceChanges` está definido como mostrado a seguir no arquivo JSON:

```
                "PollForSourceChanges": "false",
```

CodePipeline usa o Amazon CloudWatch Events para detectar alterações em seu repositório e filial de CodeCommit origem ou em seu bucket de origem do S3. A próxima etapa inclui instruções para criar esses recursos manualmente para o pipeline. Definir o sinalizador como `false` desativa verificações periódicas, que não são necessárias ao se usar os métodos de detecção de alterações recomendados.

4. Para criar uma ação de compilação, teste ou implantação em uma região diferente do seu pipeline, é necessário adicionar o seguinte à sua estrutura de pipeline. Para instruções, consulte Adicione uma ação entre regiões em CodePipeline.

   - Adicione o parâmetro de `Region` à ação da estrutura do pipeline.

- Use o `artifactStores` parâmetro para especificar um repositório de artefatos para cada AWS região em que você tem uma ação.

5. Quando estiver satisfeito com a estrutura, salve o seu arquivo com um nome como **`pipeline.json`**.

Para criar um pipeline

1. Execute o comando create-pipeline e use o parâmetro `--cli-input-json` para especificar o arquivo JSON que você criou anteriormente.

   Para criar um pipeline nomeado *`MySecondPipeline`* com um arquivo JSON chamado pipeline.json que inclua o nome "*`MySecondPipeline`*" como valor para `name` no JSON, seu comando teria a seguinte aparência:

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

   > **Important**
   >
   > Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

   Este comando retorna a estrutura de todo o pipeline que você criou.

2. Para visualizar o pipeline, abra o CodePipeline console e escolha-o na lista de pipelines ou use o get-pipeline-state comando. Para obter mais informações, consulte Veja os pipelines e os detalhes em CodePipeline.

3. Se usar a ILC para criar um pipeline, você deverá criar manualmente os recursos de detecção de alterações recomendados para o seu pipeline:

   - Para um pipeline com um CodeCommit repositório, você deve criar manualmente a regra de CloudWatch Eventos, conforme descrito emCrie uma EventBridge regra para uma CodeCommit fonte (CLI).
   - Para um pipeline com uma fonte do Amazon S3, você deve criar manualmente a regra e a AWS CloudTrail trilha de CloudWatch eventos, conforme descrito em. Conectando-se às ações de origem do Amazon S3 que usam EventBridge e AWS CloudTrail

Criar um pipeline usando modelos estáticos

Você pode criar um pipeline no console que usa um modelo para configurar um pipeline com código-fonte e propriedades que você especificar. Você deve fornecer o local do arquivo de origem e informações sobre os provedores de origem que usará para suas ações. Você pode especificar uma ação de origem para o Amazon ECR ou qualquer repositório de terceiros suportado por AWS CodeConnections, como. GitHub

O modelo criará uma pilha CloudFormation para seu pipeline que inclui os seguintes recursos:

- Um pipeline é criado com o tipo de pipeline V2. Em Tipo de pipeline, escolha uma das seguintes opções: Os tipos de pipeline diferem em características e preços. Para obter mais informações, consulte Tipos de pipeline.
- Um perfil de serviço é criado para o pipeline e referenciado no modelo.
- Um repositório de artefatos é configurado utilizando o repositório de artefatos padrão, como o bucket S3 designado como padrão, para o pipeline na Região da AWS escolhida para ele.

Para ver a coleção de modelos iniciais de código aberto que são usados para o assistente de criação de modelos estáticos, consulte o repositório em. https://github.com/aws/codepipeline-starter-templates

Ao usar modelos estáticos para criar um pipeline, a estrutura do pipeline é configurada em cada modelo de acordo com as necessidades do caso de uso. Por exemplo, o modelo para uma implantação em CloudFormation é usado como exemplo neste procedimento. O modelo gera um pipeline chamado DeployToCloudFormationServicecom a seguinte estrutura:

- Um estágio de compilação que contém uma ação de origem com configuração que você especifica no assistente.
- Um estágio de implantação com uma ação de implantação e pilha de recursos associada no CloudFormation.

Quando você usa um modelo estático para criar um pipeline, CodePipeline cria os nomes dos estágios (origem, construção, preparação). Esses nomes não podem ser alterados. Você pode usar nomes mais específicos (por exemplo, BuildToGamma ou DeployToProd) para os estágios adicionados posteriormente.

Etapa 1: Acessar o console

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Na página Bem-vindo, escolha Criar pipeline.

   Se for a primeira vez que você usa CodePipeline, escolha Começar.

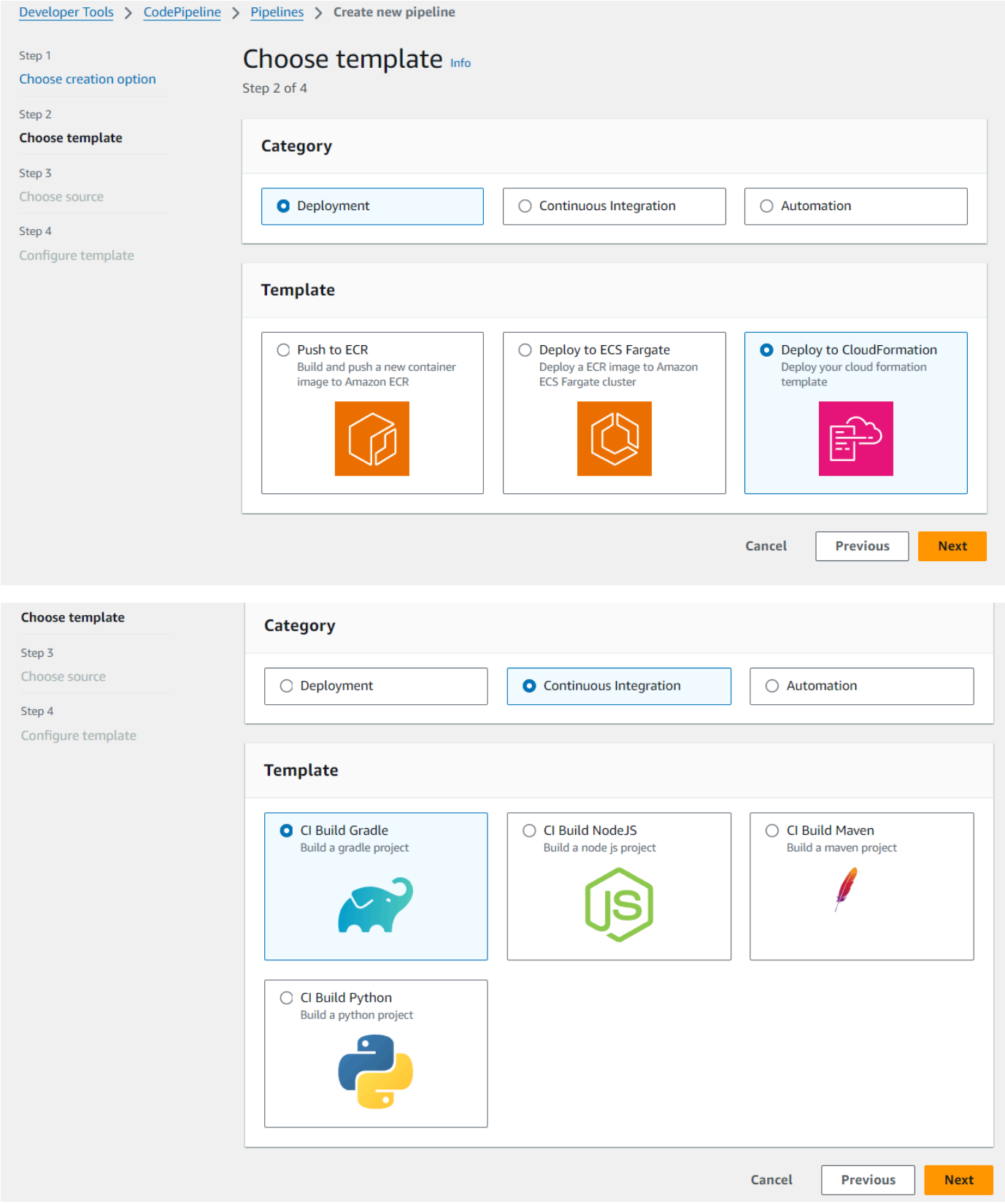
Etapa 2: escolher o modelo

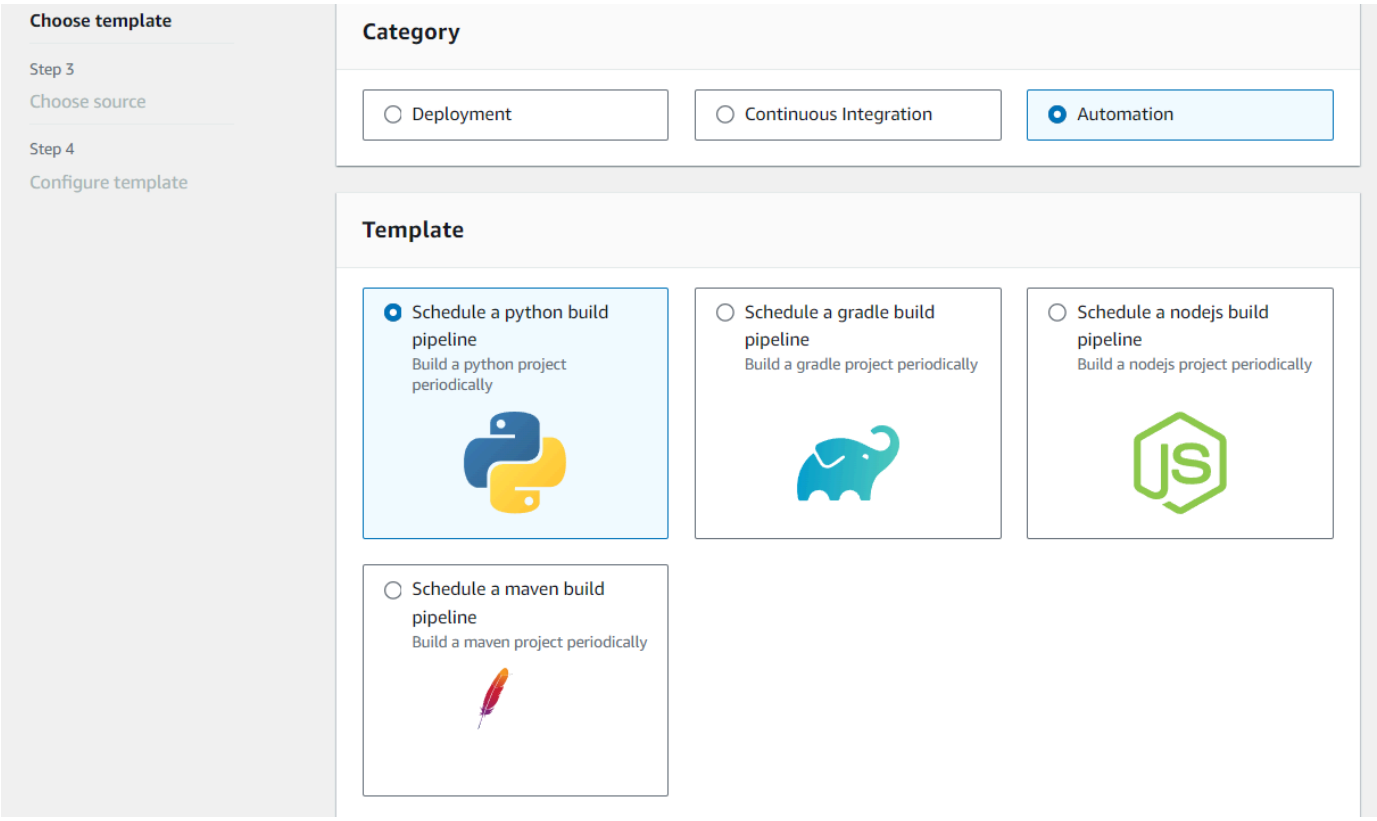
Escolha um modelo para criar um pipeline com um estágio de implantação, automação ou um pipeline de CI.

1. Na página Etapa 1: Escolher opção de criação, faça o seguinte e, em seguida, escolha Avançar:

   - Selecione Implantação se você planeja criar um estágio de implantação. Veja as opções de modelos que são implantados no ECR ou CloudFormation. Neste exemplo, escolha Implantação e, em seguida, escolha implantar em CloudFormation.
   - Selecione Integração contínua se você planeja criar um pipeline de CI. Veja as opções de pipelines de CI, como criar no Gradle.
   - Selecione Automação se você planeja criar um pipeline automatizado. Veja as opções de automação, como agendar uma compilação em python.

2.


Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose creation option
Step 2
Choose template
Step 3
Choose source
Step 4
Configure template
Choose template Info
Step 2 of 4
Category
Deployment
Continuous Integration
Automation
Template
Push to ECR
Build and push a new container image to Amazon ECR
Deploy to ECS Fargate
Deploy a ECR image to Amazon ECS Fargate cluster
Deploy to CloudFormation
Deploy your cloud formation template
Cancel
Previous
Next
Choose template
Step 3
Choose source
Step 4
Configure template
Category
Deployment
Continuous Integration
Automation
Template
CI Build Gradle
Build a gradle project
CI Build NodeJS
Build a node js project
CI Build Maven
Build a maven project
CI Build Python
Build a python project
Cancel
Previous
Next



Etapa 3: escolher origem

- Na página Etapa 3: escolher origem, em Provedor de fonte, escolha o provedor para o repositório em que o código-fonte está armazenado, especifique as opções necessárias e, em seguida, escolha Próxima etapa.

  - Para Bitbucket Cloud GitHub (via GitHub aplicativo), GitHub Enterprise Server ou GitLab.com GitLab autogerenciado:

    1. Em Conexão, escolha uma conexão existente ou crie uma nova. Para criar ou gerenciar uma conexão para sua ação GitHub de origem, consulteGitHub conexões.
    2. Escolha o repositório que deseja utilizar como local de origem do pipeline.

       Selecione a opção para adicionar um gatilho ou filtrar os tipos de gatilho para iniciar o pipeline. Para ter mais informações sobre como trabalhar com gatilhos, consulte Adicionar gatilho com tipos de eventos code push ou pull request. Para obter mais informações sobre a filtragem de padrões glob, consulte Trabalhar com padrões glob na sintaxe.
    3. Em Formato de artefato de saída, escolha o formato dos seus artefatos.

- Para armazenar artefatos de saída da GitHub ação usando o método padrão, escolha CodePipelinedefault. A ação acessa os arquivos do GitHub repositório e armazena os artefatos em um arquivo ZIP no repositório de artefatos do pipeline.
- Para armazenar um arquivo JSON que contém uma referência de URL ao repositório para que as ações downstream possam executar comandos Git diretamente, escolha Full clone (Clone completo). Esta opção só pode ser usada por ações downstream do CodeBuild .

  Se você escolher essa opção, precisará atualizar as permissões para sua função de serviço CodeBuild do projeto, conforme mostrado emSolução de problemas CodePipeline. Para um tutorial que mostra como usar a opção Clone completo, consulte Tutorial: use o clone completo com uma fonte de GitHub pipeline.

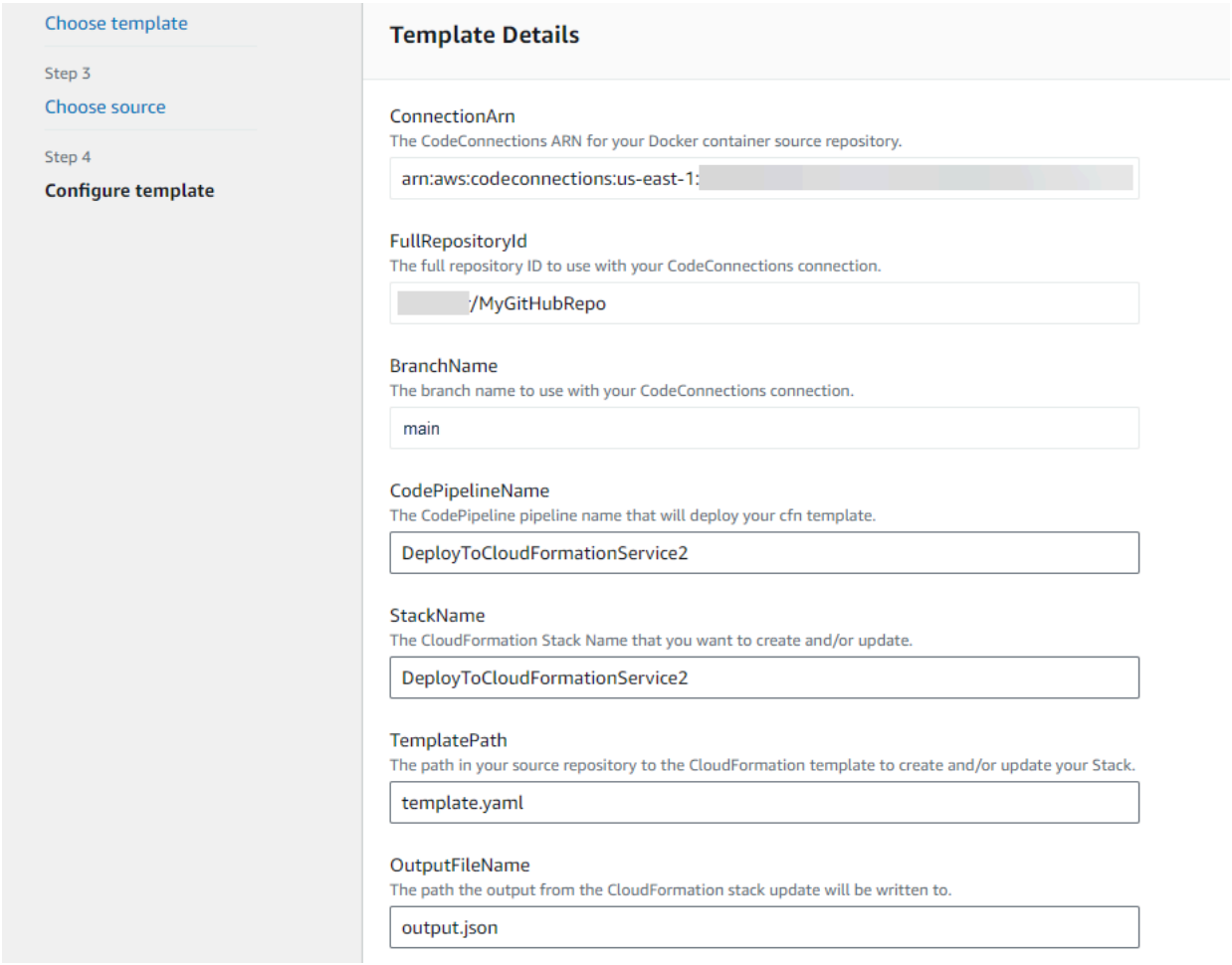
- Para Amazon ECR:
  - Em Nome do repositório, selecione o nome do repositório do Amazon ECR.
  - Em Image tag (Tag da imagem), especifique o nome da imagem e a versão, caso seja diferente da ÚLTIMA.
  - Em Artefatos de saída, escolha o artefato de saída padrão, como MyApp, que contém o nome da imagem e as informações de URI do repositório que você deseja usar na próxima etapa.

  Quando você inclui um estágio de origem do Amazon ECR no pipeline, a ação de origem gera um arquivo `imageDetail.json` como artefato de saída quando você confirma uma alteração. Para mais informações sobre o arquivo `imageDetail.json`, consulte Detail.json arquivo de imagem para ações de blue/green implantação do Amazon ECS.

> **Note**
>
> O objeto e o tipo de arquivo devem ser compatíveis com o sistema de implantação que você planeja usar (por exemplo, Elastic Beanstalk ou). CodeDeploy Os tipos de arquivos compatíveis podem incluir .zip, .tar e .tgz. Para obter mais informações sobre os tipos de contêiner compatíveis com o Elastic Beanstalk, consulte Personalizar e configurar ambientes do Elastic Beanstalk e Plataformas compatíveis. Para obter mais informações sobre como implantar revisões com CodeDeploy, consulte Carregando a revisão do seu aplicativo e preparando uma revisão.

Etapa 4: configurar o modelo

Neste exemplo, a implantação em CloudFormation foi selecionada. Nesta etapa, adicione a configuração para seu modelo.



1. Em Etapa 4: configurar o modelo, em Nome da pilha, insira um nome para o pipeline.
2. Edite a política do IAM de espaço reservado para as permissões que se aplicam ao modelo.
3. Selecione Criar pipeline usando o modelo
4. Uma mensagem é exibida informando que os recursos de pipeline estão sendo criados.

Etapa 5: exibir o pipeline

- Agora que você criou seu pipeline, você pode visualizá-lo no CodePipeline console e ver a pilha em CloudFormation. O pipeline começa a ser executado depois de ser criado. Para obter mais informações, consulte Veja os pipelines e os detalhes em CodePipeline. Para obter

mais informações sobre como fazer alterações em seu pipeline, acesse Editar um pipeline no CodePipeline.

Editar um pipeline no CodePipeline

Um pipeline descreve o processo de liberação que deseja que o AWS CodePipeline siga, incluindo os estágios e as ações que devem ser realizadas. É possível editar um pipeline para adicionar ou remover esses elementos. No entanto, ao editar um pipeline, valores, como o nome do pipeline ou metadados do pipeline não poderão ser alterados.

É possível editar o tipo, as variáveis e os gatilhos do pipeline usando a página de edição do pipeline. Também é possível adicionar ou alterar estágios e ações no pipeline.

Ao contrário da criação de um pipeline, a edição de um pipeline não executa novamente a revisão mais recente no pipeline. Se desejar executar a revisão mais recente em um pipeline que acabou de editar, você deverá executá-la manualmente outra vez. Caso contrário, o pipeline editado será executado na próxima vez que você fizer uma alteração em um local de origem configurado no estágio de origem. Para mais informações, consulte Iniciar um pipeline manualmente.

Você pode adicionar ações ao pipeline que estejam em outra região da AWS diferente em seu pipeline. Quando um AWS service (Serviço da AWS) é o provedor de uma ação e esse tipo de ação/tipo de provedor não está na mesma região da AWS que o pipeline, essa é uma ação entre regiões. Para obter mais informações sobre ações entre regiões, consulte Adicione uma ação entre regiões em CodePipeline.

O CodePipeline usa métodos de detecção de alterações para iniciar o pipeline quando uma alteração em seu código-fonte é enviada. Esses métodos de detecção são baseados em tipo de origem:

- O CodePipeline usa o Amazon CloudWatch Events para detectar alterações no repositório de origem do CodeCommit ou no bucket de origem do Amazon S3.

> **Note**
>
> Os recursos de detecção de alterações são criados automaticamente quando você usa o console. Quando o console é usado para criar ou editar um pipeline, os recursos adicionais são criados para você. Se usar a AWS CLI para criar o pipeline, você mesmo deverá criar os recursos adicionais. Para obter mais informações sobre a criação ou atualização de um pipeline do CodeCommit, consulte Crie uma EventBridge regra para uma CodeCommit fonte

(CLI). Para obter mais informações sobre o uso da CLI para criar ou atualizar um pipeline do Amazon S3, consulte Crie uma EventBridge regra para uma fonte (CLI) do Amazon S3.

Tópicos

- Editar um pipeline (console)
- Editar um pipeline (AWS CLI)

Editar um pipeline (console)

Você pode usar o console do CodePipeline para adicionar, editar ou remover estágios em um pipeline e para adicionar, editar ou remover ações em uma etapa.

Quando você atualiza um pipeline, o CodePipeline conclui normalmente todas as ações em execução e, em seguida, falha nas etapas e execuções de pipeline em que as ações em execução foram concluídas. Quando um pipeline for atualizado, você precisará executá-lo novamente. Para obter mais informações sobre a execução de um pipeline, consulte Iniciar um pipeline manualmente.

Para editar um pipeline

1. Faça login no Console de gerenciamento da AWS e abra o console do CodePipeline em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes de todos os pipelines associados à sua conta da AWS são exibidos.
2. Em Nome, selecione o nome do pipeline que você deseja editar. Isso abrirá um visão detalhada do pipeline, incluindo o estado de cada uma das ações em cada estágio do pipeline.
3. Na página de detalhes do pipeline, selecione Editar.
4. Para editar o tipo de pipeline, selecione Editar no cartão Editar: propriedades do pipeline. Escolha uma das opções a seguir e selecione Concluído.

   - Os pipelines do tipo V1 têm uma estrutura JSON que contém parâmetros padrão no nível do pipeline, do estágio e da ação.
   - Os pipelines do tipo V2 têm a mesma estrutura do tipo V1, além do suporte adicional a parâmetros, como gatilhos e variáveis em nível de pipeline.

   Os tipos de pipeline diferem em características e preços. Para obter mais informações, consulte Tipos de pipeline.

5. Para editar as variáveis do pipeline, selecione Editar variáveis no cartão Editar: Variáveis. Adicione ou altere variáveis para o nível do pipeline e selecione Concluído.

   Para obter mais informações sobre as variáveis no nível do pipeline, consulte Referência de variáveis. Para assistir a um tutorial com uma variável no nível do pipeline que é passada no momento da execução do pipeline, consulte. Tutorial: Usar variáveis no nível do pipeline

   > **Note**
   >
   > Embora seja opcional adicionar variáveis no nível do pipeline, para um pipeline especificado com variáveis no nível do pipeline em que nenhum valor é fornecido, a execução do pipeline falhará.

6. Para editar os gatilhos do pipeline, selecione Editar gatilhos no cartão Editar: Gatilhos. Adicione ou altere os gatilhos e, depois, selecione Concluído.

   Para obter mais informações sobre como adicionar gatilhos, consulte as etapas para criar uma conexão com Bitbucket Cloud, GitHub (por meio do aplicativo do GitHub), GitHub Enterprise Server, GitLab.com ou GitLab autogerenciado, como GitHub conexões.

7. Para editar estágios e ações na página Editar, siga um destes procedimentos:

   - Para editar um estágio, escolha Edit stage (Editar estágio). É possível adicionar ações em série e paralelo com as ações existentes:

     Você também pode editar ações nessa visualização, escolhendo o ícone de edição das ações. Para excluir uma ação, selecione o ícone de exclusão dessa ação.

   - Para editar uma ação, escolha o ícone de edição para a ação e depois altere os valores em Edit action. Itens marcados com um asterisco (*) são obrigatórios.
     - Para um nome de repositório e a ramificação do CodeCommit, será exibida uma mensagem mostrando a regra do Amazon CloudWatch Events a ser criada para esse pipeline. Se você remover a origem do CodeCommit, será exibida uma mensagem mostrando a regra do Amazon CloudWatch Events a ser excluída.
     - Para um bucket de origem do Amazon S3, será exibida uma mensagem mostrando a regra do Amazon CloudWatch Events e a trilha do AWS CloudTrail a serem criadas para esse pipeline. Se você remover a origem do Amazon S3, será exibida uma mensagem mostrando a regra do Amazon CloudWatch Events e a trilha do AWS CloudTrail a serem excluídas. Se

a trilha do AWS CloudTrail estiver em uso por outros pipelines, ela não será removida e o evento de dados será excluído.

- Para adicionar um estágio, selecione + Add stage (+ Adicionar estágio) no ponto do pipeline onde você deseja adicionar um estágio. Forneça um nome para o estágio e depois adicione pelo menos uma ação. Itens marcados com um asterisco (*) são obrigatórios.
- Para excluir um estágio, selecione o ícone de exclusão daquele estágio. O estágio e todas as suas ações serão excluídos.
- Para configurar um estágio para reverter automaticamente em caso de falha, escolha Editar estágio e, em seguida, escolha a caixa de seleção Configurar reversão automática em caso de falha do estágio.

Por exemplo, se você quisesse adicionar uma ação serial a um estágio de um pipeline:

1. No estágio em que você quer adicionar a ação, selecione Edit stage (Editar estágio) e depois selecione + Add action group (+ Adicionar grupo de ação).
2. Em Edit action (Editar ação), em Action name (Nome da ação), insira o nome da ação. A lista Action provider (Fornecedor de ação) exibe opções de fornecedor por categoria. Procure a categoria (por exemplo, Deploy (Implantar)). Na categoria, escolha o provedor, (por exemplo, AWS CodeDeploy). Em Região, escolha a região da AWS na qual o recurso foi criado ou você planeja criá-lo. O campo Região designa o local em que os recursos da AWS serão criados para esse tipo de ação e de provedor. Esse campo é exibido apenas para ações em que o provedor de ação é um AWS service (Serviço da AWS). Por padrão, o campo Região assume a mesma região da AWS de seu pipeline.

   Para obter exemplos de como adicionar provedores de ação e usar os campos padrão para cada provedor, consulte Criar um pipeline personalizado (console).

   Para adicionar o CodeBuild como uma ação de compilação ou uma ação de teste em um estágio, consulte Usar o CodePipeline com o CodeBuild para testar o código e executar compilações no Guia do usuário do CodeBuild.

   > **Note**
   >
   > Alguns provedores de ação, como o GitHub, exigem que você se conecte ao site do provedor antes de concluir a configuração da ação. Ao se conectar ao site de um

provedor, certifique-se de utilizar as credenciais para aquele site. Não utilize suas credenciais da AWS.

3. Quando você finalizar a configuração da ação, selecione Save (Salvar).

> **Note**
>
> Não é possível renomear um estágio a partir da visualização do console. Você pode adicionar um estágio com o nome que deseja alterar e depois excluir o antigo. Certifique-se de ter adicionado todas as ações que deseja ao estágio antes de excluir o antigo.

8. Quando terminar de editar seu pipeline, selecione Save (Salvar) para voltar à página de resumo.

> **Important**
>
> Após salvar as alterações, não será possível desfazê-las. Você deve editar o pipeline novamente. Se uma revisão estiver em execução através do pipeline quando você salvar suas alterações, a execução não será concluída. Se deseja que uma confirmação ou alteração específica seja executada pelo pipeline editado, você deve executá-la manualmente pelo pipeline. Caso contrário, a próxima confirmação ou alteração será executada automaticamente através do pipeline.

9. Para testar a ação, escolha Lançar alteração para processar essa confirmação no pipeline e confirmar uma alteração na origem especificada no estágio de origem do pipeline. Ou siga as etapas em Iniciar um pipeline manualmente para usar o AWS CLI para liberar manualmente uma alteração.

Editar um pipeline (AWS CLI)

Você pode usar o comando update-pipeline para editar um pipeline.

Quando você atualiza um pipeline, o CodePipeline conclui normalmente todas as ações em execução e, em seguida, falha nas etapas e execuções de pipeline em que as ações em execução foram concluídas. Quando um pipeline for atualizado, você precisará executá-lo novamente. Para obter mais informações sobre a execução de um pipeline, consulte Iniciar um pipeline manualmente.

> **Important**
>
> Embora seja possível usar a AWS CLI para editar pipelines que incluem ações de parceiro, você não deve editar manualmente o JSON de uma ação de parceiro. Se fizer isso, a ação de parceiro falhará após a atualização do pipeline.

Para editar um pipeline

1. Abra uma sessão de terminal (Linux, macOS, or Unix) ou prompt de comando (Windows) e execute o comando get-pipeline para copiar a estrutura do pipeline em um arquivo JSON. Por exemplo, para um pipeline nomeado **`MyFirstPipeline`**, insira o seguinte comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando retorna nada, mas o arquivo que você criou deve aparecer no diretório onde você executou o comando.

2. Abra o arquivo JSON em qualquer editor de texto plano e modifique a estrutura do arquivo para refletir as alterações que você deseja fazer no pipeline. Por exemplo, você pode adicionar ou remover etapas ou adicionar outra ação a um estágio existente.

   O exemplo a seguir mostra como você pode adicionar outro estágio de implantação no arquivo pipeline.json. Este estágio é executado após o primeiro estágio de implantação, designado como *`Staging`* (Preparação).

   > **Note**
   >
   > Essa é apenas uma parte do arquivo e não a estrutura completa. Para obter mais informações, consulte CodePipeline referência de estrutura de tubulação.

   ```
   ,
           {
               "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 obter informações sobre como usar a ILC para adicionar uma ação de aprovação a um pipeline, consulte Adicione uma ação de aprovação manual a um funil no CodePipeline.

Verifique se o parâmetro `PollForSourceChanges` está definido como mostrado a seguir no arquivo JSON:

```
                "PollForSourceChanges": "false",
```

O CodePipeline usa o Amazon CloudWatch Events para detectar alterações no repositório e na ramificação de origem do CodeCommit ou no bucket de origem do Amazon S3. A próxima etapa inclui instruções para criar esses recursos manualmente. Definir o sinalizador como `false` desativa verificações periódicas, que não são necessárias ao se usar os métodos de detecção de alterações recomendados.

3. Para adicionar uma ação de compilação, teste ou implantação em uma região diferente do seu pipeline, é necessário adicionar o seguinte à sua estrutura de pipeline. Para ter instruções detalhadas, consulte Adicione uma ação entre regiões em CodePipeline.

   - Adicione o parâmetro de `Region` à ação da estrutura do pipeline.
   - Use o parâmetro `artifactStores` para especificar um bucket de artefatos para cada região na qual exista uma ação.

4. Se você estiver trabalhando com a estrutura do pipeline recuperada usando o comando get-pipeline, você deve modificar a estrutura no arquivo JSON. Você deve remover as linhas `metadata` do arquivo para que o comando update-pipeline possa usá-lo. Remova a seção da estrutura do pipeline no arquivo JSON (as linhas de `"metadata": { }` e os campos `"created"`, `"pipelineARN"` e `"updated"`).

   Por exemplo, remova as seguintes linhas da estrutura:

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

   Salve o arquivo.

5. Se usar a ILC para editar um pipeline, você deverá gerar manualmente os recursos de detecção de alterações recomendados para o seu pipeline:

   - Para um repositório do CodeCommit, é necessário criar a regra do CloudWatch Events, conforme descrito em Crie uma EventBridge regra para uma CodeCommit fonte (CLI).
   - Para uma origem do Amazon S3, é necessário criar a regra do CloudWatch Events e uma trilha do AWS CloudTrail, conforme descrito em Conectando-se às ações de origem do Amazon S3 que usam EventBridge e AWS CloudTrail.

6. Para aplicar suas alterações, execute o comando update-pipeline especificando o arquivo JSON do pipeline:

   > **Important**
   >
   > Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

   Este comando retorna toda a estrutura do pipeline editado.

   > **Note**
   >
   > O comando update-pipeline interrompe o pipeline. Se uma revisão estiver sendo executada pelo pipeline quando você executar o comando update-pipeline, essa execução será interrompida. Você deve iniciar manualmente o pipeline para executar a revisão através do pipeline atualizado.

7. Abra o console do CodePipeline e selecione o pipeline recém-editado.

   O pipeline exibirá suas alterações. Na próxima vez que você alterar a localização de origem, o pipeline executará a revisão através da estrutura revisada do pipeline.

8. Para executar manualmente a última revisão pela estrutura revisada do pipeline, execute o comando start-pipeline-execution. Para obter mais informações, consulte Iniciar um pipeline manualmente.

Para obter mais informações sobre a estrutura de um pipeline e os valores esperados, consulte CodePipeline referência de estrutura de tubulação e Referência da API do AWS CodePipeline.

Veja os pipelines e os detalhes em CodePipeline

Você pode usar o AWS CodePipeline console ou o AWS CLI para ver detalhes sobre os pipelines associados à sua AWS conta.

Tópicos

- Visualizar pipelines (console)
- Visualizar detalhes da ação em um pipeline (console)
- Visualizar o ARN do pipeline e o ARN do perfil de serviço (console)
- Visualizar detalhes e histórico do pipeline (CLI)
- Exibir resultados das regras aplicadas às condições de estágio no histórico de execuções

Visualizar pipelines (console)

Você pode visualizar o status, as transições e as atualizações de artefatos para um pipeline.

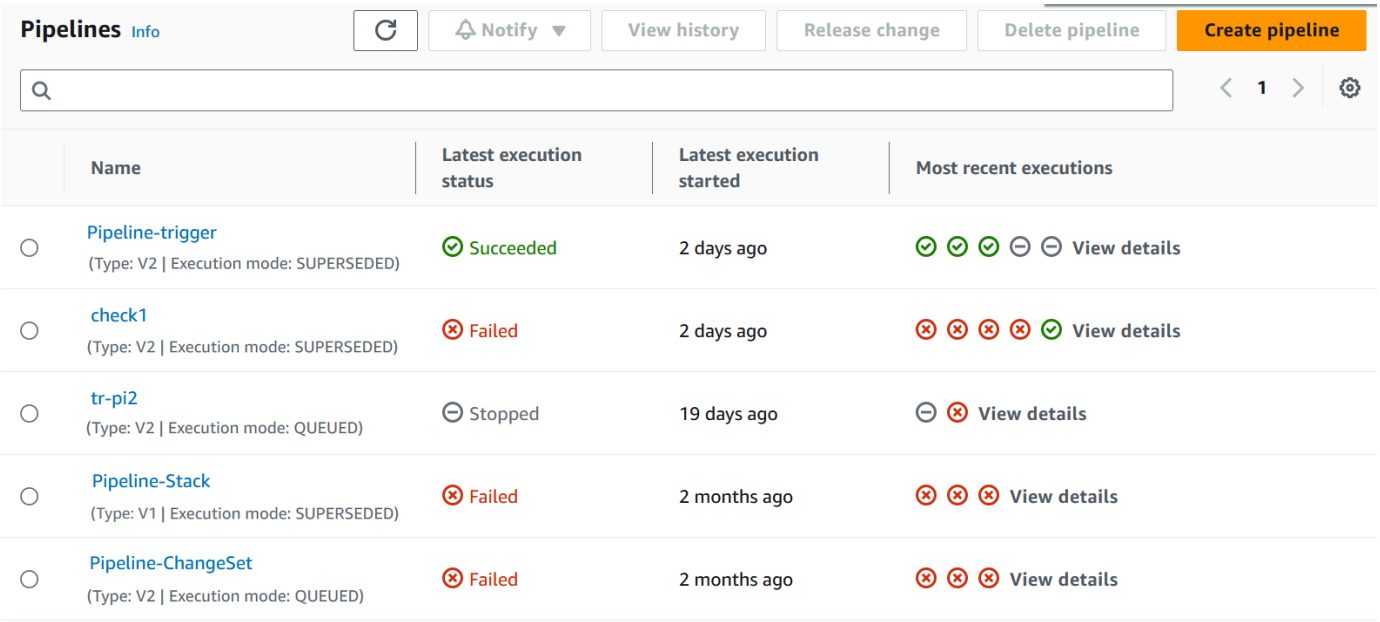
> **Note**
>
> Depois de uma hora, a visualização detalhada de um pipeline vai parar de ser atualizada automaticamente no navegador. Para ver as informações atuais, atualize a página.

Como visualizar pipelines

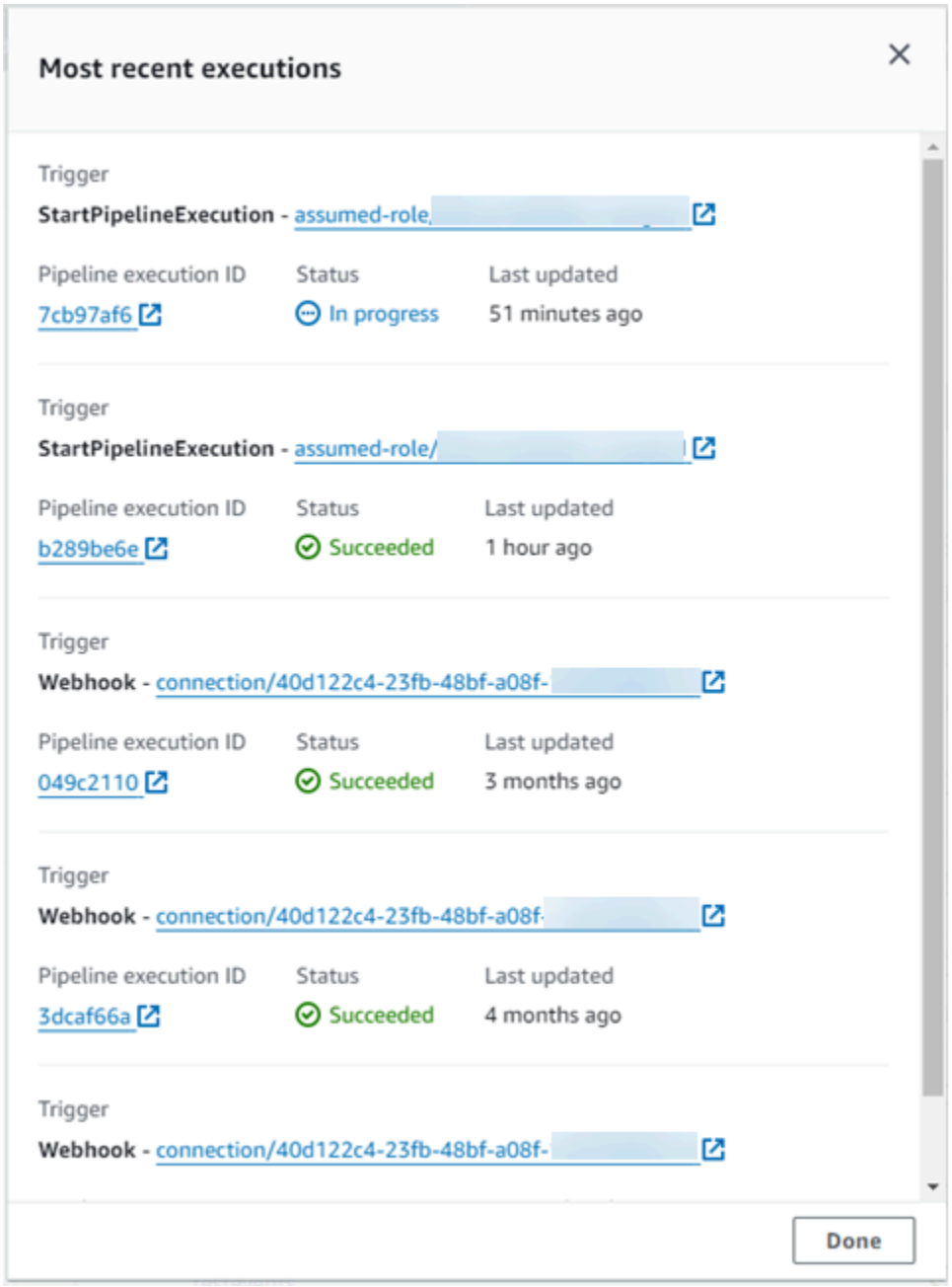
1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon. com/codesuite/codepipeline/home.

   A página Pipelines é exibida. Uma lista de todos os pipelines dessa região é mostrada.

   O nome, tipo, status, versão, data de criação e data da última modificação de todos os pipelines associados à sua AWS conta são exibidos, junto com o horário de execução iniciado mais recentemente.
2. O status das cinco execuções mais recentes é mostrado.



Selecione Visualizar detalhes ao lado de uma linha específica para exibir uma caixa de diálogo de detalhes listando as execuções mais recentes.

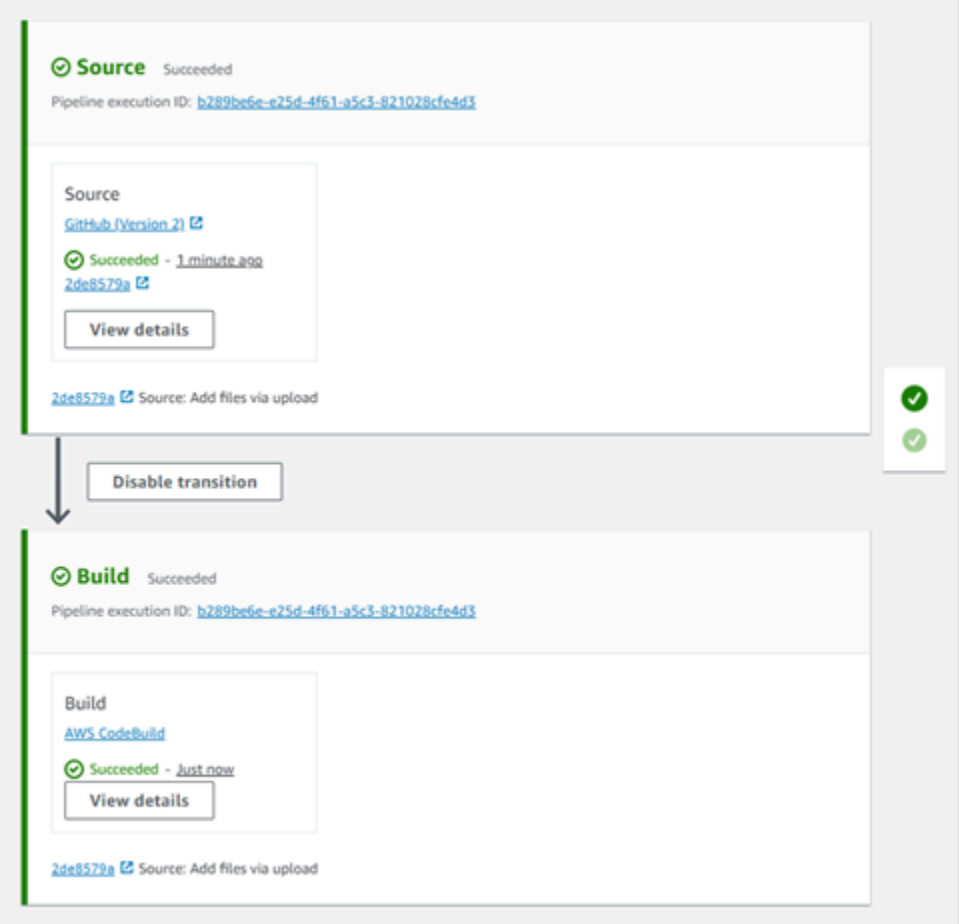


Para visualizar detalhes sobre as execuções mais recentes para o pipeline, selecione View history (Visualizar histórico). Para execuções anteriores, você pode visualizar os detalhes da revisão associados aos artefatos de origem, como execução IDs, status, horários de início e término, duração, confirmação IDs e mensagens.

> **Note**
>
> Em um pipeline no modo de execução PARALLEL, a visualização principal não mostra a estrutura do pipeline nem as execuções que estão em andamento. Em um pipeline no

modo de execução PARALLEL, a estrutura do pipeline pode ser acessada selecionando o ID da execução desejada na página de histórico de execução. No painel de navegação à esquerda, selecione Histórico, escolha o ID da execução paralela e visualize o pipeline na guia Visualização.

3. Para visualizar os detalhes de um único pipeline, em Nome, selecione o pipeline. Você verá uma visão detalhada do pipeline, incluindo o estado de cada ação em cada estágio e o estado das transições.



A visualização gráfica exibe as seguintes informações para cada estágio:

- O nome do estágio.
- Todas as ações configuradas para o estágio.
- O estado de transições entre estágios (habilitada ou desabilitada), conforme indicado pela estado da seta entre os estágios. Uma transição ativada é indicada por uma seta com um botão Disable transition (Desativar transição) próximo. Uma transição desativada é indicada por uma seta com um realce e um botão Enable transition (Ativar transição) próximo.

- Uma barra de cores indica o status do estágio:
  - Cinza: ainda não executado
  - Blue: em andamento
  - Verde: bem-sucedido
  - Vermelho: falhou

A visualização gráfica também exibe as seguintes informações sobre as ações em cada estágio:

- O nome da ação.
- O provedor da ação, como CodeDeploy.
- Quando a ação foi executada pela última vez.
- Se a ação foi bem-sucedida ou não.
- Links para outros detalhes sobre a última execução da ação, quando disponível.
- Detalhes sobre as revisões de origem que estão sendo executadas durante a execução mais recente do pipeline no estágio ou, para CodeDeploy implantações, as revisões de origem mais recentes que foram implantadas nas instâncias de destino.
- Um botão Visualizar detalhes que abre uma caixa de diálogo com detalhes sobre a execução da ação, os logs e a configuração da ação.

> **Note**
>
> A guia Registros está disponível para CodeBuild CloudFormation ações que foram executadas na conta do pipeline.

4. Para visualizar os detalhes do provedor da ação, selecione o provedor. Por exemplo, no exemplo anterior de pipeline, se você escolher entre CodeDeploy os estágios de preparação ou produção, a página do CodeDeploy console do grupo de implantação configurado para esse estágio será exibida.
5. O andamento de uma ação é exibido ao lado de uma ação em andamento (indicado por uma mensagem Em andamento). Se a ação estiver em andamento, você vê o progresso incremental e as etapas ou ações à medida que ocorrem.
6. Para aprovar ou rejeitar ações que foram configuradas para aprovação manual, selecione Revisar.

7. Para tentar novamente ações de um estágio que não foram concluídas com êxito, selecione Tentar novamente.
8. O status da última vez em que a ação foi executada, incluindo os resultados da ação, (Com êxito ou Falhou) são exibidos.

Visualizar detalhes da ação em um pipeline (console)

É possível visualizar os detalhes de um pipeline, incluindo detalhes das ações em cada estágio.

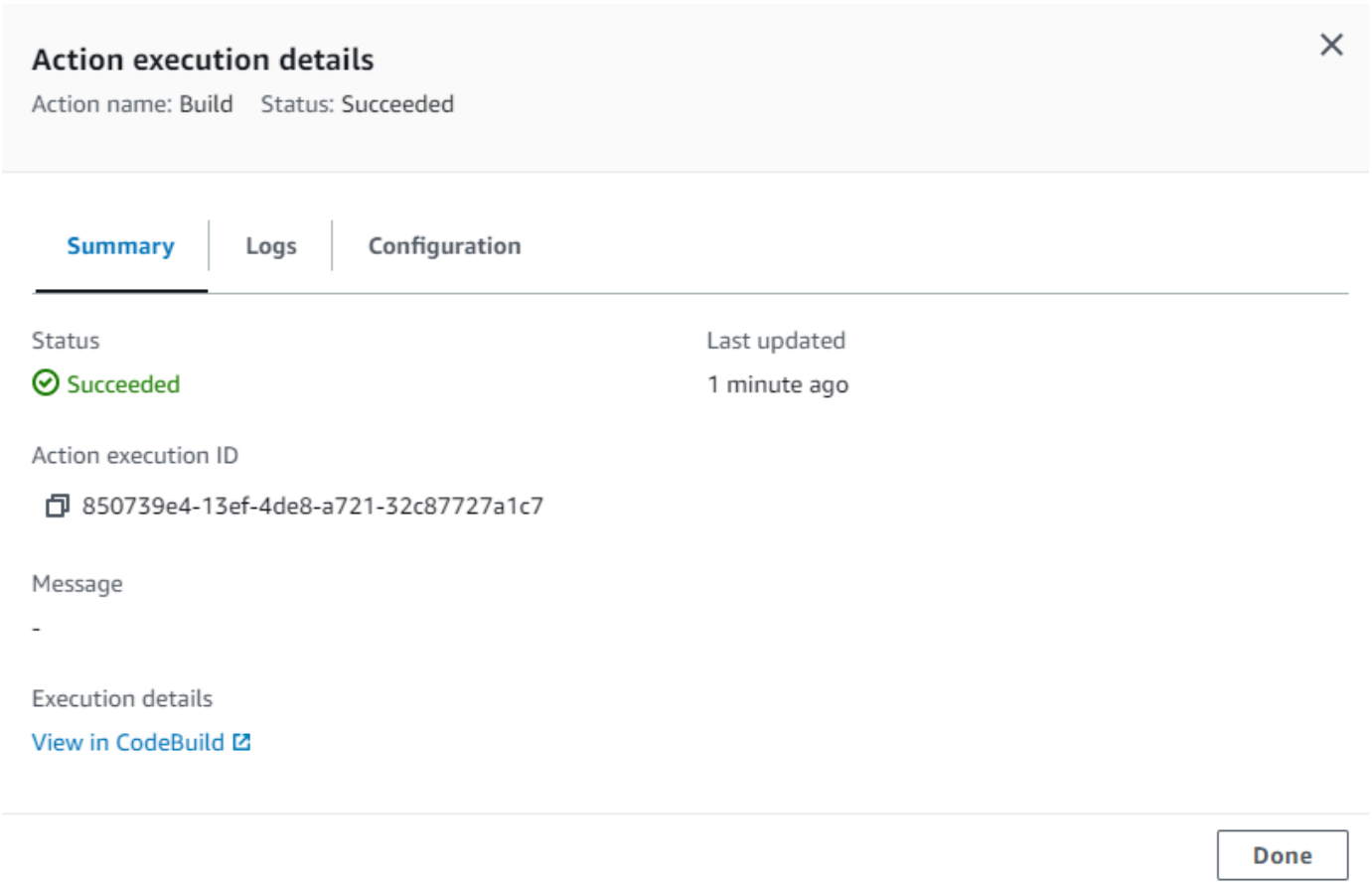
> **Note**
>
> Depois de uma hora, a visualização detalhada de um pipeline vai parar de ser atualizada automaticamente no navegador. Para ver as informações atuais, atualize a página.

Como visualizar detalhes da ação em um pipeline

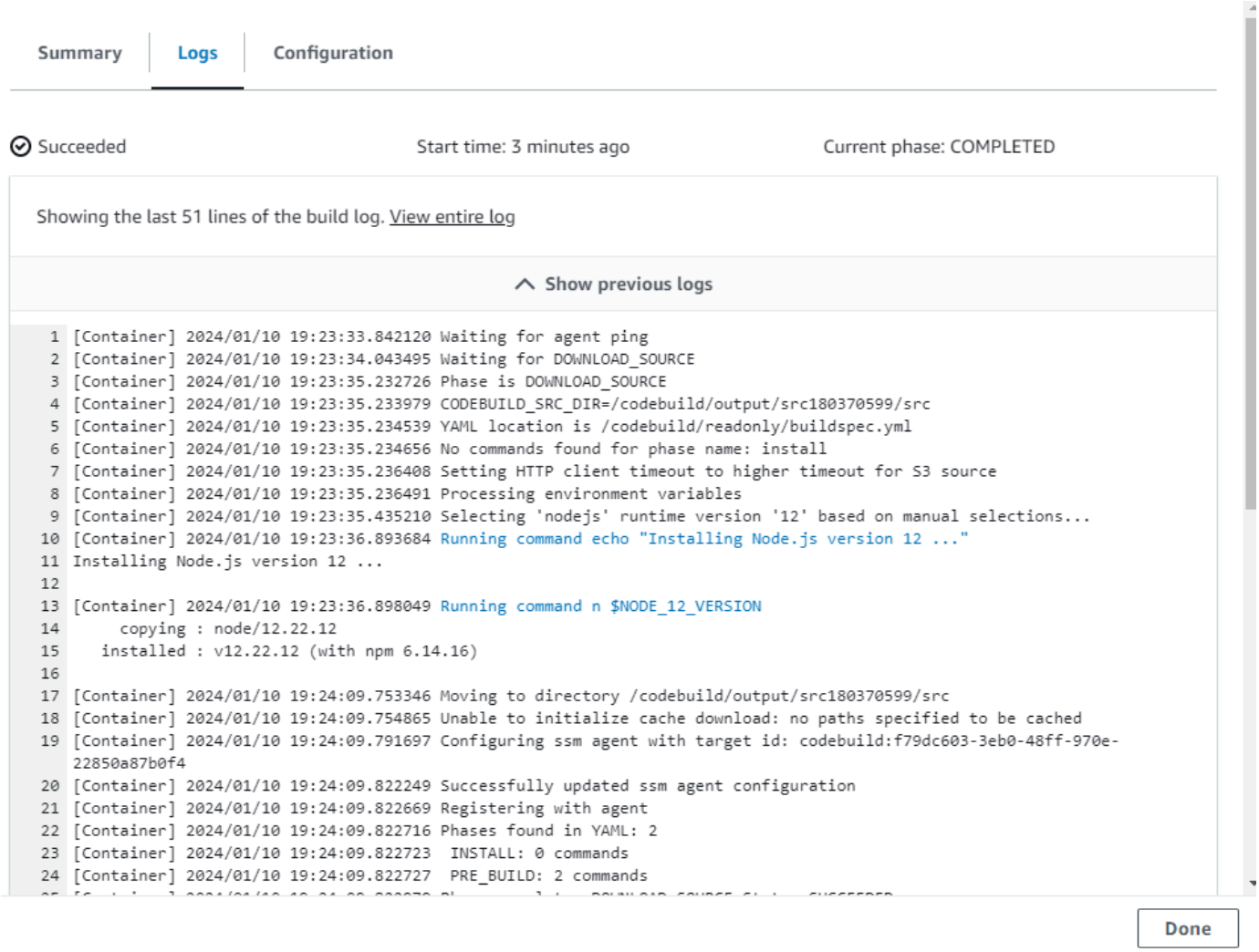
1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon. com/codesuite/codepipeline/home.

   A página Pipelines é exibida.
2. Em qualquer ação, selecione Visualizar detalhes para abrir uma caixa de diálogo com detalhes sobre a execução da ação, os logs e a configuração da ação.

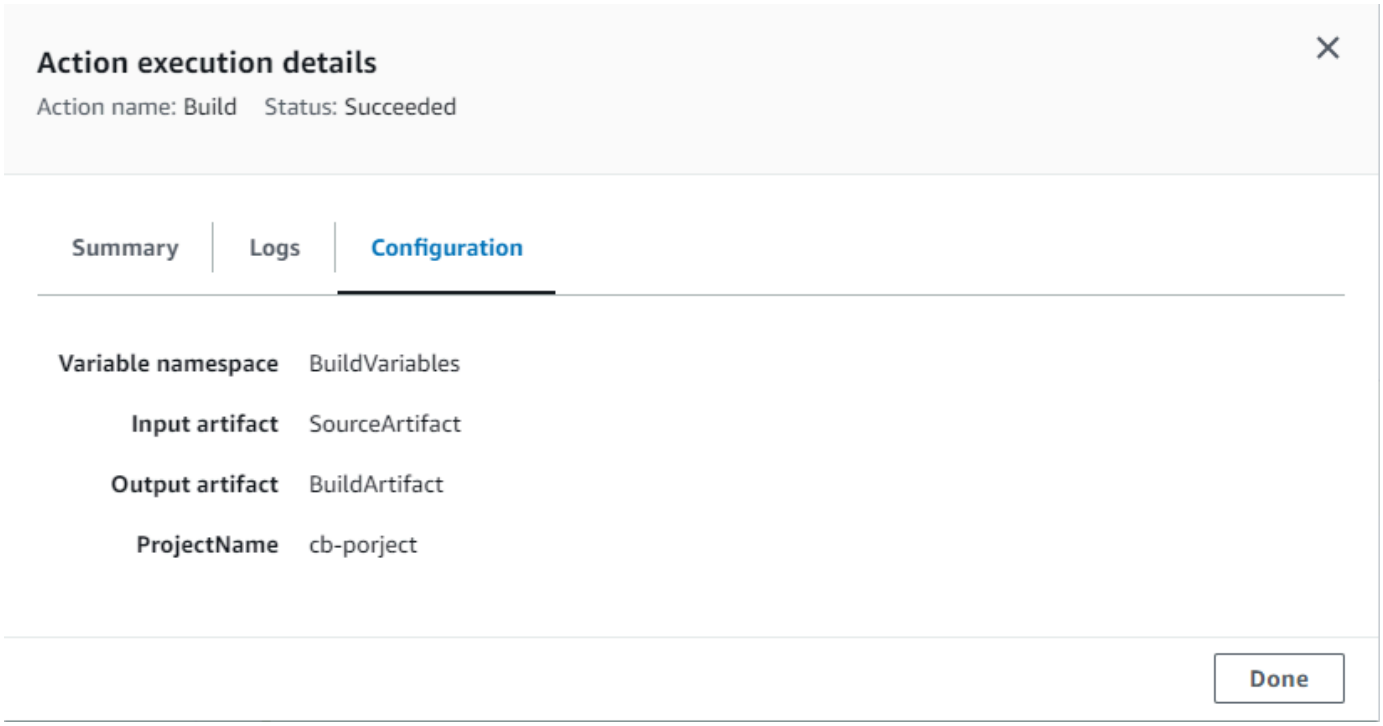
   > **Note**
   >
   > A guia Registros está disponível para CloudFormation ações CodeBuild e ações.
3. Para ver o resumo da ação em um estágio de um pipeline, selecione Visualizar detalhes da ação e, depois, escolha a guia Resumo.



4. Para ver os logs de uma ação com logs, selecione Visualizar detalhes da ação e, depois, escolha a guia Logs.



5. Para ver os detalhes da configuração de uma ação, selecione a guia Configuração.



Visualizar o ARN do pipeline e o ARN do perfil de serviço (console)

É possível usar o console para visualizar as configurações do pipeline, como o ARN do pipeline, o ARN do perfil de serviço e o armazenamento de artefatos do pipeline.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http:// console.aws.amazon. com/codesuite/codepipeline/home.

   Os nomes de todos os pipelines associados à sua AWS conta serão exibidos.
2. Selecione o nome do pipeline e, depois, escolha Configurações no painel de navegação esquerdo. A página mostra o seguinte:

   - O nome do pipeline
   - O nome do recurso da Amazon (ARN) do pipeline

     O ARN do pipeline é construído neste formato:

     arn: aws:codepipeline::: *region account pipeline-name*

     Exemplo de ARN do pipeline:

     `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

- O ARN da função de CodePipeline serviço para seu pipeline
- A versão do pipeline
- O nome e o local do armazenamento de artefatos do pipeline

Visualizar detalhes e histórico do pipeline (CLI)

Você pode executar os seguintes comandos para visualizar detalhes sobre seus pipelines e execuções de pipelines:

- list-pipelinescomando para ver um resumo de todos os pipelines associados à sua AWS conta.
- Comando get-pipeline para revisar os detalhes de um único pipeline.
- list-pipeline-executions para visualizar resumos das execuções mais recentes de um pipeline.
- get-pipeline-execution para visualizar informações sobre a execução de um pipeline, incluindo os detalhes sobre os artefatos, o ID de execução do pipeline e o nome, a versão e o status do pipeline.
- Comando get-pipeline-state para visualizar o pipeline, o estágio e o status da ação.
- list-action-executions para visualizar detalhes de execução da ação para um pipeline.

1. Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use o AWS CLI para executar o comando: list-pipelines

   ```
   aws codepipeline list-pipelines
   ```

   Este comando retorna uma lista de todos os pipelines associados à sua conta da AWS .

2. Para visualizar detalhes sobre um pipeline, execute o comando get-pipeline, especificando o nome exclusivo do pipeline. Por exemplo, para visualizar detalhes sobre um pipeline designado *MyFirstPipeline*, insira o seguinte:

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

   Este comando retorna a estrutura do pipeline.

Exibir resultados das regras aplicadas às condições de estágio no histórico de execuções

É possível verificar os resultados das regras em uma execução em que uma condição de estágio ativou uma regra e produziu um resultado para o estágio, como falha ou reversão.

Os valores de status válidos para condições e regras são os seguintes: `InProgress | Failed | Errored | Succeeded | Cancelled | Abandoned | Overridden`

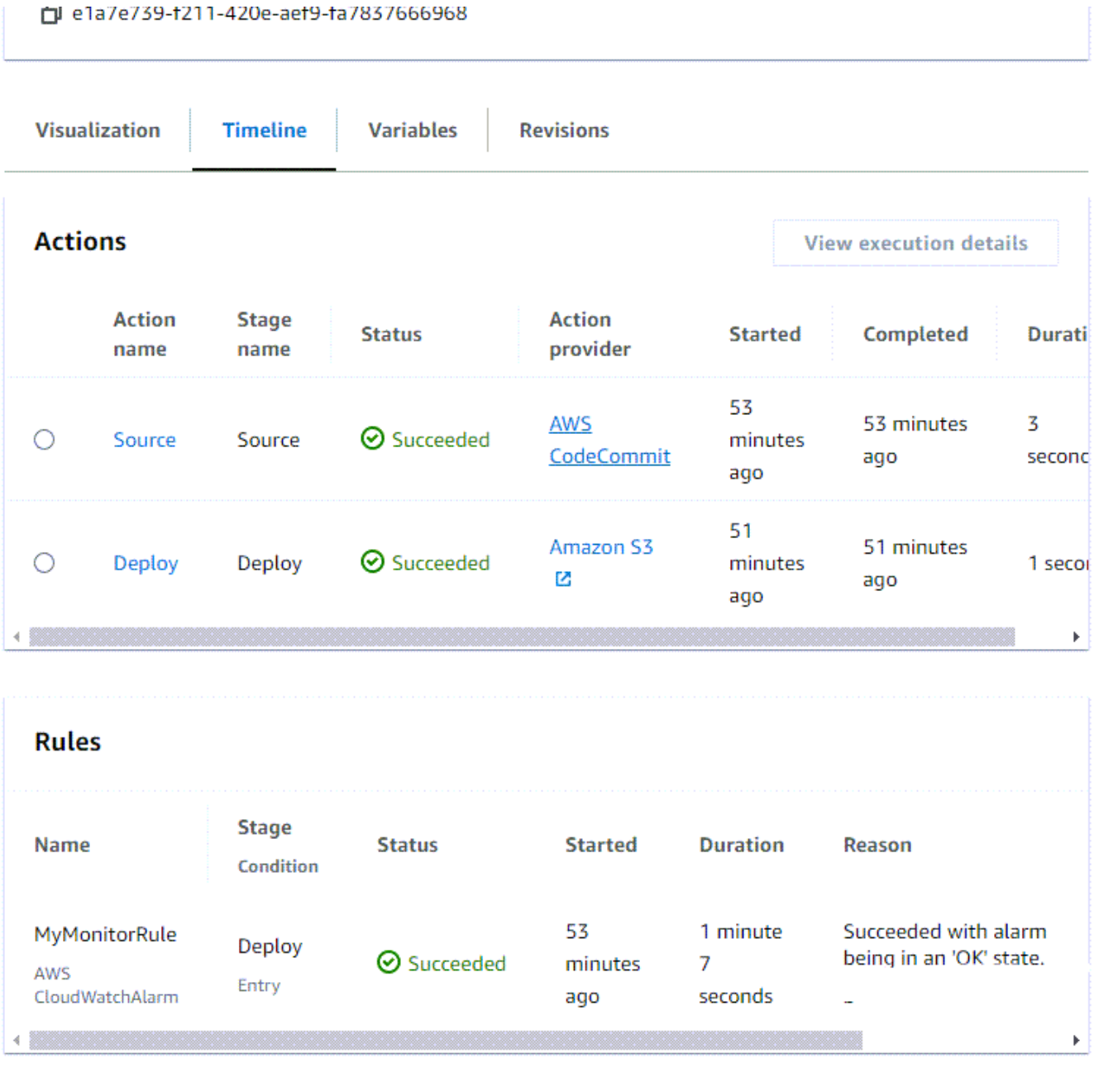
Exibir resultados de regras para condições de estágio no histórico de execução (console)

O console permite verificar os resultados das regras em uma execução na qual uma condição de estágio ativou uma regra e produziu um resultado para o estágio.

Exibir os resultados das regras para as condições do estágio (console)

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http:// console.aws.amazon. com/codesuite/codepipeline/home.

   Os nomes e status de todos os pipelines vinculados à sua Conta da AWS são exibidos.
2. Em Nome, selecione o nome do pipeline que você deseja ver.
3. Selecione Histórico e depois escolha a execução desejada. Na página do histórico, escolha a guia Linha do tempo. Em Regras, veja os resultados da regra para a execução.



Confira os resultados das regras para condições de estágio usando **`list-rule-executions`** (CLI)

É possível utilizar a CLI para verificar os resultados das regras em uma execução na qual uma condição de estágio ativou uma regra e produziu um resultado para o estágio.

- Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use o AWS CLI para executar o list-rule-executions comando de um pipeline chamado: *`MyPipeline`*

```
aws codepipeline list-rule-executions --pipeline-name MyFirstPipeline
```

O comando gera uma lista com todas as execuções de regras finalizadas vinculadas ao pipeline.

O exemplo a seguir mostra os dados retornados para um pipeline com uma condição de estágio em que a regra é nomeada*MyMonitorRule*.

```
{
    "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."
                }
```

```
            }
        }
```

Excluir um pipeline no CodePipeline.

Você sempre pode editar um pipeline para alterar sua funcionalidade, mas você também pode excluí-lo. Você pode usar o console do AWS CodePipeline ou o comando delete-pipeline na AWS CLI para excluir um pipeline.

Tópicos

- Excluir um pipeline (console)
- Excluir um pipeline (CLI)

Excluir um pipeline (console)

Para excluir um pipeline.

1. Faça login no Console de gerenciamento da AWS e abra o console do CodePipeline em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes e status de todos os pipelines associados à sua conta da AWS são exibidos.
2. Em Nome, selecione o nome do pipeline que você deseja excluir.
3. Na página de detalhes do pipeline, selecione Editar.
4. Na página Editar, selecione Excluir.
5. Digite **`delete`** no campo para confirmar e escolha Delete (Excluir).

> **Important**
>
> Esta ação não pode ser desfeita.

Excluir um pipeline (CLI)

Para usar a AWS CLI para excluir manualmente um pipeline, use o comando delete-pipeline.

> **Important**
>
> O ato de apagar um pipeline é irreversível. Não há caixa de diálogo de confirmação. Após a execução do comando, o pipeline será excluído, mas os recursos utilizados nele não serão. Isso facilita a criação de outro pipeline que usa esses recursos para automatizar o lançamento do software.

Para excluir um pipeline.

1. Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use a AWS CLI para executar o comando delete-pipeline, especificando o nome do pipeline que você deseja excluir. Por exemplo, para excluir um pipeline chamado *MyFirstPipeline*:

   ```
   aws codepipeline delete-pipeline --name MyFirstPipeline
   ```

   Esse comando não retorna nada.

2. Excluir recursos que não são mais necessários.

   > **Note**
   >
   > A exclusão de um pipeline não exclui os recursos usados no pipeline, como a aplicação do CodeDeploy ou do Elastic Beanstalk que você usou para implantar o código ou, se você criou o pipeline no console do CodePipeline, o bucket do Amazon S3 que o CodePipeline criou para armazenar os artefatos dos pipelines. Certifique-se de excluir os recursos que não são mais necessários para que você não seja cobrado por eles no futuro. Por exemplo, quando você usar o console para criar um pipeline pela primeira vez, o CodePipeline criará um bucket do Amazon S3 para armazenar todos os artefatos para todos os seus pipelines. Se você tiver excluído todos os seus pipelines, siga as etapas em Exclusão de um bucket.

Crie um pipeline CodePipeline que use recursos de outro AWS account

Pode ser que você queira criar um pipeline que use os recursos criados ou gerenciados por outra conta da AWS . Por exemplo, você pode querer usar uma conta com o pipeline e outra com os recursos do CodeDeploy .

> **Note**
>
> Depois de criar um pipeline com ações de várias contas, você deve configurar suas ações para que ainda possam acessar artefatos dentro das limitações de pipelines entre contas. As limitações a seguir se aplicam às ações entre contas:
>
> - Geralmente, uma ação só pode consumir um artefato se:
>   - A ação está na mesma conta que o pipeline OU
>   - O artefato foi criado na conta do pipeline para uma ação em outra conta OU
>   - O artefato foi produzido por uma ação anterior na mesma conta
>
>   Ou seja, você não pode transmitir um artefato de uma conta para outra se nenhuma delas for a conta do pipeline.
> - Cross-account as ações não são suportadas para os seguintes tipos de ação:
>   - Ações de compilações Jenkins

Neste exemplo, você deve criar uma chave AWS Key Management Service (AWS KMS) para usar, adicionar a chave ao pipeline e configurar políticas e funções da conta para permitir o acesso entre contas. Para uma chave AWS KMS, você pode usar o ID da chave, o ARN da chave ou o alias ARN.

> **Note**
>
> Os aliases são reconhecidos apenas na conta que criou a chave do KMS. Para ações entre contas, você só pode usar o ID da chave ou o ARN da chave para identificar a chave. Cross-account as ações envolvem o uso da função da outra conta (contaB), portanto, especificar o ID da chave usará a chave da outra conta (contaB).

Neste passo a passo e em seus exemplos, *AccountA* está a conta usada originalmente para criar o funil. Ele tem acesso ao bucket do Amazon S3 usado para armazenar artefatos do pipeline e à função de serviço usada pelo. AWS CodePipeline*AccountB*é a conta originalmente usada para criar o CodeDeploy aplicativo, o grupo de implantação e a função de serviço usados pelo CodeDeploy.

*AccountA*Para editar um pipeline para usar o CodeDeploy aplicativo criado por*AccountB*, é *AccountA* necessário:

- Solicite o ARN ou o ID da conta de *AccountB* (neste passo a passo, o *AccountB* ID é). *012ID_ACCOUNT_B*
- Crie ou use uma chave gerenciada pelo AWS KMS cliente na região para o funil e conceda permissões para usar essa chave na função de serviço (*CodePipeline_Service_Role*) *AccountB* e.
- Crie uma política de bucket do Amazon S3 que conceda *AccountB* acesso ao bucket do Amazon S3 (por exemplo,*codepipeline-us-east-2-1234567890*).
- Crie uma política que permita *AccountA* assumir uma função configurada por *AccountB* e anexar essa política à função de serviço (*CodePipeline_Service_Role*).
- Edite o pipeline para usar a AWS KMS chave gerenciada pelo cliente em vez da chave padrão.

Para *AccountB* permitir o acesso a seus recursos em um pipeline criado em*AccountA*, *AccountB* deve:

- Solicite o ARN ou o ID da conta de *AccountA* (neste passo a passo, o *AccountA* ID é). *012ID_ACCOUNT_A*
- Crie uma política aplicada à função de instância do Amazon EC2 configurada para permitir CodeDeploy o acesso ao bucket do Amazon S3 ()*codepipeline-us-east-2-1234567890*.
- Crie uma política aplicada à função de instância do Amazon EC2 configurada para permitir CodeDeploy o acesso à chave gerenciada pelo AWS KMS cliente usada para criptografar os artefatos do pipeline. *AccountA*
- Configure e anexe uma função do IAM (*CrossAccount_Role*) com uma política de relacionamento de confiança que permita que a função de CodePipeline serviço assuma *AccountA* a função.
- Crie uma política que permita o acesso aos recursos de implantação exigidos pelo pipeline e anexe-a*CrossAccount_Role*.

- Crie uma política que permita o acesso ao bucket do Amazon S3 (*codepipeline-us-east-2-1234567890*) e anexe-o a. *CrossAccount_Role*

Tópicos

- Pré-requisito: criar um AWS KMS chave de criptografia
- Etapa 1: Configurar as políticas e funções da conta
- Etapa 2: Editar o pipeline

Pré-requisito: criar um AWS KMS chave de criptografia

Customer-managed as chaves são específicas de uma região, assim como todas as AWS KMS chaves. Você deve criar sua AWS KMS chave gerenciada pelo cliente na mesma região em que o funil foi criado (por exemplo,`us-east-2`).

Para criar uma chave gerenciada pelo cliente no AWS KMS

1. Faça login no Console de gerenciamento da AWS com *AccountA* e abra o AWS KMS console.
2. À esquerda, escolha Chaves gerenciadas pelo cliente.
3. Escolha Criar chave. Em Configurar chave, deixe a opção padrão Simétrica selecionada e escolha Próximo.
4. Em Alias, insira um alias a ser usado para essa chave (por exemplo,*PipelineName-Key*). Ou então, forneça uma descrição e tags para essa chave e selecione Próximo.
5. Em Definir permissões administrativas da chave, escolha o(s) perfil(is) que devem atuar como administradores para essa chave e, em seguida, escolha Próximo.
6. Em Definir permissões de uso da chave, em Esta conta, selecione o nome da função de serviço para o pipeline (por exemplo, CodePipeline _Service_Role). Em Outras AWS contas, escolha Adicionar outra AWS conta. Insira o ID da conta *AccountB* para preencher o ARN e escolha Avançar.
7. Em Revisar e editar política de chaves, revise a política e selecione Concluir.
8. Na lista de chaves, selecione o alias da chave e copie o ARN (por exemplo, ***arn:aws:kms:us-east-2:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE***). Você precisará disso ao editar seu pipeline e configurar políticas.

Etapa 1: Configurar as políticas e funções da conta

Depois de criar a AWS KMS chave, você deve criar e anexar políticas que permitirão o acesso entre contas. Isso requer ações de ambos *AccountA* *AccountB* e.

Tópicos

- Configure políticas e funções na conta que criará o pipeline (Conta A)
- Configure políticas e funções na conta que possui o AWS recurso (Conta B)

Configure políticas e funções na conta que criará o pipeline (***Conta A***)

Para criar um pipeline que usa CodeDeploy recursos associados a outra AWS conta, é *AccountA* necessário configurar políticas para o bucket do Amazon S3 usado para armazenar artefatos e para a função de serviço para o qual. CodePipeline

Para criar uma política para o bucket do Amazon S3 que conceda acesso a AccountB (console)

1. Faça login no Console de gerenciamento da AWS com *AccountA* e abra o console do Amazon S3 em https://console.aws.amazon.com/s3/.
2. Na lista de buckets do Amazon S3, escolha o bucket do Amazon S3 em que os artefatos de seus pipelines serão armazenados. Esse bucket é chamado`codepipeline-`*region*`-`*1234567EXAMPLE*, onde *region* está a AWS região na qual você criou o pipeline e *1234567EXAMPLE* é um número aleatório de dez dígitos que garante que o nome do bucket seja exclusivo (por exemplo,). *codepipeline-us-east-2-1234567890*
3. Na página de detalhes do bucket do Amazon S3, escolha Propriedades.
4. No painel de propriedades, expanda Permissões e depois selecione Adicionar política de bucket.

> **Note**
>
> Se uma política já estiver anexada ao bucket do Amazon S3, escolha Editar política de bucket. Em seguida, você pode adicionar as instruções no exemplo a seguir para a política existente. Para adicionar uma nova política, escolha o link e siga as instruções no Gerador AWS de políticas. Para obter mais informações, consulte Visão geral das políticas do IAM.

5. Na janela Bucket Policy Editor, crie uma política que permita o *AccountB* acesso aos artefatos do pipeline e ofereça *AccountB* a capacidade de adicionar artefatos de saída se uma ação, como uma fonte personalizada ou uma ação de criação, os criar.
6. Selecione Salvar e depois feche o editor de políticas.
7. Escolha Salvar para salvar as permissões para o bucket do Amazon S3.

Para criar uma política para a função de serviço para CodePipeline (console)

1. Faça login no Console de gerenciamento da AWS com *AccountA* e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Perfis.
3. Na lista de funções, em Role Name (Nome da função), selecione o nome da função de serviço para o CodePipeline.
4. Na guia Permissions (Permissões), escolha Add inline policy (Adicionar política em linha).
5. Escolha a guia JSON e insira a política a seguir para permitir que você assuma *AccountB* a função. No exemplo a seguir, *012ID_ACCOUNT_B* é o ARN para*AccountB*:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": [
            "arn:aws:iam::111122223333:role/*"
        ]
    }
}
```

6. Selecione Revisar política.
7. Em Name (Nome), insira um nome para essa política. Selecione Criar política.

Configure políticas e funções na conta que possui o AWS recurso (*Conta B*)

Ao criar um aplicativo, uma implantação e um grupo de implantação no CodeDeploy, você também cria uma função de instância do Amazon EC2. (Essa função é criada caso você use o assistente de

apresentação de implantação de execução, mas também é possível criá-la manualmente.) Para que um funil criado em *AccountA* use CodeDeploy recursos criados em*AccountB*, você deve:

- Configurar uma política para o perfil de instância que permita o acesso ao bucket do Amazon S3 em que os artefatos do pipeline serão armazenados.
- Crie uma segunda função em *AccountB* configurado para acesso entre contas.

  Essa segunda função não deve apenas ter acesso ao bucket do Amazon S3*AccountA*, mas também deve conter uma política que permita o acesso aos CodeDeploy recursos e uma política de relacionamento de confiança que permita que a função de CodePipeline serviço assuma *AccountA* a função.

> **Note**
>
> Essas políticas são específicas para configurar CodeDeploy recursos a serem usados em um funil criado usando uma AWS conta diferente. Outros AWS recursos exigirão políticas específicas para seus requisitos de recursos.

Para criar uma política para a função de instância do Amazon EC2 configurada para CodeDeploy (console)

1. Faça login no Console de gerenciamento da AWS com *AccountB* e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação, escolha Perfis.
3. Na lista de funções, em Nome da função, escolha o nome da função de serviço usada como a função de instância do Amazon EC2 para o CodeDeploy aplicativo. Esse nome da função pode variar e mais de uma função de instância pode ser usada por um grupo de implantação. Para obter mais informações, consulte Criar um perfil de instância do IAM para suas instâncias do Amazon EC2.
4. Na guia Permissions (Permissões), escolha Add inline policy (Adicionar política em linha).
5. Escolha a guia JSON e insira a política a seguir para conceder acesso ao bucket do Amazon S3 usado pelo *AccountA* para armazenar artefatos para pipelines (neste exemplo,): *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. Selecione Revisar política.
7. Em Name (Nome), insira um nome para essa política. Selecione Criar política.
8. Crie uma segunda política para AWS KMS saber onde ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** está o ARN da chave gerenciada pelo cliente criada *AccountA* e configurada *AccountB* para permitir seu 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**
>
> Você deve usar o ID da *AccountA* conta desta política como parte do ARN do recurso para a AWS KMS chave, conforme mostrado aqui, ou a política não funcionará.

9. Selecione Revisar política.
10. Em Name (Nome), insira um nome para essa política. Selecione Criar política.

Agora, crie uma função do IAM para usar no acesso entre contas e configure-a para que a função CodePipeline de serviço *AccountA* possa assumir a função. Essa função deve conter políticas que permitam o acesso aos CodeDeploy recursos e ao bucket do Amazon S3 usados para armazenar artefatos. *AccountA*

Para configurar o perfil entre contas no IAM

1. Faça login no Console de gerenciamento da AWS com *AccountB* e abra o console do IAM em https://console.aws.amazon.com/iam.
2. No painel de navegação, escolha Perfis. Escolha Criar Perfil.
3. Em Selecionar tipo de entidade confiável, escolha Outra conta da AWS . Em Especificar contas que podem usar essa função, em ID da AWS conta, insira a ID da conta que criará o pipeline em CodePipeline (*AccountA*) e escolha Avançar: Permissões.

> **Important**
>
> Esta etapa cria a política de relacionamento de confiança entre *AccountB* *AccountA* e. No entanto, isso concede acesso de nível raiz à conta e CodePipeline recomenda reduzi-la à função CodePipeline de serviço em*AccountA*. Siga a etapa 16 para restringir permissões.

4. Em Anexar políticas de permissões AmazonS3ReadOnlyAccess, escolha e escolha Avançar: Tags.

> **Note**
>
> Esta não é a política que você utilizará. Você deve escolher uma política para concluir o assistente.

5. Escolha Próximo: revisar. Digite um nome para essa função em Nome da função (por exemplo,*CrossAccount_Role*). Você pode nomear esse perfil como quiser, desde que ele siga as convenções de nomenclatura do IAM. Considere atribuir uma nome à função que indique claramente o seu propósito. Selecione Criar função.
6. Na lista de funções, escolha a função que você acabou de criar (por exemplo,*CrossAccount_Role*) para abrir a página de resumo dessa função.
7. Na guia Permissions (Permissões), escolha Add inline policy (Adicionar política em linha).
8. Escolha a guia JSON e insira a política a seguir para permitir o acesso aos CodeDeploy recursos:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:GetApplicationRevision",
```

```
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource": "*"
    }
  ]
}
```

9. Selecione Revisar política.
10. Em Name (Nome), insira um nome para essa política. Selecione Criar política.
11. Na guia Permissions (Permissões), escolha Add inline policy (Adicionar política em linha).
12. Escolha a guia JSON e insira a política a seguir para permitir que essa função recupere artefatos de entrada e coloque artefatos de saída no bucket do Amazon S3 em: *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. Selecione Revisar política.
14. Em Name (Nome), insira um nome para essa política. Selecione Criar política.
15. Na guia Permissões, localize a lista de políticas AmazonS3ReadOnlyAccess em Nome da política e escolha o ícone de exclusão (X) ao lado da política. Quando solicitado, selecione Desanexar.
16. Selecione a guia Relação de confiança e escolha Editar política de confiança. Escolha a opção Adicionar uma entidade principal na coluna à esquerda. Para o tipo Principal, escolha Papéis

do IAM e, em seguida, forneça o ARN para a função CodePipeline de serviço em*AccountA*. Remova `arn:aws:iam::Account_A:root` da lista de Entidades principais da AWS e selecione Atualizar política.

Etapa 2: Editar o pipeline

Você não pode usar o CodePipeline console para criar ou editar um pipeline que usa recursos associados a outra AWS conta. No entanto, você pode usar o console para criar a estrutura geral do pipeline e, em seguida, usar o AWS CLI para editar o pipeline e adicionar esses recursos. Como alternativa, você pode usar a estrutura de um pipeline existente e adicionar manualmente os recursos.

Para adicionar os recursos associados a outro AWS conta (AWS CLI)

1. Em um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows), execute o comando get-pipeline no pipeline ao qual deseja adicionar recursos. Copie a saída do comando em um arquivo JSON. Por exemplo, para um pipeline chamado MyFirstPipeline, você digitaria algo assim:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   A saída é enviada para o arquivo *pipeline.json*.

2. Abra o arquivo JSON em qualquer editor de texto simples. Depois de `"type": "S3"` entrar no armazenamento de artefatos, adicione as informações de chave, ID e tipo do KMS EncryptionKey, onde *codepipeline-us-east-2-1234567890* está o nome do bucket do Amazon S3 usado para armazenar artefatos para o pipeline e ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** é o ARN da chave gerenciada pelo cliente que você acabou de criar:

   ```
   {
     "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. Adicione uma ação de implantação em um estágio para usar os CodeDeploy recursos associados*AccountB*, incluindo os `roleArn` valores da função entre contas que você criou (*CrossAccount_Role*).

   O exemplo a seguir mostra o JSON que adiciona uma ação de implantação chamada*ExternalDeploy*. Ele usa os CodeDeploy recursos criados *AccountB* em um estágio chamado*Staging*. No exemplo a seguir, o ARN para *AccountB* é*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 não é o JSON para todo o pipeline, apenas a estrutura para a ação em um estágio.

4. Você deve remover as linhas `metadata` do arquivo para que o comando update-pipeline possa usá-lo. Remova a seção da estrutura do pipeline no arquivo JSON (as linhas de `"metadata": { }` e os campos `"created"`, `"pipelineARN"` e `"updated"`).

   Por exemplo, remova as seguintes linhas da estrutura:

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

   Salve o arquivo.

5. Para aplicar suas alterações, execute o comando update-pipeline, especificando o pipeline do arquivo JSON, de modo semelhante ao seguinte:

   > **Important**
   >
   > Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

   Este comando retorna toda a estrutura do pipeline editado.

Para testar o pipeline que usa recursos associados a outro AWS account

1. Em um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows), execute o comando start-pipeline-execution, especificando o nome do pipeline, da seguinte maneira:

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

Para obter mais informações, consulte Iniciar um pipeline manualmente.

2. Faça login no Console de gerenciamento da AWS com *AccountA* e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes de todos os canais associados à sua AWS conta são exibidos.

3. Em Nome, selecione o nome do pipeline recém-editado. Isso abre uma visão detalhada do pipeline, incluindo o estado de cada ação em cada estágio do pipeline.
4. Assista o progresso através do pipeline. Aguarde uma mensagem de sucesso na ação que usa o recurso associado a outra AWS conta.

> **Note**
>
> Você receberá uma mensagem de erro se tentar visualizar os detalhes da ação enquanto estiver conectado com*AccountA*. Saia e, em seguida, entre com *AccountB* para ver os detalhes da implantação em CodeDeploy.

Migrar pipelines de sondagem para usar a detecção de alterações baseada em eventos

AWS CodePipeline oferece suporte à entrega contínua completa, de ponta a ponta, o que inclui iniciar seu pipeline sempre que houver uma alteração no código. Há duas maneiras de iniciar o pipeline após uma alteração no código: detecção de alterações baseada em eventos e sondagem. Recomendamos usar a detecção de alterações baseada em eventos para pipelines.

Use os procedimentos incluídos aqui para migrar (atualizar) seus pipelines de sondagem para o método de detecção de alterações baseada em eventos do seu pipeline.

O método recomendado de detecção de alterações baseado em eventos para tubulações é determinado pela fonte do pipeline, como. CodeCommit Nesse caso, por exemplo, o canal de pesquisa precisaria migrar para a detecção de alterações baseada em eventos com. EventBridge

Como migrar os pipelines de sondagem

Para migrar os pipelines de sondagem, determine seus pipelines de sondagem e, em seguida, determine o método recomendado de detecção de alterações baseada em eventos:

- Use as etapas em Visualizar os pipelines de sondagem em sua conta para determinar seus canais de sondagem.
- Na tabela, encontre o tipo de origem do pipeline e escolha o procedimento com a implementação que você deseja usar para migrar o pipeline de sondagem. Cada seção contém vários métodos de migração, como o uso da CLI ou do CloudFormation.

Como migrar pipelines para o método de detecção de alterações recomendado		
Origem do pipeline	Método recomendado de detecção baseada em eventos	Procedimentos de migração
AWS CodeCommit	EventBridge (recomendado).	Consulte Migre canais de votação com uma fonte CodeCommit.
Amazon S3	EventBridge e bucket habilitado para notificações de eventos (recomendado).	Consulte Migrar pipelines de sondagem com uma origem do S3 habilitada para eventos.
Amazon S3	EventBridge e uma AWS CloudTrail trilha.	Consulte Migre canais de votação com uma origem e uma trilha do S3 CloudTrail.
GitHub (via GitHub aplicativo)	Conexões (recomendadas)	Consulte Migre canais de pesquisa para uma ação de origem GitHub (por meio do aplicativo OAuth) para conexões.
GitHub (por meio do aplicativo OAuth)	Webhooks	Consulte Migre canais de pesquisa para uma ação de origem GitHub (por meio do aplicativo OAuth) para webhooks.

Important

Para atualizações de configuração de ações de pipeline aplicáveis, como pipelines com uma ação GitHub (por meio do aplicativo OAuth), você deve definir explicitamente o `PollForSourceChanges` parâmetro como `false` na configuração da ação Source para

impedir a pesquisa de um pipeline. Como resultado, é possível configurar erroneamente um pipeline com detecção de alterações e sondagem com base em eventos, por exemplo, configurando uma EventBridge regra e também omitindo o parâmetro. `PollForSourceChanges` Isso resulta em execuções de pipeline duplicadas, e o pipeline é contabilizado em relação ao limite no número total de pipelines de sondagem, que, por padrão é muito inferior aos pipelines com base em eventos. Para obter mais informações, consulte Cotas em AWS CodePipeline.

Visualizar os pipelines de sondagem em sua conta

Como primeira etapa, use um dos scripts a seguir para determinar quais pipelines em sua conta estão configurados para sondagem. Esses são os pipelines para migrar para a detecção de alterações baseada em eventos.

Visualizar pipelines de votação em sua conta (script)

Siga estas etapas para usar um script para determinar os pipelines em sua conta que estão usando sondagem.

1. Abra uma janela de terminal e execute uma das seguintes ações:

   - Execute o comando a seguir para criar um novo script chamado PollingPipelinesExtractor.sh.

     ```
     vi PollingPipelinesExtractor.sh
     ```

   - Para usar um script python, execute o comando a seguir para criar um novo script python chamado. PollingPipelinesExtractor.py

     ```
     vi PollingPipelinesExtractor.py
     ```

2. Copie e cole o código a seguir no PollingPipelinesExtractor script. Execute um destes procedimentos:

   - Copie e cole o código a seguir no 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 e cole o código a seguir no 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. Para cada região em que você tem pipelines, você deve executar o script para essa região. Para executar o script, faça o seguinte:

   - Execute o comando a seguir para executar o script chamado PollingPipelinesExtractor.sh. Neste exemplo, a região é us-west-2.

     ```
     ./PollingPipelinesExtractor.sh us-west-2
     ```

   - Para o script python, execute o comando a seguir para executar o script python chamado. PollingPipelinesExtractor.py Neste exemplo, a região é us-west-2.

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

   No exemplo de saída do script a seguir, a Região us-west-2 retornou uma lista de pipelines de sondagem e mostra o horário da última execução de cada pipeline.

```
 % ./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...%
```

Analise a saída do script e, para cada pipeline na lista, atualize a fonte de sondagem para o método recomendado de detecção de alterações baseada em eventos.

> **Note**
>
> Seus pipelines de sondagem são determinados pela configuração de ação do pipeline para o parâmetro `PollForSourceChanges`. Se a configuração de origem do pipeline tiver o `PollForSourceChanges` parâmetro omitido, o CodePipeline padrão será pesquisar alterações na fonte em seu repositório. Esse comportamento será o equivalente a ter `PollForSourceChanges` incluído e definido como true. Para obter mais informações, consulte os parâmetros de configuração da ação de origem do seu pipeline, como os parâmetros de configuração da ação de origem do Amazon S3 em Referência da ação de origem do Amazon S3.

Observe que esse script também gera um arquivo.csv contendo a lista de pipelines de sondagem em sua conta e salva o arquivo.csv na pasta de trabalho atual.

Migre canais de votação com uma fonte CodeCommit

Você pode migrar seu pipeline de pesquisa EventBridge para usá-lo na detecção de alterações em seu repositório de CodeCommit origem ou em seu bucket de origem do Amazon S3.

CodeCommit-- Para um pipeline com uma CodeCommit origem, modifique o pipeline para que a detecção de alterações seja automatizada EventBridge. Escolha um dos seguintes métodos para implementar a migração:

- Console: Migre canais de pesquisa (CodeCommit ou fonte do Amazon S3) (console)
- CLI: Migrar canais de votação (CodeCommit fonte) (CLI)
- CloudFormation: Migrar canais de votação (fonte) (CodeCommit CloudFormation modelo)

Migre canais de pesquisa (CodeCommit ou fonte do Amazon S3) (console)

Você pode usar o CodePipeline console para atualizar seu pipeline EventBridge para detectar alterações em seu repositório de CodeCommit origem ou em seu bucket de origem do Amazon S3.

> **Note**
>
> Quando você usa o console para editar um pipeline que tem um repositório de CodeCommit origem ou um bucket de origem do Amazon S3, a regra e a função do IAM são criadas para você. Se você usar o AWS CLI para editar o pipeline, você mesmo deverá criar a EventBridge regra e a função do IAM. Para obter mais informações, consulte CodeCommit ações de origem e EventBridge.

Siga essas etapas para editar um pipeline que conduz verificações periódicas. Se você deseja criar um pipeline, consulte Criar um pipeline, estágios e ações.

Para editar o estágio de origem do pipeline

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes de todos os canais associados à sua AWS conta são exibidos.
2. Em Nome, selecione o nome do pipeline que você deseja editar. Isso abrirá um visão detalhada do pipeline, incluindo o estado de cada uma das ações em cada estágio do pipeline.
3. Na página de detalhes do pipeline, selecione Editar.
4. Em Edit stage (Editar estágio), selecione o ícone de edição na ação de origem.
5. Expanda as opções de detecção de alterações e escolha Usar CloudWatch eventos para iniciar automaticamente meu pipeline quando ocorrer uma alteração (recomendado).

   Uma mensagem aparece mostrando a EventBridge regra a ser criada para esse pipeline. Selecione Atualizar.

   Se estiver atualizando um pipeline que tem uma origem do Amazon S3, você verá a mensagem a seguir. Selecione Atualizar.
6. Quando terminar de editar seu pipeline, selecione Salvar alterações do pipeline para voltar à página de resumo.

   Uma mensagem exibe o nome da EventBridge regra a ser criada para seu funil. Escolha Save and continue.
7. Para testar sua ação, libere uma alteração usando o AWS CLI para confirmar uma alteração na fonte especificada no estágio de origem do pipeline.

Migrar canais de votação (CodeCommit fonte) (CLI)

Siga estas etapas para editar um funil que está usando sondagem (verificações periódicas) para usar uma EventBridge regra para iniciar o funil. Se você deseja criar um pipeline, consulte Criar um pipeline, estágios e ações.

Para criar um pipeline orientado por eventos com CodeCommit, você edita o `PollForSourceChanges` parâmetro do seu funil e cria os seguintes recursos:

- EventBridge evento
- Função do IAM para permitir que esse evento inicie o pipeline

Para editar o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

1. Execute o get-pipeline comando para copiar a estrutura do pipeline em um arquivo JSON. Por exemplo, para um pipeline nomeado `MyFirstPipeline`, execute o seguinte comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando retorna nada, mas o arquivo que você criou deve aparecer no diretório onde você executou o comando.

2. Abra o arquivo JSON em qualquer editor de texto plano e altere o `PollForSourceChanges` parâmetro `false` para editar o estágio de origem, como mostrado no exemplo a seguir.

   Por que estou fazendo essa alteração? A alteração deste parâmetro para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em eventos.

```
"configuration": {
    "PollForSourceChanges": "false",
    "BranchName": "main",
    "RepositoryName": "MyTestRepo"
},
```

3. Se você estiver trabalhando com a estrutura do pipeline recuperada por meio do comando get-pipeline, remova as linhas `metadata` do arquivo JSON. Caso contrário, o comando update-pipeline não é capaz de utilizá-la. Remova as linhas `"metadata": { }`, `"created"`, `"pipelineARN"` e os campos `"updated"`.

   Por exemplo, remova as seguintes linhas da estrutura:

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

   Salve o arquivo.

4. Para aplicar suas alterações, execute o comando update-pipeline especificando o arquivo JSON do pipeline:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

   Este comando retorna toda a estrutura do pipeline editado.

> **Note**
>
> O comando update-pipeline interrompe o pipeline. Se uma revisão estiver sendo executada pelo pipeline quando você executar o comando update-pipeline, essa execução será interrompida. Você deve iniciar manualmente o pipeline para executar a

revisão através do pipeline atualizado. Use o comando **`start-pipeline-execution`** para iniciar manualmente o pipeline.

Para criar uma EventBridge regra com CodeCommit como origem do evento e CodePipeline como destino

1. Adicione permissões para usar EventBridge CodePipeline para invocar a regra. Para obter mais informações, consulte Usando políticas baseadas em recursos para a Amazon. EventBridge

   a. Use o exemplo a seguir para criar a política de confiança que EventBridge permite assumir a função de serviço. Nomeie a política de confiança `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Use o comando a seguir para criar a função `Role-for-MyRule` e anexar a política de confiança.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Crie o JSON de política de permissões, conforme mostrado neste exemplo, para o pipeline denominado `MyFirstPipeline`. Nomeie a politica de permissões `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Use o comando a seguir para anexar a política de permissões `CodePipeline-Permissions-Policy-for-EB` à função `Role-for-MyRule`.

   Por que estou fazendo essa alteração? Adicionar essa política à função cria permissões 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. Use o comando put-rule e inclua os parâmetros `--name`, `--event-pattern` e `--role-arn`.

   Por que estou fazendo essa alteração? Esse comando permite que o CloudFormation crie o evento.

   O comando de exemplo a seguir cria uma regra chamada `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 adicionar CodePipeline como destino, chame o put-targets comando e inclua os seguintes parâmetros:

   - O parâmetro `--rule` é usado com `rule_name` criado por meio de put-rule.
   - O parâmetro `--targets` é usado com o `Id` da lista do destino na lista de destinos e o `ARN` do pipeline de destino.

   O exemplo de comando a seguir especifica que, para a regra chamada `MyCodeCommitRepoRule`, o `Id` do destino é composto do número um, indicando que, em uma lista de destinos para a regra, esse é o destino 1. O exemplo de comando também especifica um exemplo `ARN` para o pipeline. O pipeline é iniciado quando uma alteração é feita no repositório.

   ```
   aws events put-targets --rule MyCodeCommitRepoRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

4. (Opcional) Para configurar um transformador de entrada com substituições de origem para um ID de imagem específico, use o JSON a seguir no comando da CLI. O exemplo a seguir configura uma substituição na qual:

   - O `actionName`, `Source` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O `revisionType`, `COMMIT_ID` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O`revisionValue`, **<** *`revisionValue`* **>** neste exemplo, é derivado da variável de evento de origem.

   ```
   {
       "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 canais de votação (fonte) (CodeCommit CloudFormation modelo)

Para criar um pipeline orientado por eventos com AWS CodeCommit, você edita o `PollForSourceChanges` parâmetro do seu funil e adiciona os seguintes recursos ao seu modelo:

- Uma EventBridge regra
- Uma função do IAM para sua EventBridge regra

Se você usa CloudFormation para criar e gerenciar seus funis, seu modelo inclui conteúdo como o seguinte.

> **Note**
>
> A propriedade `Configuration` no estágio de origem denominado `PollForSourceChanges`. Se essa propriedade não estiver incluída em seu modelo, `PollForSourceChanges` será definido como `true` por padrão.

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 atualizar seu funil CloudFormation modelo e EventBridge regra de criação

1. No modelo, abaixo`Resources`, use o `AWS::IAM::Role` CloudFormation recurso para configurar a função do IAM que permite que seu evento inicie seu pipeline. Essa entrada cria uma função que utiliza duas políticas:

   - A primeira política permite que a função seja assumida.
   - A segunda política fornece permissões para iniciar o pipeline.

   Por que estou fazendo essa alteração? Adicionar o `AWS::IAM::Role` recurso permite CloudFormation criar permissões para EventBridge. Esse recurso é adicionado à sua CloudFormation pilha.

   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. No modelo, em`Resources`, use o `AWS::Events::Rule` CloudFormation recurso para adicionar uma EventBridge regra. Esse padrão de evento cria um evento que monitora as alterações por push no seu repositório. Quando EventBridge detecta uma alteração no estado do repositório, a regra é invocada `StartPipelineExecution` em seu pipeline de destino.

   Por que estou fazendo essa alteração? Adicionar o `AWS::Events::Rule` recurso permite CloudFormation criar o evento. Esse recurso é adicionado à sua CloudFormation pilha.

   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 um transformador de entrada com substituições de origem para um ID de imagem específico, use o trecho YAML a seguir. O exemplo a seguir configura uma substituição na qual:

   - O `actionName`, `Source` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O `revisionType`, `COMMIT_ID` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O`revisionValue`, **<** *`revisionValue`* **>** neste exemplo, é derivado da variável de evento de origem.
   - As variáveis de saída para `BranchName` e `Value` são especificadas.

```
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
    sourceRevisions:
      actionName: Source
      revisionType: COMMIT_ID
      revisionValue: <revisionValue>
    variables:
    - name: BranchName
      value: value
```

4. Salve o modelo atualizado em seu computador local e abra o CloudFormation console.
5. Selecione sua pilha e clique em Create Change Set for Current Stack (Criar conjunto de alterações para a pilha atual).
6. Carregue o modelo e visualize as alterações listadas no CloudFormation. Essas são as alterações a serem feitas na pilha. Seus novos recursos devem ser exibidos na lista.
7. Clique em Executar.

Para editar o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Em muitos casos, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro ao criar um pipeline. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

- No modelo, altere `PollForSourceChanges` para `false`. Se você não incluir `PollForSourceChanges` na sua definição de pipeline, adicione-o e configure para `false`.

  Por que estou fazendo essa alteração? A alteração deste parâmetro para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em 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

Quando você cria esses recursos com CloudFormation, seu funil é acionado quando os arquivos em seu repositório são criados ou atualizados. Aqui está o trecho do modelo final:

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
                            }
                        ]
                    },

...
```

Migrar pipelines de sondagem com uma origem do S3 habilitada para eventos

Para um pipeline com uma origem do Amazon S3, modifique o pipeline para que a detecção de alterações seja automatizada por meio de EventBridge e com um bucket de origem habilitado para notificações de eventos. Esse é o método recomendado se você estiver usando a CLI ou CloudFormation para migrar seu pipeline.

> **Note**
>
> Isso inclui o uso de um bucket habilitado para notificações de eventos, em que você não precisa criar uma CloudTrail trilha separada. Se você estiver usando o console, uma regra de evento e uma CloudTrail trilha serão configuradas para você. Para obter essas etapas, consulte Migre canais de votação com uma origem e uma trilha do S3 CloudTrail.

- CLI: Migre canais de pesquisa com uma origem e CloudTrail trilha (CLI) do S3
- CloudFormation: Migre canais de votação com uma origem e uma trilha do S3 (CloudTrail CloudFormation modelo)

Migrar pipelines de sondagem com uma origem do S3 habilitada para eventos (CLI)

Siga estas etapas para editar um funil que está usando pesquisas (verificações periódicas) para usar um evento em EventBridge vez disso. Se você deseja criar um pipeline, consulte Criar um pipeline, estágios e ações.

Para criar um pipeline baseado em eventos com o Amazon S3, edite o parâmetro `PollForSourceChanges` de seu pipeline e crie os seguintes recursos:

- EventBridge regra do evento
- Função do IAM para permitir que o EventBridge evento inicie seu funil

Criar uma EventBridge regra com o Amazon S3 como origem e CodePipeline destino do evento e aplicar a política de permissões

1. Conceda permissões EventBridge para usar CodePipeline para invocar a regra. Para obter mais informações, consulte Usando políticas baseadas em recursos para a Amazon. EventBridge

a. Use o exemplo a seguir para criar a política de confiança que permita ao EventBridge assumir a função de serviço. Chame-o de `trustpolicyforEB.json`.

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "events.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

b. Use o comando a seguir para criar a função `Role-for-MyRule` e anexar a política de confiança.

   Por que estou fazendo essa alteração? A adição dessa política de confiança à função cria permissões para EventBridge.

   ```
   aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
    file://trustpolicyforEB.json
   ```

c. Crie o JSON da política de permissões, conforme mostrado aqui para o pipeline chamado `MyFirstPipeline`. Nomeie a politica de permissões `permissionspolicyforEB.json`.

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "codepipeline:StartPipelineExecution"
               ],
   ```

```
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Use o comando a seguir para anexar a nova política de permissões `CodePipeline-Permissions-Policy-for-EB` à função `Role-for-MyRule` criada.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Use o comando put-rule e inclua os parâmetros `--name`, `--event-pattern` e `--role-arn`.

   O exemplo de comando a seguir cria uma regra chamada `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 adicionar CodePipeline como destino, chame o put-targets comando e inclua `--rule` `--targets` os parâmetros e.

   O comando a seguir especifica que, para a regra denominada `EnabledS3SourceRule`, o `Id` do destino é composto do número um, indicando que, em uma lista de destinos para a regra, esse é o destino 1. O comando também especifica um ARN de exemplo para o pipeline. O pipeline é iniciado quando uma alteração é feita no repositório.

```
aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-
AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

Para editar o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar

a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

1. Execute o get-pipeline comando para copiar a estrutura do pipeline em um arquivo JSON. Por exemplo, para um pipeline nomeado `MyFirstPipeline`, execute o seguinte comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando retorna nada, mas o arquivo que você criou deve aparecer no diretório onde você executou o comando.

2. Abra o arquivo JSON em qualquer editor de texto plano e, para editar o estágio de origem, altere o parâmetro `PollForSourceChanges` de um bucket denominado `amzn-s3-demo-source-bucket` para `false`, conforme mostrado neste exemplo.

   Por que estou fazendo essa alteração? A configuração deste parâmetro para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em eventos.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Se você estiver trabalhando com a estrutura do pipeline recuperada por meio do comando get-pipeline, é necessário remover as linhas `metadata` do arquivo JSON. Caso contrário, o comando update-pipeline não é capaz de utilizá-la. Remova as linhas `"metadata": { }`, `"created"`, `"pipelineARN"` e os campos `"updated"`.

   Por exemplo, remova as seguintes linhas da estrutura:

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

```
},
```

Salve o arquivo.

4. Para aplicar suas alterações, execute o comando update-pipeline especificando o arquivo JSON do pipeline:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

Este comando retorna toda a estrutura do pipeline editado.

> **Note**
>
> O comando update-pipeline interrompe o pipeline. Se uma revisão estiver sendo executada pelo pipeline quando você executar o comando update-pipeline, essa execução será interrompida. Você deve iniciar manualmente o pipeline para executar a revisão através do pipeline atualizado. Use o comando start-pipeline-execution para iniciar manualmente o pipeline.

Migre canais de votação com uma fonte S3 habilitada para eventos (CloudFormation modelo)

Esse procedimento refere-se a um pipeline em que o bucket de origem tem eventos habilitados.

Use estas etapas para alterar o pipeline com uma origem do Amazon S3 da sondagem para a detecção de alterações baseada em eventos.

Para criar um pipeline baseado em eventos com o Amazon S3, edite o parâmetro `PollForSourceChanges` de seu pipeline e adicione os seguintes recursos ao modelo:

- EventBridge regra e função do IAM para permitir que esse evento inicie seu pipeline.

Se você usa CloudFormation para criar e gerenciar seus funis, seu modelo inclui conteúdo como o seguinte.

> **Note**
>
> A propriedade `Configuration` no estágio de origem denominado `PollForSourceChanges`. Se o seu modelo não inclui essa propriedade, `PollForSourceChanges` é definido como `true` por padrão.

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
                            }
                        ]
                    },


...
```

Criar uma EventBridge regra com o Amazon S3 como origem e CodePipeline destino do evento e aplicar a política de permissões

1. No modelo, abaixo`Resources`, use o `AWS::IAM::Role` CloudFormation recurso para configurar a função do IAM que permite que seu evento inicie seu pipeline. Essa entrada cria uma função que utiliza duas políticas:

   - A primeira política permite que a função seja assumida.
   - A segunda política fornece permissões para iniciar o pipeline.

   Por que estou fazendo essa alteração? Adicionar `AWS::IAM::Role` recursos permite CloudFormation criar permissões para EventBridge. Esse recurso é adicionado à sua CloudFormation pilha.

   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. Use o `AWS::Events::Rule` CloudFormation recurso para adicionar uma EventBridge regra. Esse padrão de evento cria um evento que monitora a criação ou exclusão de objetos no bucket de origem do Amazon S3. Além disso, inclua um destino de seu pipeline. Quando um objeto é criado, essa regra é invoca `StartPipelineExecution` em seu pipeline de destino.

   Por que estou fazendo essa alteração? Adicionar o `AWS::Events::Rule` recurso permite CloudFormation criar o evento. Esse recurso é adicionado à sua CloudFormation pilha.

   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. Salve o modelo atualizado no computador local e abra o console do CloudFormation .
4. Selecione sua pilha e clique em Create Change Set for Current Stack (Criar conjunto de alterações para a pilha atual).
5. Carregue o modelo atualizado e, em seguida, visualize as alterações listadas no CloudFormation. Essas são as alterações que serão feitas na pilha. Seus novos recursos devem ser exibidos na lista.
6. Clique em Executar.

Para editar o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

- No modelo, altere `PollForSourceChanges` para `false`. Se você não incluir `PollForSourceChanges` na sua definição de pipeline, adicione-o e configure para `false`.

  Por que estou fazendo essa alteração? A alteração de `PollForSourceChanges` para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em 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

Quando você usa CloudFormation para criar esses recursos, seu funil é acionado quando os arquivos em seu repositório são criados ou atualizados.

> **Note**
>
> Não pare por aqui. Embora seu pipeline tenha sido criado, você deve criar um segundo CloudFormation modelo para seu pipeline do Amazon S3. Se você não criar o segundo modelo, o pipeline não apresentará funcionalidades de detecção de alterações.

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 canais de votação com uma origem e uma trilha do S3 CloudTrail

Para um pipeline com uma origem do Amazon S3, modifique o pipeline para que a detecção de alterações seja automatizada EventBridge. Escolha um dos seguintes métodos para implementar a migração:

- Console: Migre canais de pesquisa (CodeCommit ou fonte do Amazon S3) (console)
- CLI: Migre canais de pesquisa com uma origem e CloudTrail trilha (CLI) do S3
- CloudFormation: Migre canais de votação com uma origem e uma trilha do S3 (CloudTrail CloudFormation modelo)

Migre canais de pesquisa com uma origem e CloudTrail trilha (CLI) do S3

Siga estas etapas para editar um funil que está usando pesquisas (verificações periódicas) para usar um evento em EventBridge vez disso. Se você deseja criar um pipeline, consulte Criar um pipeline, estágios e ações.

Para criar um pipeline baseado em eventos com o Amazon S3, edite o parâmetro `PollForSourceChanges` de seu pipeline e crie os seguintes recursos:

- AWS CloudTrail política de trilha, bucket e bucket que o Amazon S3 pode usar para registrar os eventos.
- EventBridge evento
- Função do IAM para permitir que o EventBridge evento inicie seu funil

Para criar um AWS CloudTrail rastrear e ativar o registro

Para usar o AWS CLI para criar uma trilha, chame o create-trail comando, especificando:

- O nome da trilha.
- O bucket ao qual você já aplicou a política de bucket do AWS CloudTrail.

Para obter mais informações, consulte Criação de uma trilha com a interface de linha de AWS comando.

1. Use o comando create-trail e inclua os parâmetros `--name` e `--s3-bucket-name`.

   Por que estou fazendo essa alteração? Isso cria a trilha do CloudTrail necessária para o bucket de origem do S3.

   O comando a seguir usa `--name` e `--s3-bucket-name` para criar uma trilha denominada `my-trail` e um bucket chamado de `amzn-s3-demo-source-bucket`.

   ```
   aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-source-bucket
   ```

2. Use o comando start-logging e inclua o parâmetro `--name`.

   Por que estou fazendo essa alteração? Esse comando inicia o CloudTrail registro do seu bucket de origem e envia eventos para EventBridge o.

   Exemplo:

   O comando a seguir utiliza `--name` para iniciar o registro em log em uma trilha denominada `my-trail`.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Use o comando put-event-selectors e inclua os parâmetros `--trail-name` e `--event-selectors`. Use seletores de eventos para especificar que você deseja que sua trilha registre eventos de dados para seu bucket de origem e envie os eventos para a EventBridge regra.

   Por que estou fazendo essa alteração? Esse comando filtra eventos.

   Exemplo:

O comando a seguir utiliza `--trail-name` e `--event-selectors` para especificar eventos de dados para um bucket de origem e prefixo denominado `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"] }] }]'
```

Criar uma EventBridge regra com o Amazon S3 como origem e CodePipeline destino do evento e aplicar a política de permissões

1. Conceda permissões EventBridge para usar CodePipeline para invocar a regra. Para obter mais informações, consulte Usando políticas baseadas em recursos para a Amazon. EventBridge

   a. Use o exemplo a seguir para criar a política de confiança que permita assumir EventBridge a função de serviço. Chame-o de `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Use o comando a seguir para criar a função `Role-for-MyRule` e anexar a política de confiança.

      Por que estou fazendo essa alteração? Adicionar essa política de confiança à função cria permissões para EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Crie o JSON da política de permissões, conforme mostrado aqui para o pipeline chamado `MyFirstPipeline`. Nomeie a politica de permissões `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Use o comando a seguir para anexar a nova política de permissões `CodePipeline-Permissions-Policy-for-EB` à função `Role-for-MyRule` criada.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Use o comando put-rule e inclua os parâmetros `--name`, `--event-pattern` e `--role-arn`.

O exemplo de comando a seguir cria uma regra chamada `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 adicionar CodePipeline como destino, chame o put-targets comando e inclua `--rule` `--targets` os parâmetros e.

   O comando a seguir especifica que, para a regra denominada `MyS3SourceRule`, o `Id` do destino é composto do número um, indicando que, em uma lista de destinos para a regra, esse é o destino 1. O comando também especifica um ARN de exemplo para o pipeline. O pipeline é iniciado quando uma alteração é feita no repositório.

   ```
   aws events put-targets --rule MyS3SourceRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

4. (Opcional) Para configurar um transformador de entrada com substituições de origem para um ID de imagem específico, use o JSON a seguir no comando da CLI. O exemplo a seguir configura uma substituição na qual:

   - O `actionName`, `Source` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O `revisionType`, `S3_OBJECT_VERSION_ID` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O`revisionValue`, **<** *`revisionValue`* **>** neste exemplo, é derivado da variável de evento de origem.

   ```
   {
       "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 o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

1. Execute o get-pipeline comando para copiar a estrutura do pipeline em um arquivo JSON. Por exemplo, para um pipeline nomeado `MyFirstPipeline`, execute o seguinte comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando retorna nada, mas o arquivo que você criou deve aparecer no diretório onde você executou o comando.

2. Abra o arquivo JSON em qualquer editor de texto plano e, para editar o estágio de origem, altere o parâmetro `PollForSourceChanges` de um bucket denominado `amzn-s3-demo-source-bucket` para `false`, conforme mostrado neste exemplo.

   Por que estou fazendo essa alteração? A configuração deste parâmetro para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em eventos.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Se você estiver trabalhando com a estrutura do pipeline recuperada por meio do comando get-pipeline, é necessário remover as linhas `metadata` do arquivo JSON. Caso contrário, o comando update-pipeline não é capaz de utilizá-la. Remova as linhas `"metadata": { }`, `"created"`, `"pipelineARN"` e os campos `"updated"`.

   Por exemplo, remova as seguintes linhas da estrutura:

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

   Salve o arquivo.

4. Para aplicar suas alterações, execute o comando update-pipeline especificando o arquivo JSON do pipeline:

   > **Important**
   >
   > Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

   Este comando retorna toda a estrutura do pipeline editado.

   > **Note**
   >
   > O comando update-pipeline interrompe o pipeline. Se uma revisão estiver sendo executada pelo pipeline quando você executar o comando update-pipeline, essa execução será interrompida. Você deve iniciar manualmente o pipeline para executar a revisão através do pipeline atualizado. Use o comando start-pipeline-execution para iniciar manualmente o pipeline.

Migre canais de votação com uma origem e uma trilha do S3 (CloudTrail CloudFormation modelo)

Use estas etapas para alterar o pipeline com uma origem do Amazon S3 da sondagem para a detecção de alterações baseada em eventos.

Para criar um pipeline baseado em eventos com o Amazon S3, edite o parâmetro `PollForSourceChanges` de seu pipeline e adicione os seguintes recursos ao modelo:

- EventBridge exige que todos os eventos do Amazon S3 sejam registrados. Você deve criar uma trilha, um bucket e uma política de bucket do AWS CloudTrail que o Amazon S3 possa usar para registrar em log os eventos que ocorrem. Para obter mais informações, consulte Registrar em log os eventos de dados para trilhas e Registrar em log eventos de gerenciamento para trilhas.
- EventBridge regra e função do IAM para permitir que esse evento inicie seu pipeline.

Se você usa CloudFormation para criar e gerenciar seus funis, seu modelo inclui conteúdo como o seguinte.

> **Note**
>
> A propriedade `Configuration` no estágio de origem denominado `PollForSourceChanges`. Se o seu modelo não inclui essa propriedade, `PollForSourceChanges` é definido como `true` por padrão.

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
                        }
                    ]
                },


...
```

Criar uma EventBridge regra com o Amazon S3 como origem e CodePipeline destino do evento e aplicar a política de permissões

1. No modelo, abaixo`Resources`, use o `AWS::IAM::Role` CloudFormation recurso para configurar a função do IAM que permite que seu evento inicie seu pipeline. Essa entrada cria uma função que utiliza duas políticas:

   - A primeira política permite que a função seja assumida.
   - A segunda política fornece permissões para iniciar o pipeline.

   Por que estou fazendo essa alteração? Adicionar `AWS::IAM::Role` recursos permite CloudFormation criar permissões para EventBridge. Esse recurso é adicionado à sua CloudFormation pilha.

   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. Use o `AWS::Events::Rule` CloudFormation recurso para adicionar uma EventBridge regra. Esse padrão de evento cria um evento que monitora `CopyObject`, `PutObject` e `CompleteMultipartUpload` no bucket de origem do Amazon S3. Além disso, inclua um destino de seu pipeline. Quando `CopyObject`, `PutObject` ou `CompleteMultipartUpload` ocorrer, essa regra invoca `StartPipelineExecution` em seu pipeline de destino.

   Por que estou fazendo essa alteração? Adicionar o `AWS::Events::Rule` recurso permite CloudFormation criar o evento. Esse recurso é adicionado à sua CloudFormation pilha.

   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. Adicione este trecho ao primeiro modelo para permitir a funcionalidade de pilha 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 um transformador de entrada com substituições de origem para um ID de imagem específico, use o trecho YAML a seguir. O exemplo a seguir configura uma substituição na qual:

   - O `actionName`, `Source` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O `revisionType`, `S3_OBJECT_VERSION_ID` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.

- `OrevisionValue`, < *revisionValue* > neste exemplo, é derivado da variável de evento de origem.

```
---
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. Salve o modelo atualizado em seu computador local e abra o CloudFormation console.
6. Selecione sua pilha e clique em Create Change Set for Current Stack (Criar conjunto de alterações para a pilha atual).
7. Carregue o modelo atualizado e, em seguida, visualize as alterações listadas no CloudFormation. Essas são as alterações que serão feitas na pilha. Seus novos recursos devem ser exibidos na lista.
8. Clique em Executar.

Para editar o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

- No modelo, altere `PollForSourceChanges` para `false`. Se você não incluir `PollForSourceChanges` na sua definição de pipeline, adicione-o e configure para `false`.

  Por que estou fazendo essa alteração? A alteração de `PollForSourceChanges` para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em 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 criar um segundo modelo para os recursos do seu pipeline Amazon S3 CloudTrail

- Em um modelo separado, em`Resources`, use os `AWS::CloudTrail::Trail` CloudFormation recursos `AWS::S3::BucketAWS::S3::BucketPolicy`, e para fornecer uma definição simples de bucket e uma trilha para CloudTrail.

  Por que estou fazendo essa alteração? Dado o limite atual de cinco trilhas por conta, a CloudTrail trilha deve ser criada e gerenciada separadamente. (Consulte Limites em AWS CloudTrail.) No entanto, você pode incluir vários buckets do Amazon S3 em uma única trilha, para que possa criar a trilha uma vez e adicionar buckets do Amazon S3 para outros pipelines, conforme for necessário. Cole o seguinte no arquivo do segundo modelo de exemplo.

  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

Quando você usa CloudFormation para criar esses recursos, seu funil é acionado quando os arquivos em seu repositório são criados ou atualizados.

> **Note**
>
> Não pare por aqui. Embora seu pipeline tenha sido criado, você deve criar um segundo CloudFormation modelo para seu pipeline do Amazon S3. Se você não criar o segundo modelo, o pipeline não apresentará funcionalidades de detecção de alterações.

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 canais de pesquisa para uma ação de origem GitHub (por meio do aplicativo OAuth) para conexões

Você pode migrar uma ação de origem GitHub (por meio do aplicativo OAuth) para usar conexões para seu repositório externo. Esse é o método de detecção de alterações recomendado para pipelines com uma ação de origem GitHub (por meio do aplicativo OAuth).

Para um pipeline com uma ação de origem GitHub (via aplicativo OAuth), recomendamos modificar o pipeline para usar uma ação GitHub (via GitHub aplicativo) para que a detecção de alterações seja automatizada. AWS CodeConnections Para obter mais informações sobre como trabalhar com as conexões, consulte GitHub conexões.

Crie uma conexão com GitHub (console)

Você pode usar o console para criar uma conexão com GitHub o.

Etapa 1: Substitua sua ação GitHub (por meio do aplicativo OAuth)

Use a página de edição do pipeline para substituir sua ação GitHub (via aplicativo OAuth) por uma ação GitHub (via GitHub aplicativo).

Para substituir sua ação GitHub (por meio do aplicativo OAuth)

1. Faça login no CodePipeline console.
2. Selecione o pipeline e escolha Editar. Selecione Editar estágio no estágio de origem. É exibida uma mensagem recomendando que você atualize a ação.

3. Em Provedor de ações, escolha GitHub (via GitHub aplicativo).
4. Execute um destes procedimentos:

   - Em Conexão, se você ainda não tiver criado uma conexão com seu provedor, escolha Conectar GitHub a. Prossiga para a Etapa 2: Crie uma conexão com GitHub.
   - Em Conexão, se você já tiver criado uma conexão com seu provedor, escolha a conexão. Vá para a Etapa 3: Salvar a ação de origem para sua conexão.

Etapa 2: criar uma conexão com GitHub

Depois de escolher criar a conexão, a GitHub página Conectar a será exibida.

Para criar uma conexão com GitHub

1. Nas configurações de GitHub conexão, o nome da conexão é mostrado em Nome da conexão.

   Em GitHub Aplicativos, escolha uma instalação de aplicativo ou escolha Instalar um novo aplicativo para criar um.

   > **Note**
   >
   > Você instala uma aplicação para todas as suas conexões com um provedor específico. Se você já instalou o GitHub aplicativo, escolha-o e pule esta etapa.

2. Se a página de autorização for GitHub exibida, faça login com suas credenciais e escolha continuar.
3. Na página de instalação do aplicativo, uma mensagem mostra que o AWS CodeStar aplicativo está tentando se conectar à sua GitHub conta.

   > **Note**
   >
   > Você só instala o aplicativo uma vez para cada GitHub conta. Se você instalou a aplicação anteriormente, poderá escolher Configure (Configurar) para prosseguir para uma página de modificação para a instalação da aplicação ou usar o botão Back (Voltar) para retornar ao console.

4. Na página Instalar AWS CodeStar, escolha Instalar.

5. Na GitHub página Conectar a, o ID de conexão da sua nova instalação é exibido. Selecione Conectar.

Etapa 3: Salve sua ação GitHub de origem

Conclua as atualizações na página Editar ação para salvar a nova ação de origem.

Para salvar sua ação GitHub de origem

1. Em Nome do repositório, escolha o nome do repositório de terceiros. Em Ramificação, insira a ramificação onde deseja que o pipeline detecte alterações de origem.

   > **Note**
   >
   > No Repositório, digite `owner-name/repository-name` conforme mostrado neste exemplo:
   >
   > ```
   > my-account/my-repository
   > ```

2. Em Formato de artefato de saída, escolha o formato dos seus artefatos.

   - Para armazenar artefatos de saída da GitHub ação usando o método padrão, escolha CodePipeline default. A ação acessa os arquivos do GitHub repositório e armazena os artefatos em um arquivo ZIP no armazenamento de artefatos do pipeline.
   - Para armazenar um arquivo JSON que contém uma referência de URL ao repositório para que as ações downstream possam executar comandos Git diretamente, escolha Full clone (Clone completo). Essa opção só pode ser usada por ações CodeBuild downstream.

     Se você escolher essa opção, precisará atualizar as permissões para sua função de serviço de CodeBuild projeto, conforme mostrado emAdicione CodeBuild GitClone permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com. Para um tutorial que mostra como usar a opção Clone completo, consulte Tutorial: use o clone completo com uma fonte de GitHub pipeline.

3. Em Artefatos de saída, é necessário manter o nome do artefato de saída para essa ação, como `SourceArtifact`. Escolha Concluído para fechar a página Editar ação.
4. Escolha Concluído para fechar a página de edição do estágio. Escolha Salvar para fechar a página de edição do pipeline.

Crie uma conexão com GitHub (CLI)

Você pode usar o AWS Command Line Interface (AWS CLI) para criar uma conexão com GitHub o.

Para fazer isso, use o comando create-connection.

> **Important**
>
> Uma conexão criada por meio do AWS CLI ou AWS CloudFormation está no `PENDING` status por padrão. Depois de criar uma conexão com a CLI ou CloudFormation, use o console para editar a conexão e definir seu status`AVAILABLE`.

Para criar uma conexão com GitHub

1. Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows). Use o AWS CLI para executar o create-connection comando, especificando o `--provider-type` e `--connection-name` para sua conexão. Neste exemplo, o nome do provedor de terceiros é `GitHub` e o nome da conexão especificada é `MyConnection`.

   ```
   aws codeconnections create-connection --provider-type GitHub --connection-name
    MyConnection
   ```

   Se tiver êxito, esse comando gerará as informações do ARN de conexão semelhantes às seguintes.

   ```
   {
       "ConnectionArn": "arn:aws:codeconnections:us-west-2:account_id:connection/
   aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
   }
   ```

2. Use o console para concluir a conexão.

Migre canais de pesquisa para uma ação de origem GitHub (por meio do aplicativo OAuth) para webhooks

Você pode migrar seu pipeline para usar webhooks para detectar alterações no seu GitHub repositório de origem. Essa migração para webhooks é somente para a ação GitHub (via aplicativo OAuth).

- Console: Migre canais de pesquisa para webhooks (por meio do aplicativo OAuthGitHub) (ações de origem) (console)
- CLI: Migre canais de pesquisa para webhooks (por meio do aplicativo OAuthGitHub) (ações de origem) (CLI)
- CloudFormation: Atualize pipelines para eventos push GitHub (por meio do aplicativo OAuth) (ações de origem) (CloudFormation modelo)

> **Important**
>
> Ao criar CodePipeline webhooks, não use suas próprias credenciais nem reutilize o mesmo token secreto em vários webhooks. Para garantir um nível ideal de segurança, gere um token secreto exclusivo para cada webhook que você criar. O token secreto é uma string arbitrária que você fornece, GitHub usada para calcular e assinar as cargas do webhook enviadas para CodePipeline, a fim de proteger a integridade e a autenticidade das cargas do webhook. Usar as próprias credenciais ou reutilizar o mesmo token em vários webhooks pode levar a vulnerabilidades de segurança.

Migre canais de pesquisa para webhooks (por meio do aplicativo OAuthGitHub) (ações de origem) (console)

Para a ação de origem GitHub (via aplicativo OAuth), você pode usar o CodePipeline console para atualizar seu pipeline e usar webhooks para detectar alterações em seu repositório de origem. GitHub

Siga estas etapas para editar um funil que esteja usando pesquisas (verificações periódicas) como alternativa. EventBridge Se você deseja criar um pipeline, consulte Criar um pipeline, estágios e ações.

Ao utilizar o console, o parâmetro `PollForSourceChanges` para o seu pipeline é alterado. O GitHub webhook é criado e registrado para você.

Para editar o estágio de origem do pipeline

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes de todos os canais associados à sua AWS conta são exibidos.

2. Em Nome, selecione o nome do pipeline que você deseja editar. Isso abrirá um visão detalhada do pipeline, incluindo o estado de cada uma das ações em cada estágio do pipeline.
3. Na página de detalhes do pipeline, selecione Editar.
4. Em Edit stage (Editar estágio), selecione o ícone de edição na ação de origem.
5. Expanda as opções de detecção de alterações e escolha Usar Amazon CloudWatch Events para iniciar automaticamente meu pipeline quando ocorrer uma alteração (recomendado).

   Uma mensagem é exibida para informar que CodePipeline cria um webhook GitHub para detectar alterações na fonte: AWS CodePipeline criará um webhook para você. Você pode recusar as opções abaixo. Selecione Atualizar. Além do webhook, CodePipeline cria o seguinte:

   - Um segredo, gerado aleatoriamente e usado para autorizar a conexão com o. GitHub
   - O URL do webhook, gerado por meio do uso do endpoint público da região.

   CodePipeline registra o webhook com. GitHub Isso inscreve o URL para receber eventos de repositório.
6. Quando terminar de editar seu pipeline, selecione Salvar alterações do pipeline para voltar à página de resumo.

   Uma mensagem exibe o nome do webhook a ser criado para o pipeline. Escolha Save and continue.
7. Para testar sua ação, libere uma alteração usando o AWS CLI para confirmar uma alteração na fonte especificada no estágio de origem do pipeline.

Migre canais de pesquisa para webhooks (por meio do aplicativo OAuthGitHub) (ações de origem) (CLI)

Siga essas etapas para fazer com que um pipeline que conduz sondagem (verificações periódicas) passe a utilizar um webhook. Se você deseja criar um pipeline, consulte Criar um pipeline, estágios e ações.

Para criar um pipeline baseado em eventos, edite o parâmetro `PollForSourceChanges` de seu pipeline e crie os seguintes recursos manualmente:

- GitHub parâmetros de webhook e autorização

Para criar e registrar seu webhook

> **Note**
>
> Ao usar a CLI ou CloudFormation criar um pipeline e adicionar um webhook, você deve desativar as verificações periódicas. Para desativar as verificações periódicas, é necessário adicionar explicitamente o parâmetro `PollForSourceChanges` e defini-lo como falso, conforme descrito no procedimento final abaixo. Caso contrário, o padrão para uma CLI ou CloudFormation pipeline é que o `PollForSourceChanges` padrão é true e não é exibido na saída da estrutura do pipeline. Para obter mais informações sobre PollForSourceChanges padrões, consulte. Configurações válidas para o parâmetro `PollForSourceChanges`

1. Em um editor de texto, crie e salve um arquivo JSON para o webhook que deseja criar. Use esse exemplo de arquivo para um 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. Use o comando put-webhook e inclua os parâmetros `--cli-input` e `--region`.

   O exemplo de comando a seguir cria um webhook com o arquivo JSON `webhook_json`.

```
aws codepipeline put-webhook --cli-input-json file://webhook_json.json --region
 "eu-central-1"
```

3. Na saída mostrada neste exemplo, a URL e o ARN são retornados para um 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"
    }]
}
```

Este exemplo adiciona tags ao webhook incluindo a chave de tag `Project` e o valor `ProjectA` no webhook. Para obter mais informações sobre como marcar recursos em CodePipeline, consulteMarcando atributos.

4. Use o comando register-webhook-with-third-party e inclua o parâmetro `--webhook-name`.

   O exemplo de comando a seguir registra um webhook denominado `my-webhook`.

```
aws codepipeline register-webhook-with-third-party --webhook-name my-webhook
```

Para editar o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

1. Execute o get-pipeline comando para copiar a estrutura do pipeline em um arquivo JSON. Por exemplo, para um pipeline chamado `MyFirstPipeline`, você deve digitar o seguinte comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando retorna nada, mas o arquivo que você criou deve aparecer no diretório onde você executou o comando.

2. Abra o arquivo JSON em qualquer editor de texto plano e altere ou adicione o parâmetro `PollForSourceChanges` para editar o estágio de origem. Neste exemplo, para um repositório denominado `UserGitHubRepo`, o parâmetro é definido como `false`.

   Por que estou fazendo essa alteração? A alteração deste parâmetro para desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em eventos.

   ```
   "configuration": {
       "Owner": "name",
       "Repo": "UserGitHubRepo",
       "PollForSourceChanges": "false",
       "Branch": "main",
       "OAuthToken": "****"
   },
   ```

3. Se você estiver trabalhando com a estrutura de pipeline recuperada por meio do comando get-pipeline, é necessário editar a estrutura no arquivo JSON por meio da remoção das linhas `metadata` do arquivo. Caso contrário, o comando update-pipeline não é capaz de utilizá-la.

Remova a seção `"metadata"` da estrutura do pipeline no arquivo JSON, incluindo os campos: `{ }`, `"created"`, `"pipelineARN"` e `"updated"`.

Por exemplo, remova as seguintes linhas da estrutura:

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

Salve o arquivo.

4. Para aplicar suas alterações, execute o comando update-pipeline, especificando o pipeline do arquivo JSON, de modo semelhante ao seguinte:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

Este comando retorna toda a estrutura do pipeline editado.

> **Note**
>
> O comando update-pipeline interrompe o pipeline. Se uma revisão estiver sendo executada pelo pipeline quando você executar o comando update-pipeline, essa execução será interrompida. Você deve iniciar manualmente o pipeline para executar a revisão através do pipeline atualizado. Use o comando start-pipeline-execution para iniciar manualmente o pipeline.

Atualize pipelines para eventos push GitHub (por meio do aplicativo OAuth) (ações de origem) (CloudFormation modelo)

Siga estas etapas para atualizar seu funil (com uma GitHub fonte) de verificações periódicas (pesquisas) para detecção de alterações baseada em eventos usando webhooks.

Para criar um pipeline orientado por eventos com AWS CodeCommit, você edita o `PollForSourceChanges` parâmetro do seu funil e adiciona um recurso de GitHub webhook ao seu modelo.

Se você usa CloudFormation para criar e gerenciar seus funis, seu modelo tem conteúdo como o seguinte.

> **Note**
>
> Observe a propriedade de configuração `PollForSourceChanges` no estágio de origem. Se o seu modelo não inclui essa propriedade, `PollForSourceChanges` é definido como `true` por padrão.

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
                            }
                        ]
                    },


...
```

Como adicionar parâmetros e criar um webhook em seu modelo

É altamente recomendável que você use AWS Secrets Manager para armazenar suas credenciais. Se você usar o Secrets Manager, já terá configurado e armazenado seus parâmetros secretos no Secrets Manager. Este exemplo usa referências dinâmicas ao Secrets Manager para as GitHub credenciais do seu webhook. Para obter mais informações, consulte Usar referências dinâmicas para especificar valores de modelo.

> **Important**
>
> Ao transmitir parâmetros secretos, não insira o valor diretamente no modelo. O valor é renderizado como texto sem formatação e, portanto, é legível. Por motivos de segurança, não use texto sem formatação em seu CloudFormation modelo para armazenar suas credenciais.

Ao usar a CLI ou CloudFormation criar um pipeline e adicionar um webhook, você deve desativar as verificações periódicas.

> **Note**
>
> Para desativar as verificações periódicas, é necessário adicionar explicitamente o parâmetro `PollForSourceChanges` e defini-lo como falso, conforme descrito no procedimento final abaixo. Caso contrário, o padrão para uma CLI ou CloudFormation pipeline é que o `PollForSourceChanges` padrão é true e não é exibido na saída da estrutura do pipeline. Para obter mais informações sobre PollForSourceChanges padrões, consulte. Configurações válidas para o parâmetro `PollForSourceChanges`

1. No modelo, em `Resources`, adicione os parâmetros:

   YAML

   ```
   Parameters:
           GitHubOwner:
             Type: String

   ...
   ```

   JSON

   ```
   {
    "Parameters": {
     "BranchName": {
      "Description": "GitHub branch name",
      "Type": "String",
      "Default": "main"
     },
     "GitHubOwner": {
      "Type": "String"
     },

   ...
   ```

2. Use o `AWS::CodePipeline::Webhook` CloudFormation recurso para adicionar um webhook.

> **Note**
>
> O `TargetAction` especificado deve corresponder à propriedade de `Name` da ação de origem definida no pipeline.

Se `RegisterWithThirdParty` estiver definido como`true`, certifique-se de que o usuário associado ao `OAuthToken` possa definir os escopos necessários em GitHub. O token e o webhook exigem os seguintes GitHub escopos:

- `repo` – usado para se obter o controle total para ler e efetuar pull de artefatos de repositórios públicos e privados para um pipeline.
- `admin:repo_hook` – usado para se obter o controle total dos ganchos do repositório.

Caso contrário, GitHub retorna um 404. Para obter mais informações sobre o 404 retornado, 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. Salve o modelo atualizado em seu computador local e abra o CloudFormation console.
4. Selecione sua pilha e clique em Create Change Set for Current Stack (Criar conjunto de alterações para a pilha atual).
5. Carregue o modelo e visualize as alterações listadas no CloudFormation. Essas são as alterações a serem feitas na pilha. Seus novos recursos devem ser exibidos na lista.
6. Clique em Executar.

Para editar o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

- No modelo, altere `PollForSourceChanges` para `false`. Se você não incluir `PollForSourceChanges` na sua definição de pipeline, adicione-o e configure para falso.

  Por que estou fazendo essa alteração? A alteração deste parâmetro para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em 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

Quando você cria esses recursos com CloudFormation, o webhook definido é criado no GitHub repositório especificado. Seu pipeline é acionado durante a confirmação.

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

...
```

Crie a função CodePipeline de serviço

Ao criar um pipeline, crie uma função de serviço ou use uma função de serviço existente.

Você pode usar o CodePipeline console ou o AWS CLI para criar uma função CodePipeline de serviço. Uma função de serviço é necessária para criar um pipeline e o pipeline sempre é associado a essa função de serviço.

Antes de criar um pipeline com a AWS CLI, você deve criar uma função CodePipeline de serviço para seu pipeline. Para ver um exemplo CloudFormation de modelo com a função de serviço e a política especificadas, consulte os tutoriais em. Tutorial: criar um pipeline que usa variáveis de AWS CloudFormation ações de implantação

A função de serviço não é uma função AWS gerenciada, mas é criada inicialmente para a criação do pipeline e, à medida que novas permissões são adicionadas à política da função de serviço, talvez seja necessário atualizar a função de serviço do seu funil. Assim que o pipeline for criado com uma

função de serviço, não será possível aplicar uma função de serviço diferente para esse pipeline. Anexe a política recomendada à função de serviço.

Para obter mais informações sobre a função de serviço, consulte Gerenciar a função CodePipeline de serviço.

Crie a função CodePipeline de serviço (console)

Ao usar o console para criar um pipeline, você cria a função de CodePipeline serviço com o assistente de criação de pipeline.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Escolha Create pipeline (Criar pipeline) e preencha a página Step 1: Choose pipeline settings (Etapa 1: Escolher as configurações de pipeline) no assistente de criação de pipeline.

   > **Note**
   >
   > Depois de criar um pipeline, não é possível alterar o nome dele. Para obter informações sobre outras limitações, consulte Cotas em AWS CodePipeline.

2. Em Função de serviço, escolha Nova função de serviço para permitir CodePipeline a criação de uma nova função de serviço no IAM.
3. Conclua a criação do pipeline. O perfil de serviço do pipeline estará disponível para visualização em sua lista de perfis do IAM. Você poderá visualizar o ARN do perfil de serviço associado a um pipeline executando o comando `get-pipeline` com a AWS CLI.

Crie a função CodePipeline de serviço (CLI)

Antes de criar um pipeline com a AWS CLI ou CloudFormation, você deve criar uma função de CodePipeline serviço para seu pipeline e anexar a política da função de serviço e a política de confiança. Para usar a CLI para criar seu perfil de serviço, execute as etapas abaixo para criar primeiro o arquivo JSON de uma política de confiança e o arquivo JSON da política de perfil como arquivos separados no diretório em que você executará os comandos da CLI.

> **Note**
>
> É recomendável que você permita que apenas usuários administrativos criem perfis de serviço. Uma pessoa com permissões para criar uma função e anexar qualquer política pode aumentar as próprias permissões. Em vez disso, crie uma política que permita criar apenas as funções necessárias ou peça para um administrador criar a função de serviço no nome dessas pessoas.

1. Em uma janela de terminal, insira o comando a seguir para criar um arquivo chamado `TrustPolicy.json`, no qual você colará o arquivo JSON da política de perfil. Este exemplo usa o comando VIM.

```
vim TrustPolicy.json
```

2. Cole a seguinte informação JSON no arquivo.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codepipeline.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   Para salvar e sair do arquivo, insira o seguinte comando VIM:

```
:wq
```

3. Em uma janela de terminal, insira o comando a seguir para criar um arquivo chamado `RolePolicy.json`, no qual você colará o arquivo JSON da política de perfil. Este exemplo usa o comando VIM.

```
vim RolePolicy.json
```

4. Cole a política JSON no arquivo. Use a política do perfil de serviço mínima, conforme indicado em CodePipeline política de função de serviço. Além disso, adicione as permissões indicadas ao perfil de serviço com base nas ações que você pretende usar. Para obter uma lista de ações e um link para as permissões do perfil de serviço necessárias de cada ação, consulte Adicionar permissões à função CodePipeline de serviço.

   Não se esqueça de reduzir ao máximo as permissões diminuindo o escopo até o nível do recurso no campo `Resource`.

   Para salvar e sair do arquivo, insira o seguinte comando VIM:

```
:wq
```

5. Use o comando a seguir para criar o perfil e anexar a política do perfil de confiança. O formato do nome de política normalmente é o mesmo formato do nome de função. Este exemplo usa o nome da função `MyRole` e a política `TrustPolicy` que foi criada como um arquivo separado.

```
aws iam create-role --role-name MyRole --assume-role-policy-document file://
TrustPolicy.json
```

6. Insira o comando a seguir para criar a política de perfil e anexá-la ao perfil. O formato do nome de política normalmente é o mesmo formato do nome de função. Este exemplo usa o nome da função `MyRole` e a política `MyRole` que foi criada como um arquivo separado.

```
aws iam put-role-policy --role-name MyRole --policy-name RolePolicy --policy-
document file://RolePolicy.json
```

7. Para visualizar o nome de perfil e a política de confiança criado, insira o seguinte comando para o perfil chamado `MyRole`:

```
aws iam get-role --role-name MyRole
```

8. Use o ARN da função de serviço ao criar seu pipeline com a AWS CLI ou. CloudFormation

Marcando atributos

Uma tag é um rótulo de atributo personalizado que você ou AWS atribui a um AWS recurso. Cada AWS tag tem duas partes:

- Uma chave de tag (por exemplo `CostCenter`, `Environment`, `Project` ou `Secret`). As chaves de tag diferenciam maiúsculas de minúsculas
- Um campo opcional conhecido como um valor de tag (por exemplo, `111122223333`, `Production` ou um nome de equipe). Omitir o valor da tag é o mesmo que usar uma string vazia. Assim como as chaves de tag, os valores das tags diferenciam maiúsculas de minúsculas.

Juntos, esses são conhecidos como pares de chave-valor.

As tags ajudam você a identificar e organizar seus AWS recursos. Muitos Serviços da AWS oferecem suporte à marcação, então você pode atribuir a mesma tag a recursos de serviços diferentes para indicar que os recursos estão relacionados. Por exemplo, a mesma tag atribuída a um pipeline pode ser atribuída a um bucket de origem do Amazon S3.

Para obter dicas sobre como usar tags, consulte a postagem AWS Tagging Strategies no blog AWS Answers.

É possível marcar os seguintes tipos de recursos no CodePipeline:

- Marque um funil em CodePipeline
- Marque uma ação personalizada em CodePipeline

Você pode usar as AWS CLI CodePipeline APIs ou os AWS SDKs para:

- Adicionar tags a um pipeline, ação personalizada ou webhook ao criá-los.
- Adicionar, gerenciar e remover tags de um pipeline, ação personalizada ou webhook.

Você também pode usar o console para adicionar, gerenciar e remover tags de um pipeline.

Além de identificar, organizar e monitorar seus recursos com etiquetas, você pode usar etiquetas em políticas do IAM para ajudar a controlar quem pode visualizar e interagir com o seu recurso. Para obter exemplos de políticas de acesso baseadas em tags, consulte Usando tags para controlar o acesso aos CodePipeline recursos.

Marque um funil em CodePipeline

As tags são pares de valores-chave associados AWS aos recursos. Você pode aplicar tags aos pipelines no CodePipeline. Para obter informações sobre marcação de CodePipeline recursos, casos de uso, restrições de valor e chave de tag e tipos de recursos compatíveis, consulte. Marcando atributos

Você pode usar a CLI para especificar tags ao criar um pipeline. Você pode usar o console ou a CLI para adicionar ou remover tags e atualizar os valores de tags em um pipeline. Você pode adicionar até 50 tags para cada pipeline.

Tópicos

- Marcar pipelines (console)
- Marcar pipelines (CLI)

Marcar pipelines (console)

Você pode usar o console ou a CLI para marcar recursos. Os pipelines são os únicos recursos do CodePipeline que podem ser gerenciados com o console ou a CLI.

Tópicos

- Adicionar tags a um pipeline (console)
- Visualizar tags de um pipeline (console)
- Editar tags de um pipeline (console)
- Remover tags de um pipeline (console)

Adicionar tags a um pipeline (console)

Você pode usar o console para adicionar tags a um pipeline existente.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http:// console.aws.amazon. com/codesuite/codepipeline/home.
2. Na página Pipelines, selecione o pipeline ao qual você deseja adicionar tags.
3. No painel de navegação, escolha Settings (Configurações).
4. Em Pipeline tags (Tags do pipeline), escolha Edit (Editar).

5. Nos campos Key (Chave) e Value (Valor), insira um par de chaves para cada conjunto de tags que você deseja adicionar. (O campo Value (Valor) é opcional.) Por exemplo, em Key (Chave), insira **`Project`**. Em Valor, informe **`ProjectA`**.
6. (Opcional) Escolha Add tag (Adicionar tag) para adicionar mais linhas e inserir mais tags.
7. Selecione Enviar. As tags são listadas em configurações de pipeline.

Visualizar tags de um pipeline (console)

Você pode usar o console para listar tags de pipelines existentes.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http:// console.aws.amazon. com/codesuite/codepipeline/home.
2. Na página Pipelines, selecione o pipeline no qual você deseja visualizar tags.
3. No painel de navegação, escolha Settings (Configurações).
4. Em Pipeline tags (Tags do pipeline), visualize as tags para o pipeline nas colunas Key (Chave) e Value (Valor).

Editar tags de um pipeline (console)

Você pode usar o console para editar as tags adicionadas a pipelines.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http:// console.aws.amazon. com/codesuite/codepipeline/home.
2. Na página Pipelines, selecione o pipeline no qual você deseja atualizar tags.
3. No painel de navegação, escolha Settings (Configurações).
4. Em Pipeline tags (Tags do pipeline), escolha Edit (Editar).
5. Nos campos Key (Chave) e Value (Valor), atualize os valores em cada campo conforme necessário. Por exemplo, para a chave **`Project`**, em Value (Valor), altere **`ProjectA`** para **`ProjectB`**.
6. Selecione Enviar.

Remover tags de um pipeline (console)

Você pode usar o console para excluir tags de pipelines. Ao remover tags do recurso associado, as tags são excluídas.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Na página Pipelines, selecione o pipeline do qual você deseja remover tags.
3. No painel de navegação, escolha Settings (Configurações).
4. Em Pipeline tags (Tags do pipeline), escolha Edit (Editar).
5. Ao lado da chave e do valor para cada tag que você deseja excluir, escolha Remove tag (Remover tag).
6. Selecione Enviar.

Marcar pipelines (CLI)

Você pode usar a CLI para marcar recursos. Você deve usar o console para gerenciar tags em pipelines.

Tópicos

- Adicionar tags a um pipeline (CLI)
- Visualizar tags de um pipeline (CLI)
- Editar tags de um pipeline (CLI)
- Remover tags de um pipeline (CLI)

Adicionar tags a um pipeline (CLI)

Você pode usar o console ou o AWS CLI para marcar pipelines.

Para adicionar uma tag a um pipeline ao criá-lo, consulte Criar um pipeline, estágios e ações.

Nestas etapas, partimos do princípio de que você já instalou uma versão recente da AWS CLI ou atualizou para a versão atual. Para obter mais informações, consulte Instalar a AWS Command Line Interface.

No terminal ou na linha de comando, execute o comando tag-resource, especificando o nome de recurso da Amazon (ARN) do pipeline no qual você deseja adicionar tags e a chave e o valor da tag que você deseja adicionar. Você pode adicionar mais de uma tag a um pipeline. Por exemplo, para marcar um pipeline chamado `MyPipeline` com duas tags, uma chave de tag `DeploymentEnvironment` com o valor de tag de `Test` e uma chave de tag `IscontainerBased` com o valor de tag 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
```

Se houver êxito, o comando não retornará nada.

Visualizar tags de um pipeline (CLI)

Siga estas etapas para usar o AWS CLI para visualizar as AWS tags de um pipeline. Se não foram adicionadas tags, a lista retornará vazia.

No terminal ou na linha de comando, execute o comando list-tags-for-resource. Por exemplo, para ver uma lista de chaves e valores de tag para um pipeline chamado *MyPipeline* com o 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
```

Se houver êxito, o comando retornará informações semelhantes às seguintes:

```
{
    "tags": {
        "Project": "ProjectA",
        "IscontainerBased": "true"
    }
}
```

Editar tags de um pipeline (CLI)

Siga estas etapas para usar o AWS CLI para editar uma tag para um funil. Você pode alterar o valor para uma chave existente ou adicionar outra chave. Você também pode remover tags de um pipeline, como mostrado na próxima seção.

No terminal ou na linha de comando, execute o comando tag-resource, especificando o ARN do pipeline em que você deseja atualizar uma tag e especifique a chave e o valor da tag:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tags key=Project,value=ProjectA
```

Se houver êxito, o comando não retornará nada.

Remover tags de um pipeline (CLI)

Siga estas etapas para usar o AWS CLI para remover uma tag de um pipeline. Ao remover tags do recurso associado, as tags são excluídas.

> **Note**
>
> Se você excluir um pipeline, todas as associações de tag serão removidas do pipeline excluído. Você não precisa remover as tags antes de excluir um pipeline.

No terminal ou na linha de comando, execute o comando untag-resource, especificando o ARN do pipeline de onde você deseja remover tags e a chave da tag que você deseja remover. Por exemplo, para remover várias tags em um pipeline chamado *MyPipeline* com as chaves de tag *Project* e*IscontainerBased*:

```
aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tag-keys Project IscontainerBased
```

Se houver êxito, o comando não retornará nada. Para verificar as tags associadas ao pipeline, execute o comando list-tags-for-resource.

Criar uma regra de notificação

Você pode usar regras de notificação para notificar os usuários sobre alterações importantes, como quando um pipeline inicia a execução. As regras de notificação especificam os eventos e o tópico do Amazon SNS utilizado para enviar notificações. Para obter mais informações, consulte O que são notificações?

Você pode usar o console ou o AWS CLI para criar regras de notificação para AWS CodePipeline.

Como criar uma regra de notificação (console)

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Escolha Pipelines e selecione um pipeline onde você deseja adicionar notificações.

3. Na página do pipeline, escolha Notify (Notificar) e Create notification rule (Criar regra de notificação). Você também pode ir para a página Settings (Configurações) do pipeline e escolher Create notification rule (Criar regra de notificação).
4. Em Notification name (Nome da notificação), insira um nome para a regra.
5. Em Tipo de detalhe, escolha Básico se quiser que somente as informações fornecidas à Amazon sejam EventBridge incluídas na notificação. Escolha Completo se quiser incluir informações fornecidas à Amazon EventBridge e informações que possam ser fornecidas pelo gerenciador de notificações CodePipeline ou pelo gerenciador de notificações.

   Para obter mais informações, consulte Noções básicas sobre o conteúdo e a segurança de notificações.
6. Em Events that trigger notifications (Eventos que acionam notificações), selecione os eventos para os quais você deseja enviar notificações. Para obter mais informações, consulte Eventos para regras de notificação em pipelines.
7. Em Targets (Destinos), siga um destes procedimentos:

   - Se você já tiver configurado um recurso a ser usado com notificações, em Escolher tipo de destino, escolha Amazon Q Developer em aplicações de chat (Slack) ou Tópico do SNS. Em Escolher destino, escolha o nome do cliente (para um cliente do Slack configurado no Amazon Q Developer em aplicações de chat) ou o nome do recurso da Amazon (ARN) do tópico do Amazon SNS (para tópicos do Amazon SNS já configurados com a política necessária para notificações).
   - Se você não configurou um recurso para usar com notificações, escolha Create target (Criar destino) e selecione SNS topic (Tópico do SNS). Forneça um nome para o tópico após codestar-notifications- e escolha Create (Criar).

   > **Note**
   >
   > - Ao criar o tópico do Amazon SNS como parte da criação da regra de notificação, a política que permite ao recurso publicar eventos no tópico é aplicada para você. O uso de um tópico criado para regras de notificação ajuda a garantir que você inscreva somente os usuários para os quais deseja enviar notificações sobre esse recurso.
   > - Você não pode criar um Amazon Q Developer no cliente de aplicações de chat como parte da criação de uma regra de notificação. Se escolher o Amazon Q Developer em aplicações de chat (Slack), você verá um botão direcionando para configurar um

cliente no Amazon Q Developer em aplicações de chat. A escolha dessa opção abre o Amazon Q Developer no console de aplicações de chat. Para obter mais informações, consulte Configurar integrações entre notificações e o Amazon Q Developer em aplicações de chat.

- Se quiser usar um tópico do Amazon SNS existente como destino, você deverá adicionar a política necessária para o AWS CodeStar Notifications, além de quaisquer outras políticas que possam existir para esse tópico. Para obter mais informações, consulte Configurar tópicos do Amazon SNS existentes para notificações e Noções básicas sobre conteúdos de notificações e segurança.

8. Para concluir a criação da regra, escolha Submit (Enviar).
9. Você precisa inscrever os usuários no tópico do Amazon SNS para a regra antes que eles possam receber notificações. Para obter mais informações, consulte Inscrever usuários em tópicos do Amazon SNS que são destinos. Você também pode configurar a integração entre as notificações e o Amazon Q Developer em aplicações de chat a fim de enviar notificações para salas de chat do Amazon Chime ou canais do Slack. Para obter mais informações, consulte Configurar integrações entre notificações e o Amazon Q Developer em aplicações de chat.

Criar uma regra de notificação (AWS CLI)

1. Em um terminal ou prompt de comando, execute o comando create-notification rule para gerar o esqueleto JSON:

```
aws codestar-notifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

É possível nomear o arquivo como você quiser. Neste exemplo, o nome do arquivo é *rule.json*.

2. Abra o arquivo JSON em um editor de texto simples e edite-o para incluir o recurso, os tipos de evento e o destino que você deseja para a regra. O exemplo a seguir mostra uma regra de notificação com **MyNotificationRule** o nome de um pipeline nomeado *MyDemoPipeline* em uma AWS conta com o ID*123456789012*. As notificações são enviadas com o tipo de detalhe completo para um tópico do Amazon SNS chamado *codestar-notifications-MyNotificationTopic* quando as execuções do pipeline começam:

```
{
```

```
    "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"
}
```

Salve o arquivo.

3. Usando o arquivo que você acabou de editar, no terminal ou na linha de comando, execute o comando create-notification-rule novamente para criar a regra de notificação:

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. Se for bem-sucedido, o comando retornará o ARN da regra de notificação, semelhante ao seguinte:

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

Ações de origem e métodos de detecção de alterações

Quando você adiciona uma ação de origem ao seu pipeline, as ações funcionam com os recursos adicionais descritos na tabela.

> **Note**
>
> As ações de origem CodeCommit e do S3 exigem um recurso de detecção de alterações configurado (uma EventBridge regra) ou use a opção de pesquisar alterações na fonte no repositório. Para pipelines com uma ação de origem do Bitbucket ou do GitHub Enterprise Server, você não precisa configurar um webhook ou usar a pesquisa como padrão. GitHub A ação do Connections gerencia a detecção de alterações para você.

Fonte	Usa recursos adicionais?	Etapas	
Amazon S3 com recursos CloudTrail	Essa ação de origem usa uma regra de evento e CloudTrail recursos adicionais. Ao usar a CLI ou CloudFormation para criar essa ação, você também cria e gerencia esses recursos.	Consulte Criar um pipeline, estágios e ações e Conectando-se às ações de origem do Amazon S3 que usam EventBridge e AWS CloudTrail	
Amazon S3 sem recursos CloudTrail	Essa ação de origem usa um bucket habilitado para eventos com uma regra de evento sem a necessidade de CloudTrail recursos adicionais. Ao usar a CLI ou CloudFormation para criar essa ação, você também cria e gerencia esses recursos.	Consulte Criar um pipeline, estágios e ações e Conexão a ações de origem do Amazon S3 com uma fonte habilitada para eventos	

Fonte	Usa recursos adicionais?	Etapas
Bitbucket Cloud	Esta ação de origem usa um recurso de conexão.	Consulte Conexões do Bitbucket Cloud
AWS CodeCommit	Amazon EventBridge (recomendado). Este é o padrão para pipelines com uma origem do CodeCommit criada ou editada no console.	Consulte Criar um pipeline, estágios e ações e CodeCommit ações de origem e EventBridge
Amazon ECR	Amazon EventBridge. É criado pelo assistente para pipelines com uma origem do Amazon ECR criada ou editada no console.	Consulte Criar um pipeline, estágios e ações e Ações e recursos de origem do Amazon ECR EventBridge.
GitHub ou nuvem GitHub corporativa	Esta ação de origem usa um recurso de conexão.	Consulte GitHub conexões
GitHub Servidor corporativo	Esta ação de origem usa um recurso de conexão e um recurso de host.	Consulte GitHub Conexões do Enterprise Server
GitLab.com	Esta ação de origem usa um recurso de conexão.	Consulte GitLab.com conexões
GitLab autogerenciado	Esta ação de origem usa um recurso de conexão e um recurso de host.	Consulte Conexões para GitLab autogerenciamento

Se você tiver um pipeline que usa sondagem, pode atualizá-lo para usar o método de detecção recomendado. Para obter mais informações, consulte Atualizar pipelines de sondagem para o método de detecção de alterações recomendado.

Se você quiser desativar a detecção de alterações para uma ação de origem que usa conexões, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

Conecte-se a provedores de origem primários usando ações de origem

Você pode usar o AWS CodePipeline console ou o AWS CLI para se conectar aos provedores de ação de origem, como CodeCommit ou o S3.

> **Note**
>
> Quando o console é usado para criar ou editar um pipeline, os recursos de detecção de alterações são criados para você. Se você usar a AWS CLI para criar o pipeline, deverá criar você mesmo os recursos adicionais. Para obter mais informações, consulte CodeCommit ações de origem e EventBridge.

Tópicos

- Ações e recursos de origem do Amazon ECR EventBridge
- Conexão a ações de origem do Amazon S3 com uma fonte habilitada para eventos
- Conectando-se às ações de origem do Amazon S3 que usam EventBridge e AWS CloudTrail
- CodeCommit ações de origem e EventBridge

Ações e recursos de origem do Amazon ECR EventBridge

Para adicionar uma ação de origem do Amazon ECR CodePipeline, você pode escolher entre:

- Use o CodePipeline console Create Pipeline Wizard (Criar um pipeline personalizado (console)) ou a página de ação Editar para escolher a opção de provedor do Amazon ECR. O console cria uma EventBridge regra que inicia seu pipeline quando a fonte muda.
- Usar a CLI para adicionar a configuração da ação `ECR` e criar recursos adicionais da seguinte forma:
  - Use o exemplo de configuração de ação `ECR` em Referência da ação de origem do Amazon ECR para criar sua ação, conforme mostrado em Criar um pipeline (CLI).
  - O método de detecção de alterações assume como padrão iniciar o pipeline por meio da sondagem da origem. Você deve desabilitar as verificações periódicas e criar a regra de

detecção de alterações manualmente. Use um dos seguintes métodos: Crie uma EventBridge regra para uma fonte do Amazon ECR (console), Crie uma EventBridge regra para uma fonte do Amazon ECR (CLI) ou Crie uma EventBridge regra para uma fonte do Amazon ECR (CloudFormation modelo).

Tópicos

- Crie uma EventBridge regra para uma fonte do Amazon ECR (console)
- Crie uma EventBridge regra para uma fonte do Amazon ECR (CLI)
- Crie uma EventBridge regra para uma fonte do Amazon ECR (CloudFormation modelo)

Crie uma EventBridge regra para uma fonte do Amazon ECR (console)

Para criar uma EventBridge regra para uso em CodePipeline operações (fonte do Amazon ECR)

1. Abra o EventBridge console da Amazon em https://console.aws.amazon.com/events/.
2. No painel de navegação, escolha Eventos.
3. Selecione Criar regra e, em Origem do evento, em Nome do serviço, escolha Elastic Container Registry (ECR).
4. Em Event Source (Origem do evento), selecione Event Pattern (Padrão do evento).

   Selecione Edit (Editar) e cole o seguinte exemplo de padrão de evento na janela Event Source (Origem do evento) para um repositório `eb-test` com uma tag de imagem 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 o padrão completo de eventos suportado pelos eventos do Amazon ECR, consulte Amazon ECR Events EventBridge e/ou Amazon Elastic Container Registry Events.

5. Escolha Salvar.

   No painel Event Pattern Preview, visualize a regra.

6. Em Metas, escolha CodePipeline.
7. Insira o ARN do pipeline a ser iniciado por esta regra.

> **Note**
>
> Você pode encontrar o ARN do pipeline na saída de metadados após executar o comando get-pipeline. O ARN do pipeline é construído neste formato: arn:aws:codepipeline::: *region account pipeline-name*
> Exemplo de ARN do pipeline: `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

8. Crie ou especifique uma função de serviço do IAM que conceda EventBridge permissões para invocar o destino associado à sua EventBridge regra (nesse caso, o destino é CodePipeline).

   - Escolha Criar uma nova função para esse recurso específico para criar uma função de serviço que dê EventBridge permissões para iniciar suas execuções de pipeline.
   - Escolha Usar função existente para inserir uma função de serviço que dê EventBridge permissões para iniciar suas execuções de pipeline.

9. (Opcional) Para especificar substituições de origem com um ID de imagem específico, use o transformador de entrada para passar os dados como parâmetros JSON. Você também pode usar o transformador de entrada para passar variáveis de pipeline.

- Expanda Additional settings (Configurações adicionais).

  Em Configurar entrada de destino, escolha Configurar transformador de entrada.

  Na janela de diálogo, escolha Inserir um próprio. Na caixa Caminho de entrada, digite os pares de chave-valor a seguir.

```
{"revisionValue": "$.detail.image-digest"}
```

- Na caixa Modelo, digite os pares de chave-valor a seguir.

```
{
    "sourceRevisions": [
        {
            "actionName": "Source",
            "revisionType": "IMAGE_DIGEST",
            "revisionValue": "<revisionValue>"
        }
    ],
     "variables": [
        {
            "name": "Variable_Name",
            "value": "Variable_Value"
        }
    ]
}
```

- Escolha Confirmar.

10. Revise a configuração da regra para garantir que ela atenda aos requisitos.
11. Escolha Configure details (Configurar detalhes).
12. Na página Configure rule details (Configurar detalhes da regra), informe um nome e uma descrição para a regra e selecione State (Estado) para habilitá-la.
13. Se você estiver satisfeito com a regra, escolha Create rule.

Crie uma EventBridge regra para uma fonte do Amazon ECR (CLI)

Use o comando put-rule, especificando:

- Um nome que identifique de forma exclusiva a regra que você está criando. Esse nome deve ser exclusivo em todos os funis que você cria CodePipeline associados à sua AWS conta.
- O padrão de evento para a origem e os campos detalhados usados pela regra. Para obter mais informações, consulte Amazon EventBridge e Event Patterns.

Para criar uma EventBridge regra com o Amazon ECR como origem e destino CodePipeline do evento

1. Adicione permissões para usar EventBridge CodePipeline para invocar a regra. Para obter mais informações, consulte Usando políticas baseadas em recursos para a Amazon. EventBridge

   a. Use o exemplo a seguir para criar a política de confiança que permite ao EventBridge assumir a função de serviço. Nomeie a política de confiança `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Use o comando a seguir para criar a função `Role-for-MyRule` e anexar a política de confiança.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Crie o JSON de política de permissões, conforme mostrado neste exemplo, para o pipeline denominado `MyFirstPipeline`. Nomeie a politica de permissões `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Use o comando a seguir para anexar a política de permissões `CodePipeline-Permissions-Policy-for-EB` à função `Role-for-MyRule`.

   Por que estou fazendo essa alteração? Adicionar essa política à função cria permissões 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. Use o comando put-rule e inclua os parâmetros `--name`, `--event-pattern` e `--role-arn`.

   Por que estou fazendo essa alteração? Você deve criar um evento com uma regra que especifique como deve ser feito um envio de imagem por push e um destino que nomeie o pipeline a ser iniciado pelo evento.

   O comando de exemplo a seguir cria uma regra chamada `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 o padrão completo de eventos suportado pelos eventos do Amazon ECR, consulte Amazon ECR Events EventBridge e/ou Amazon Elastic Container Registry Events.

3. Para adicionar CodePipeline como destino, chame o put-targets comando e inclua os seguintes parâmetros:

   - O parâmetro `--rule` é usado com `rule_name` criado por meio de put-rule.
   - O parâmetro `--targets` é usado com o `Id` da lista do destino na lista de destinos e o `ARN` do pipeline de destino.

   O exemplo de comando a seguir especifica que, para a regra chamada `MyECRRepoRule`, o `Id` do destino é composto do número um, indicando que, em uma lista de destinos para a regra, esse é o destino 1. O comando de exemplo também especifica um exemplo `Arn` para o pipeline e o exemplo `RoleArn` para a regra. O pipeline é iniciado quando uma alteração é feita no repositório.

   ```
   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 um transformador de entrada com substituições de origem para um ID de imagem específico, use o JSON a seguir no comando da CLI. O exemplo a seguir configura uma substituição na qual:

   - O `actionName`, `Source` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O `revisionType`, `IMAGE_DIGEST` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O`revisionValue`, **<** *`revisionValue`* **>** neste exemplo, é derivado da variável de evento de origem.

   ```
   {
   ```

```
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "ARN",
            "InputTransformer": {
                "InputPathsMap": {
                    "revisionValue": "$.detail.image-digest"
                },
                "InputTemplate": {
                    "sourceRevisions": [
                        {
                            "actionName": "Source",
                            "revisionType": "IMAGE_DIGEST",
                            "revisionValue": "<revisionValue>"
                        }
                    ]
                }
            }
        }
    ]
}
```

Crie uma EventBridge regra para uma fonte do Amazon ECR (CloudFormation modelo)

Para usar CloudFormation para criar uma regra, use o trecho de modelo conforme mostrado aqui.

Para atualizar seu funil CloudFormation modelo e EventBridge regra de criação

1. No modelo, abaixo`Resources`, use o `AWS::IAM::Role` CloudFormation recurso para configurar a função do IAM que permite que seu evento inicie seu pipeline. Essa entrada cria uma função que utiliza duas políticas:

   - A primeira política permite que a função seja assumida.
   - A segunda política fornece permissões para iniciar o pipeline.

   Por que estou fazendo essa alteração? Você deve criar uma função que possa ser assumida EventBridge para iniciar uma execução em nosso pipeline.

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. No modelo, abaixo`Resources`, use o `AWS::Events::Rule` CloudFormation recurso para adicionar uma EventBridge regra para a fonte do Amazon ECR. Esse padrão de evento cria um evento que monitora as confirmações no seu repositório. Quando EventBridge detecta uma alteração no estado do repositório, a regra é invocada `StartPipelineExecution` em seu pipeline de destino.

   Por que estou fazendo essa alteração? Você deve criar um evento com uma regra que especifique como deve ser feito um envio de imagem por push e um destino que nomeie o pipeline a ser iniciado pelo evento.

   Esse trecho usa uma imagem chamada `eb-test` com uma tag de `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 o padrão completo de eventos suportado pelos eventos do Amazon ECR, consulte Amazon ECR Events EventBridge e/ou Amazon Elastic Container Registry Events.

3. (Opcional) Para configurar um transformador de entrada com substituições de origem para um ID de imagem específico, use o trecho YAML a seguir. O exemplo a seguir configura uma substituição na qual:

   - O `actionName`, `Source` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.

- O `revisionType`, `IMAGE_DIGEST` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
- O`revisionValue`, **<** *`revisionValue`* **>** neste exemplo, é derivado da variável de evento de origem.

```
---
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: ARN
  InputTransformer:
    InputPathsMap:
      revisionValue: "$.detail.image-digest"
    InputTemplate:
      sourceRevisions:
        actionName: Source
        revisionType: IMAGE_DIGEST
        revisionValue: '<revisionValue>'
```

4. Salve o modelo atualizado em seu computador local e abra o console do CloudFormation .
5. Selecione sua pilha e clique em Create Change Set for Current Stack (Criar conjunto de alterações para a pilha atual).
6. Carregue o modelo e visualize as alterações listadas no CloudFormation. Essas são as alterações a serem feitas na pilha. Seus novos recursos devem ser exibidos na lista.
7. Clique em Executar.

Conexão a ações de origem do Amazon S3 com uma fonte habilitada para eventos

As instruções nesta seção fornecem as etapas para criar a ação de origem do S3 que não exige a criação ou o gerenciamento de AWS CloudTrail recursos.

> **Important**
>
> O procedimento para criar essa ação sem AWS CloudTrail recursos não está disponível no console. Para usar a CLI, consulte os procedimentos aqui ou Migrar pipelines de sondagem com uma origem do S3 habilitada para eventos.

Para um pipeline com uma origem do Amazon S3, modifique o pipeline para que a detecção de alterações seja automatizada por meio de EventBridge e com um bucket de origem habilitado para notificações de eventos. Esse é o método recomendado se você estiver usando a CLI ou CloudFormation para migrar seu pipeline.

> **Note**
>
> Isso inclui o uso de um bucket habilitado para notificações de eventos, em que você não precisa criar uma CloudTrail trilha separada. Se você estiver usando o console, uma regra de evento e uma CloudTrail trilha serão configuradas para você. Para obter essas etapas, consulte Migre canais de votação com uma origem e uma trilha do S3 CloudTrail.

- CLI: Migre canais de pesquisa com uma origem e CloudTrail trilha (CLI) do S3
- CloudFormation: Migre canais de votação com uma origem e uma trilha do S3 (CloudTrail CloudFormation modelo)

Criar pipelines com uma origem do S3 habilitada para eventos (CLI)

Siga estas etapas para criar um pipeline com uma fonte do S3 que usa um evento in EventBridge para detecção de alterações. Para ver as etapas completas de criar um pipeline com a CLI, consulte Criar um pipeline, estágios e ações.

Para criar um pipeline baseado em eventos com o Amazon S3, edite o parâmetro `PollForSourceChanges` de seu pipeline e crie os seguintes recursos:

- EventBridge regra do evento
- Função do IAM para permitir que o EventBridge evento inicie seu funil

Criar uma EventBridge regra com o Amazon S3 como origem e CodePipeline destino do evento e aplicar a política de permissões

1. Conceda permissões EventBridge para usar CodePipeline para invocar a regra. Para obter mais informações, consulte Usando políticas baseadas em recursos para a Amazon. EventBridge

   a. Use o exemplo a seguir para criar a política de confiança que permita ao EventBridge assumir a função de serviço. Chame-o de `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Use o comando a seguir para criar a função `Role-for-MyRule` e anexar a política de confiança.

      Por que estou fazendo essa alteração? A adição dessa política de confiança à função cria permissões para EventBridge.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Crie o JSON da política de permissões, conforme mostrado aqui para o pipeline chamado `MyFirstPipeline`. Nomeie a politica de permissões `permissionspolicyforEB.json`.

      JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Use o comando a seguir para anexar a nova política de permissões `CodePipeline-Permissions-Policy-for-EB` à função `Role-for-MyRule` criada.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Use o comando put-rule e inclua os parâmetros `--name`, `--event-pattern` e `--role-arn`.

   O exemplo de comando a seguir cria uma regra chamada `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 adicionar CodePipeline como destino, chame o put-targets comando e inclua `--rule` `--targets` os parâmetros e.

   O comando a seguir especifica que, para a regra denominada `EnabledS3SourceRule`, o `Id` do destino é composto do número um, indicando que, em uma lista de destinos para a regra, esse é o destino 1. O comando também especifica um ARN de exemplo para o pipeline. O pipeline é iniciado quando uma alteração é feita no repositório.

```
aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-
AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

Para editar o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

1. Execute o get-pipeline comando para copiar a estrutura do pipeline em um arquivo JSON. Por exemplo, para um pipeline nomeado `MyFirstPipeline`, execute o seguinte comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando retorna nada, mas o arquivo que você criou deve aparecer no diretório onde você executou o comando.

2. Abra o arquivo JSON em qualquer editor de texto plano e, para editar o estágio de origem, altere o parâmetro `PollForSourceChanges` de um bucket denominado `amzn-s3-demo-source-bucket` para `false`, conforme mostrado neste exemplo.

   Por que estou fazendo essa alteração? A configuração deste parâmetro para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em eventos.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Se você estiver trabalhando com a estrutura do pipeline recuperada por meio do comando get-pipeline, é necessário remover as linhas `metadata` do arquivo JSON. Caso contrário, o comando update-pipeline não é capaz de utilizá-la. Remova as linhas `"metadata": { }`, `"created"`, `"pipelineARN"` e os campos `"updated"`.

   Por exemplo, remova as seguintes linhas da estrutura:

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

Salve o arquivo.

4. Para aplicar suas alterações, execute o comando update-pipeline especificando o arquivo JSON do pipeline:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

Este comando retorna toda a estrutura do pipeline editado.

> **Note**
>
> O comando update-pipeline interrompe o pipeline. Se uma revisão estiver sendo executada pelo pipeline quando você executar o comando update-pipeline, essa execução será interrompida. Você deve iniciar manualmente o pipeline para executar a revisão através do pipeline atualizado. Use o comando start-pipeline-execution para iniciar manualmente o pipeline.

Crie pipelines com uma fonte S3 habilitada para eventos (CloudFormation modelo)

Esse procedimento refere-se a um pipeline em que o bucket de origem tem eventos habilitados.

Use estas etapas a fim de criar um pipeline com uma origem do Amazon S3 para detecção de alterações baseada em eventos.

Para criar um pipeline baseado em eventos com o Amazon S3, edite o parâmetro `PollForSourceChanges` de seu pipeline e adicione os seguintes recursos ao modelo:

- EventBridge regra e função do IAM para permitir que esse evento inicie seu pipeline.

Se você usa CloudFormation para criar e gerenciar seus funis, seu modelo inclui conteúdo como o seguinte.

> **Note**
>
> A propriedade `Configuration` no estágio de origem denominado `PollForSourceChanges`. Se o seu modelo não inclui essa propriedade, `PollForSourceChanges` é definido como `true` por padrão.

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
                            }
                        ]
                    },
```

```
...
```

Criar uma EventBridge regra com o Amazon S3 como origem e CodePipeline destino do evento e aplicar a política de permissões

1. No modelo, abaixo`Resources`, use o `AWS::IAM::Role` CloudFormation recurso para configurar a função do IAM que permite que seu evento inicie seu pipeline. Essa entrada cria uma função que utiliza duas políticas:

   - A primeira política permite que a função seja assumida.
   - A segunda política fornece permissões para iniciar o pipeline.

   Por que estou fazendo essa alteração? Adicionar `AWS::IAM::Role` recursos permite CloudFormation criar permissões para EventBridge. Esse recurso é adicionado à sua CloudFormation pilha.

   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. Use o `AWS::Events::Rule` CloudFormation recurso para adicionar uma EventBridge regra. Esse padrão de evento cria um evento que monitora a criação ou exclusão de objetos no bucket de origem do Amazon S3. Além disso, inclua um destino de seu pipeline. Quando um objeto é criado, essa regra é invoca `StartPipelineExecution` em seu pipeline de destino.

   Por que estou fazendo essa alteração? Adicionar o `AWS::Events::Rule` recurso permite CloudFormation criar o evento. Esse recurso é adicionado à sua CloudFormation pilha.

   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. Salve o modelo atualizado no computador local e abra o console do CloudFormation .
4. Selecione sua pilha e clique em Create Change Set for Current Stack (Criar conjunto de alterações para a pilha atual).
5. Carregue o modelo atualizado e, em seguida, visualize as alterações listadas no CloudFormation. Essas são as alterações que serão feitas na pilha. Seus novos recursos devem ser exibidos na lista.
6. Clique em Executar.

Para editar o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas

vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

- No modelo, altere `PollForSourceChanges` para `false`. Se você não incluir `PollForSourceChanges` na sua definição de pipeline, adicione-o e configure para `false`.

  Por que estou fazendo essa alteração? A alteração de `PollForSourceChanges` para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em 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

Quando você usa CloudFormation para criar esses recursos, seu funil é acionado quando os arquivos em seu repositório são criados ou atualizados.

> **Note**
>
> Não pare por aqui. Embora seu pipeline tenha sido criado, você deve criar um segundo CloudFormation modelo para seu pipeline do Amazon S3. Se você não criar o segundo modelo, o pipeline não apresentará funcionalidades de detecção de alterações.

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"
        }
    ]
}
```

Conectando-se às ações de origem do Amazon S3 que usam EventBridge e AWS CloudTrail

As instruções nesta seção fornecem as etapas para criar a ação de origem do S3 que usa AWS CloudTrail recursos que você deve criar e gerenciar. Para usar a ação de origem do S3 sem exigir AWS CloudTrail recursos adicionais, use as instruções da CLI em. EventBridge Migrar pipelines de sondagem com uma origem do S3 habilitada para eventos

> **Important**
>
> Este procedimento fornece as etapas para criar a ação de origem do S3 que usa recursos do AWS CloudTrail que você deve criar e gerenciar. O procedimento para criar essa ação sem AWS CloudTrail recursos não está disponível no console. Para usar a CLI, consulte Migrar pipelines de sondagem com uma origem do S3 habilitada para eventos.

Para adicionar uma ação de origem do Amazon S3 CodePipeline, você escolhe:

- Use o assistente de criação de pipeline do CodePipeline console (Criar um pipeline personalizado (console)) ou a página Editar ação para escolher a opção de provedor do S3. O console cria uma EventBridge regra e uma CloudTrail trilha que iniciam seu funil quando a fonte muda.

- Use o AWS CLI para adicionar a configuração da S3 ação e criar recursos adicionais da seguinte forma:
  - Use o exemplo de configuração de ação S3 em Referência da ação de origem do Amazon S3 para criar sua ação, conforme mostrado em Criar um pipeline (CLI).
  - O método de detecção de alterações assume como padrão iniciar o pipeline por meio da sondagem da origem. Você deve desabilitar as verificações periódicas e criar a regra e a trilha de detecção de alterações manualmente. Use um dos seguintes métodos: Crie uma EventBridge regra para uma fonte do Amazon S3 (console), Crie uma EventBridge regra para uma fonte (CLI) do Amazon S3 ou Crie uma EventBridge regra para uma fonte do Amazon S3 (CloudFormation modelo).

AWS CloudTrail é um serviço que registra e filtra eventos em seu bucket de origem do Amazon S3. A trilha envia as alterações da fonte filtrada para a EventBridge regra. A EventBridge regra detecta a alteração na origem e, em seguida, inicia seu funil.

Requisitos:

- Se você não estiver criando uma trilha, use uma AWS CloudTrail trilha existente para registrar eventos em seu bucket de origem do Amazon S3 e enviar eventos filtrados para a EventBridge regra.
- Crie ou use um bucket S3 existente onde AWS CloudTrail possa armazenar seus arquivos de log. AWS CloudTrail deve ter as permissões necessárias para entregar arquivos de log a um bucket do Amazon S3. O bucket não pode ser configurado como um bucket de Pagamento pelo solicitante. Quando você cria um bucket do Amazon S3 como parte da criação ou atualização de uma trilha no console, AWS CloudTrail anexa as permissões necessárias a um bucket para você. Para obter mais informações, consulte a Política de bucket do Amazon S3 para CloudTrail.

Crie uma EventBridge regra para uma fonte do Amazon S3 (console)

Antes de configurar uma regra em EventBridge, você deve criar uma AWS CloudTrail trilha. Para obter mais informações, consulte Criar uma trilha no console.

> **Important**
>
> Se você usa o console para criar ou editar seu funil, sua EventBridge regra e AWS CloudTrail trilha são criadas para você.

Para criar uma trilha

1. Abra o AWS CloudTrail console.
2. No painel de navegação, selecione Trilhas.
3. Escolha Create Trail (Criar trilha). Em Trail name (Nome da trilha), informe um nome para a trilha.
4. Em Storage location (Local de armazenamento), crie ou especifique o bucket a ser usado para armazenar os arquivos de log. Por padrão, os buckets e objetos do Amazon S3 são privados. Somente o proprietário do recurso (a AWS conta que criou o bucket) pode acessar o bucket e seus objetos. O bucket deve ter uma política de recursos que permita AWS CloudTrail permissões para acessar os objetos no bucket.
5. Em Bucket e pasta de log de trilha, especifique um bucket do Amazon S3 e o prefixo do objeto (nome da pasta) para registrar em log os eventos de dados de todos os objetos da pasta. Para cada trilha, você pode adicionar até 250 objetos do Amazon S3. Preencha as informações necessárias da chave de criptografia e escolha Próximo.
6. Em Tipo de evento, escolha Eventos de gerenciamento.
7. Em Eventos de gerenciamento, escolha Editar. A trilha registra a atividade da API no nível do objeto do Amazon S3 (por exemplo, `GetObject` e `PutObject`) no bucket e no prefixo especificados.
8. Selecione Write (Gravar).
9. Se você estiver satisfeito com a trilha, escolha Criar trilha.

Para criar uma EventBridge regra que vise seu pipeline com uma fonte do Amazon S3

1. Abra o EventBridge console da Amazon em https://console.aws.amazon.com/events/.
2. No painel de navegação, escolha Regras. Deixe o barramento padrão selecionado ou escolha um barramento de eventos. Escolha Criar regra.
3. Em Nome, insira um nome para a regra.
4. Em Tipo de regra, escolha Regra com um padrão de eventos. Escolha Próximo.
5. Em Origem do evento, escolha AWS eventos ou eventos de EventBridge parceiros.
6. Em Tipo de evento de amostra, escolha Eventos do AWS .
7. Em Eventos de amostra, digite S3 como a palavra-chave que servirá como base para a filtragem. Escolha a chamada de AWS API via CloudTrail.
8. Em Método de criação, escolha Padrão personalizado (editor JSON).

Cole o padrão de evento fornecido abaixo. Certifique-se de adicionar o nome do bucket e a chave de objeto do S3 (ou o nome da chave) que identifica o objeto no bucket como `requestParameters`. Neste exemplo, uma regra é criada para um bucket chamado `amzn-s3-demo-source-bucket` e a chave de objeto `my-files.zip`. Quando você usa a janela Edit (Editar) para especificar os recursos, a regra é atualizada para usar um padrão de evento personalizado.

Veja a seguir um exemplo de padrão de evento para copiar e colar:

```
{
    "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. Escolha Próximo.
10. Em Tipos de destino, escolha Serviço da AWS .
11. Em Selecionar um alvo, escolha CodePipeline. Em ARN do pipeline, insira o ARN do pipeline a ser iniciado por esta regra.

> **Note**
>
> Para obter o ARN do pipeline, execute o comando get-pipeline. O ARN do pipeline é exibido na saída. Ele é construído neste formato:
> arn:aws:codepipeline::: *region account pipeline-name*
> Exemplo de ARN do pipeline:
> arn: aws: codepipeline: us-east- 2:80398 EXAMPLE:MyFirstPipeline

12. Para criar ou especificar uma função de serviço do IAM que conceda EventBridge permissões para invocar a meta associada à sua EventBridge regra (nesse caso, a meta é CodePipeline):

    - Escolha Criar uma nova função para esse recurso específico para criar uma função de serviço que dê EventBridge permissões para iniciar suas execuções de pipeline.
    - Escolha Usar função existente para inserir uma função de serviço que dê EventBridge permissões para iniciar suas execuções de pipeline.

13. (Opcional) Para especificar substituições de origem com um ID de imagem específico, use o transformador de entrada para passar os dados como parâmetros JSON. Você também pode usar o transformador de entrada para passar variáveis de pipeline.

    - Expanda Additional settings (Configurações adicionais).

      Em Configurar entrada de destino, escolha Configurar transformador de entrada.

      Na janela de diálogo, escolha Inserir um próprio. Na caixa Caminho de entrada, digite os pares de chave-valor a seguir.

      ```
      {"revisionValue": "$.detail.object.version-id"}
      ```

    - Na caixa Modelo, digite os pares de chave-valor a seguir.

      ```
                                  {
          "sourceRevisions": [
              {
                  "actionName": "Source",
                  "revisionType": "S3_OBJECT_VERSION_ID",
                  "revisionValue": "<revisionValue>"
              }
      ```

```
    ],
     "variables": [
        {
            "name": "Variable_Name",
            "value": "Variable_Value"
        }
    ]
}
```

- Escolha Confirmar.

14. Escolha Próximo.
15. Na página Tags, selecione Próximo.
16. Na página Revisar e criar, revise a configuração da regra. Se você estiver satisfeito com a regra, escolha Create rule.

Crie uma EventBridge regra para uma fonte (CLI) do Amazon S3

Para criar um AWS CloudTrail rastrear e ativar o registro

Para usar o AWS CLI para criar uma trilha, chame o create-trail comando, especificando:

- O nome da trilha.
- O bucket ao qual você já aplicou a política de bucket do AWS CloudTrail.

Para obter mais informações, consulte Criação de uma trilha com a interface de linha de AWS comando.

1. Use o comando create-trail e inclua os parâmetros `--name` e `--s3-bucket-name`.

   Por que estou fazendo essa alteração? Isso cria a trilha do CloudTrail necessária para o bucket de origem do S3.

   O comando a seguir usa `--name` e `--s3-bucket-name` para criar uma trilha denominada `my-trail` e um bucket chamado de `amzn-s3-demo-source-bucket`.

   ```
   aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-source-bucket
   ```

2. Use o comando start-logging e inclua o parâmetro `--name`.

   Por que estou fazendo essa alteração? Esse comando inicia o CloudTrail registro do seu bucket de origem e envia eventos para EventBridge o.

   Exemplo:

   O comando a seguir utiliza `--name` para iniciar o registro em log em uma trilha denominada `my-trail`.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Use o comando put-event-selectors e inclua os parâmetros `--trail-name` e `--event-selectors`. Use seletores de eventos para especificar que você deseja que sua trilha registre eventos de dados para seu bucket de origem e envie os eventos para a EventBridge regra.

   Por que estou fazendo essa alteração? Esse comando filtra eventos.

   Exemplo:

   O comando a seguir utiliza `--trail-name` e `--event-selectors` para especificar eventos de dados para um bucket de origem e prefixo denominado `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"] }] }]'
   ```

Criar uma EventBridge regra com o Amazon S3 como origem e CodePipeline destino do evento e aplicar a política de permissões

1. Conceda permissões EventBridge para usar CodePipeline para invocar a regra. Para obter mais informações, consulte Usando políticas baseadas em recursos para a Amazon. EventBridge

   a. Use o exemplo a seguir para criar a política de confiança que permita assumir EventBridge a função de serviço. Chame-o de `trustpolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Use o comando a seguir para criar a função `Role-for-MyRule` e anexar a política de confiança.

   Por que estou fazendo essa alteração? Adicionar essa política de confiança à função cria permissões para EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Crie o JSON da política de permissões, conforme mostrado aqui para o pipeline chamado `MyFirstPipeline`. Nomeie a politica de permissões `permissionspolicyforEB.json`.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
```

```
            ]
        }
    ]
}
```

d. Use o comando a seguir para anexar a nova política de permissões `CodePipeline-Permissions-Policy-for-EB` à função `Role-for-MyRule` criada.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Use o comando put-rule e inclua os parâmetros `--name`, `--event-pattern` e `--role-arn`.

   O exemplo de comando a seguir cria uma regra chamada `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 adicionar CodePipeline como destino, chame o put-targets comando e inclua `--rule` `--targets` os parâmetros e.

   O comando a seguir especifica que, para a regra denominada `MyS3SourceRule`, o `Id` do destino é composto do número um, indicando que, em uma lista de destinos para a regra, esse é o destino 1. O comando também especifica um ARN de exemplo para o pipeline. O pipeline é iniciado quando uma alteração é feita no repositório.

```
aws events put-targets --rule MyS3SourceRule --targets
 Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

4. (Opcional) Para configurar um transformador de entrada com substituições de origem para um ID de imagem específico, use o JSON a seguir no comando da CLI. O exemplo a seguir configura uma substituição na qual:

   - O `actionName`, `Source` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O `revisionType`, `S3_OBJECT_VERSION_ID` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.

- `OrevisionValue`, < *`revisionValue`* > neste exemplo, é derivado da variável de evento de origem.

```
{
    "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 o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

1. Execute o get-pipeline comando para copiar a estrutura do pipeline em um arquivo JSON. Por exemplo, para um pipeline nomeado `MyFirstPipeline`, execute o seguinte comando:

```
aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
```

Este comando retorna nada, mas o arquivo que você criou deve aparecer no diretório onde você executou o comando.

2. Abra o arquivo JSON em qualquer editor de texto plano e, para editar o estágio de origem, altere o parâmetro `PollForSourceChanges` de um bucket denominado `amzn-s3-demo-source-bucket` para `false`, conforme mostrado neste exemplo.

   Por que estou fazendo essa alteração? A configuração deste parâmetro para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em eventos.

```
"configuration": {
    "S3Bucket": "amzn-s3-demo-source-bucket",
    "PollForSourceChanges": "false",
    "S3ObjectKey": "index.zip"
},
```

3. Se você estiver trabalhando com a estrutura do pipeline recuperada por meio do comando get-pipeline, é necessário remover as linhas `metadata` do arquivo JSON. Caso contrário, o comando update-pipeline não é capaz de utilizá-la. Remova as linhas `"metadata": { }`, `"created"`, `"pipelineARN"` e os campos `"updated"`.

   Por exemplo, remova as seguintes linhas da estrutura:

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

   Salve o arquivo.

4. Para aplicar suas alterações, execute o comando update-pipeline especificando o arquivo JSON do pipeline:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

Este comando retorna toda a estrutura do pipeline editado.

> **Note**
>
> O comando update-pipeline interrompe o pipeline. Se uma revisão estiver sendo executada pelo pipeline quando você executar o comando update-pipeline, essa execução será interrompida. Você deve iniciar manualmente o pipeline para executar a revisão através do pipeline atualizado. Use o comando start-pipeline-execution para iniciar manualmente o pipeline.

Crie uma EventBridge regra para uma fonte do Amazon S3 (CloudFormation modelo)

Para usar CloudFormation para criar uma regra, atualize seu modelo conforme mostrado aqui.

Criar uma EventBridge regra com o Amazon S3 como origem e CodePipeline destino do evento e aplicar a política de permissões

1. No modelo, abaixo`Resources`, use o `AWS::IAM::Role` CloudFormation recurso para configurar a função do IAM que permite que seu evento inicie seu pipeline. Essa entrada cria uma função que utiliza duas políticas:

   - A primeira política permite que a função seja assumida.
   - A segunda política fornece permissões para iniciar o pipeline.

Por que estou fazendo essa alteração? Adicionar `AWS::IAM::Role` recursos permite CloudFormation criar permissões para EventBridge. Esse recurso é adicionado à sua CloudFormation pilha.

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. Use o `AWS::Events::Rule` CloudFormation recurso para adicionar uma EventBridge regra. Esse padrão de evento cria um evento que monitora `CopyObject`, `PutObject` e `CompleteMultipartUpload` no bucket de origem do Amazon S3. Além disso, inclua um destino de seu pipeline. Quando `CopyObject`, `PutObject` ou `CompleteMultipartUpload` ocorrer, essa regra invoca `StartPipelineExecution` em seu pipeline de destino.

   Por que estou fazendo essa alteração? Adicionar o `AWS::Events::Rule` recurso permite CloudFormation criar o evento. Esse recurso é adicionado à sua CloudFormation pilha.

   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. Adicione este trecho ao primeiro modelo para permitir a funcionalidade de pilha 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 um transformador de entrada com substituições de origem para um ID de imagem específico, use o trecho YAML a seguir. O exemplo a seguir configura uma substituição na qual:

   - O `actionName`, `Source` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O `revisionType`, `S3_OBJECT_VERSION_ID` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O`revisionValue`, **<** *`revisionValue`* **>** neste exemplo, é derivado da variável de evento de origem.

```
---
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. Salve o modelo atualizado em seu computador local e abra o CloudFormation console.
6. Selecione sua pilha e clique em Create Change Set for Current Stack (Criar conjunto de alterações para a pilha atual).

7. Carregue o modelo atualizado e, em seguida, visualize as alterações listadas no CloudFormation. Essas são as alterações que serão feitas na pilha. Seus novos recursos devem ser exibidos na lista.
8. Clique em Executar.

Para editar o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

- No modelo, altere `PollForSourceChanges` para `false`. Se você não incluir `PollForSourceChanges` na sua definição de pipeline, adicione-o e configure para `false`.

  Por que estou fazendo essa alteração? A alteração de `PollForSourceChanges` para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em 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 criar um segundo modelo para os recursos do seu pipeline Amazon S3 CloudTrail

- Em um modelo separado, em`Resources`, use os `AWS::CloudTrail::Trail` CloudFormation recursos `AWS::S3::BucketAWS::S3::BucketPolicy`, e para fornecer uma definição simples de bucket e uma trilha para CloudTrail.

  Por que estou fazendo essa alteração? Dado o limite atual de cinco trilhas por conta, a CloudTrail trilha deve ser criada e gerenciada separadamente. (Consulte Limites em AWS CloudTrail.) No entanto, você pode incluir vários buckets do Amazon S3 em uma única trilha,

para que possa criar a trilha uma vez e adicionar buckets do Amazon S3 para outros pipelines, conforme for necessário. Cole o seguinte no arquivo do segundo modelo de exemplo.

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 ações de origem e EventBridge

Para adicionar uma ação de CodeCommit origem em CodePipeline, você pode escolher entre:

- Use o assistente de criação de pipeline do CodePipeline console (Criar um pipeline personalizado (console)) ou a página Editar ação para escolher a opção CodeCommit de provedor. O console cria uma EventBridge regra que inicia seu pipeline quando a fonte muda.
- Use o AWS CLI para adicionar a configuração da `CodeCommit` ação e criar recursos adicionais da seguinte forma:
  - Use o exemplo de configuração de ação `CodeCommit` em CodeCommit referência de ação de origem para criar sua ação, conforme mostrado em Criar um pipeline (CLI).
  - O método de detecção de alterações assume como padrão iniciar o pipeline por meio da sondagem da origem. Você deve desabilitar as verificações periódicas e criar a regra de detecção de alterações manualmente. Use um dos seguintes métodos: Crie uma EventBridge regra para uma CodeCommit fonte (console), Crie uma EventBridge regra para uma CodeCommit fonte (CLI) ou Crie uma EventBridge regra para uma CodeCommit fonte (CloudFormation modelo).

Tópicos

- Crie uma EventBridge regra para uma CodeCommit fonte (console)
- Crie uma EventBridge regra para uma CodeCommit fonte (CLI)
- Crie uma EventBridge regra para uma CodeCommit fonte (CloudFormation modelo)

Crie uma EventBridge regra para uma CodeCommit fonte (console)

> **Important**
>
> Se você usar o console para criar ou editar seu funil, sua EventBridge regra será criada para você.

Para criar uma EventBridge regra para uso em CodePipeline operações

1. Abra o EventBridge console da Amazon em https://console.aws.amazon.com/events/.
2. No painel de navegação, escolha Regras. Deixe o barramento padrão selecionado ou escolha um barramento de eventos. Escolha Criar regra.
3. Em Nome, insira um nome para a regra.
4. Em Tipo de regra, escolha Regra com um padrão de eventos. Escolha Próximo.
5. Em Origem do evento, escolha AWS eventos ou eventos de EventBridge parceiros.
6. Em Tipo de evento de amostra, escolha Eventos do AWS .
7. Em Exemplos de eventos, digite CodeCommit como a palavra-chave a ser filtrada. Escolha Alteração do estado do CodeCommit repositório.
8. Em Método de criação, escolha Padrão personalizado (editor JSON).

   Cole o padrão de evento fornecido abaixo. Veja a seguir um exemplo de padrão de eventos do CodeCommit na janela Event (Evento) para um repositório `MyTestRepo` com uma ramificação 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. Em Metas, escolha CodePipeline.
10. Insira o ARN do pipeline a ser iniciado por esta regra.

> **Note**
>
> Você pode encontrar o ARN do pipeline na saída de metadados após executar o comando get-pipeline. O ARN do pipeline é construído neste formato: arn:aws:codepipeline::: *region account pipeline-name*
> Exemplo de ARN do pipeline:
> `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

11. Para criar ou especificar uma função de serviço do IAM que conceda EventBridge permissões para invocar a meta associada à sua EventBridge regra (nesse caso, a meta é CodePipeline):

    - Escolha Criar uma nova função para esse recurso específico para criar uma função de serviço que dê EventBridge permissões para iniciar suas execuções de pipeline.
    - Escolha Usar função existente para inserir uma função de serviço que dê EventBridge permissões para iniciar suas execuções de pipeline.

12. (Opcional) Para especificar substituições de origem com um ID de imagem específico, use o transformador de entrada para passar os dados como parâmetros JSON. Você também pode usar o transformador de entrada para passar variáveis de pipeline.

    - Expanda Additional settings (Configurações adicionais).

      Em Configurar entrada de destino, escolha Configurar transformador de entrada.

Na janela de diálogo, escolha Inserir um próprio. Na caixa Caminho de entrada, digite os pares de chave-valor a seguir.

```
{"revisionValue": "$.detail.image-digest"}
```

- Na caixa Modelo, digite os pares de chave-valor a seguir.

```
{
    "sourceRevisions": [
        {
            "actionName": "Source",
            "revisionType": "COMMIT_ID",
            "revisionValue": "<revisionValue>"
        }
    ],
    "variables": [
        {
            "name": "Branch_Name",
            "value": "value"
        }
    ]
}
```

- Escolha Confirmar.

13. Escolha Próximo.
14. Na página Tags, selecione Próximo.
15. Na página Revisar e criar, revise a configuração da regra. Se você estiver satisfeito com a regra, escolha Create rule.

Crie uma EventBridge regra para uma CodeCommit fonte (CLI)

Use o comando put-rule, especificando:

- Um nome que identifique de forma exclusiva a regra que você está criando. Esse nome deve ser exclusivo em todos os funis que você cria CodePipeline associados à sua AWS conta.
- O padrão de evento para a origem e os campos detalhados usados pela regra. Para obter mais informações, consulte Amazon EventBridge e Event Patterns.

Para criar uma EventBridge regra com CodeCommit como origem do evento e CodePipeline como destino

1. Adicione permissões para usar EventBridge CodePipeline para invocar a regra. Para obter mais informações, consulte Usando políticas baseadas em recursos para a Amazon. EventBridge

   a. Use o exemplo a seguir para criar a política de confiança que EventBridge permite assumir a função de serviço. Nomeie a política de confiança `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Use o comando a seguir para criar a função `Role-for-MyRule` e anexar a política de confiança.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Crie o JSON de política de permissões, conforme mostrado neste exemplo, para o pipeline denominado `MyFirstPipeline`. Nomeie a politica de permissões `permissionspolicyforEB.json`.

      JSON

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

```
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Use o comando a seguir para anexar a política de permissões `CodePipeline-Permissions-Policy-for-EB` à função `Role-for-MyRule`.

   Por que estou fazendo essa alteração? Adicionar essa política à função cria permissões 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. Use o comando put-rule e inclua os parâmetros `--name`, `--event-pattern` e `--role-arn`.

   Por que estou fazendo essa alteração? Esse comando permite que o CloudFormation crie o evento.

   O comando de exemplo a seguir cria uma regra chamada `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 adicionar CodePipeline como destino, chame o put-targets comando e inclua os seguintes parâmetros:

   - O parâmetro `--rule` é usado com `rule_name` criado por meio de put-rule.
   - O parâmetro `--targets` é usado com o `Id` da lista do destino na lista de destinos e o ARN do pipeline de destino.

O exemplo de comando a seguir especifica que, para a regra chamada `MyCodeCommitRepoRule`, o `Id` do destino é composto do número um, indicando que, em uma lista de destinos para a regra, esse é o destino 1. O exemplo de comando também especifica um exemplo ARN para o pipeline. O pipeline é iniciado quando uma alteração é feita no repositório.

```
aws events put-targets --rule MyCodeCommitRepoRule --targets
 Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

4. (Opcional) Para configurar um transformador de entrada com substituições de origem para um ID de imagem específico, use o JSON a seguir no comando da CLI. O exemplo a seguir configura uma substituição na qual:

   - O `actionName`, `Source` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O `revisionType`, `COMMIT_ID` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
   - O`revisionValue`, **<** *`revisionValue`* **>** neste exemplo, é derivado da variável de evento de origem.

```
{
    "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 o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Ao criar um pipeline com esse método, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro se não for explicitamente definido como falso. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

1. Execute o get-pipeline comando para copiar a estrutura do pipeline em um arquivo JSON. Por exemplo, para um pipeline nomeado `MyFirstPipeline`, execute o seguinte comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando retorna nada, mas o arquivo que você criou deve aparecer no diretório onde você executou o comando.

2. Abra o arquivo JSON em qualquer editor de texto plano e altere o `PollForSourceChanges` parâmetro `false` para editar o estágio de origem, como mostrado no exemplo a seguir.

   Por que estou fazendo essa alteração? A alteração deste parâmetro para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em eventos.

   ```
   "configuration": {
       "PollForSourceChanges": "false",
       "BranchName": "main",
       "RepositoryName": "MyTestRepo"
   },
   ```

3. Se você estiver trabalhando com a estrutura do pipeline recuperada por meio do comando get-pipeline, remova as linhas `metadata` do arquivo JSON. Caso contrário, o comando

update-pipeline não é capaz de utilizá-la. Remova as linhas `"metadata": { }`, `"created"`, `"pipelineARN"` e os campos `"updated"`.

Por exemplo, remova as seguintes linhas da estrutura:

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

Salve o arquivo.

4. Para aplicar suas alterações, execute o comando update-pipeline especificando o arquivo JSON do pipeline:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

Este comando retorna toda a estrutura do pipeline editado.

> **Note**
>
> O comando update-pipeline interrompe o pipeline. Se uma revisão estiver sendo executada pelo pipeline quando você executar o comando update-pipeline, essa execução será interrompida. Você deve iniciar manualmente o pipeline para executar a revisão através do pipeline atualizado. Use o comando **`start-pipeline-execution`** para iniciar manualmente o pipeline.

Crie uma EventBridge regra para uma CodeCommit fonte (CloudFormation modelo)

Para usar CloudFormation para criar uma regra, atualize seu modelo conforme mostrado aqui.

Para atualizar seu funil CloudFormation modelo e EventBridge regra de criação

1. No modelo, abaixo`Resources`, use o `AWS::IAM::Role` CloudFormation recurso para configurar a função do IAM que permite que seu evento inicie seu pipeline. Essa entrada cria uma função que utiliza duas políticas:

   - A primeira política permite que a função seja assumida.
   - A segunda política fornece permissões para iniciar o pipeline.

   Por que estou fazendo essa alteração? Adicionar o `AWS::IAM::Role` recurso permite CloudFormation criar permissões para EventBridge. Esse recurso é adicionado à sua CloudFormation pilha.

   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. No modelo, em`Resources`, use o `AWS::Events::Rule` CloudFormation recurso para adicionar uma EventBridge regra. Esse padrão de evento cria um evento que monitora as alterações por push no seu repositório. Quando EventBridge detecta uma alteração no estado do repositório, a regra é invocada `StartPipelineExecution` em seu pipeline de destino.

   Por que estou fazendo essa alteração? Adicionar o `AWS::Events::Rule` recurso permite CloudFormation criar o evento. Esse recurso é adicionado à sua CloudFormation pilha.

   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 um transformador de entrada com substituições de origem para um ID de imagem específico, use o trecho YAML a seguir. O exemplo a seguir configura uma substituição na qual:

   - O `actionName`, `Source` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.

- O `revisionType`, `COMMIT_ID` neste exemplo, é o valor dinâmico, definido na criação do pipeline, não derivado do evento de origem.
- O`revisionValue`, **<** *`revisionValue`* **>** neste exemplo, é derivado da variável de evento de origem.
- As variáveis de saída para `BranchName` e `Value` são especificadas.

```
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
    sourceRevisions:
      actionName: Source
      revisionType: COMMIT_ID
      revisionValue: <revisionValue>
    variables:
    - name: BranchName
      value: value
```

4. Salve o modelo atualizado em seu computador local e abra o CloudFormation console.
5. Selecione sua pilha e clique em Create Change Set for Current Stack (Criar conjunto de alterações para a pilha atual).
6. Carregue o modelo e visualize as alterações listadas no CloudFormation. Essas são as alterações a serem feitas na pilha. Seus novos recursos devem ser exibidos na lista.
7. Clique em Executar.

Para editar o PollForSourceChanges parâmetro do seu funil

> **Important**
>
> Em muitos casos, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro ao criar um pipeline. Ao adicionar a detecção de alterações baseada em eventos, é necessário adicionar o parâmetro a sua saída e defini-lo como falso para desativar a sondagem. Caso contrário, o pipeline inicia duas vezes para uma única alteração

de origem. Para obter detalhes, consulte Configurações válidas para o parâmetro `PollForSourceChanges`.

- No modelo, altere `PollForSourceChanges` para `false`. Se você não incluir `PollForSourceChanges` na sua definição de pipeline, adicione-o e configure para `false`.

  Por que estou fazendo essa alteração? A alteração deste parâmetro para `false` desativa as verificações periódicas para que você possa utilizar apenas a detecção de alterações baseada em 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
    }
  ]
},
```

Adicione provedores de origem terceirizados aos pipelines usando AWS CodeConnections

Você pode usar o AWS CodePipeline console ou o AWS CLI para conectar seu pipeline a repositórios de terceiros.

> **Note**
>
> Quando o console é usado para criar ou editar um pipeline, os recursos de detecção de alterações são criados para você. Se você usar a AWS CLI para criar o pipeline, deverá criar você mesmo os recursos adicionais. Para obter mais informações, consulte CodeCommit ações de origem e EventBridge.

Tópicos

- DevOps Conexões Azure
- Conexões do Bitbucket Cloud
- GitHub conexões
- GitHub Conexões do Enterprise Server
- GitLab.com conexões
- Conexões para GitLab autogerenciamento
- Usar uma conexão compartilhada com outra conta

DevOps Conexões Azure

As conexões permitem que você autorize e estabeleça configurações que associem seu provedor terceirizado aos seus AWS recursos. Para associar seu repositório de terceiros como origem do pipeline, use uma conexão.

> **Note**
>
> Em vez de criar ou usar uma conexão existente na conta, você pode usar uma conexão compartilhada entre outra Conta da AWS. Consulte Usar uma conexão compartilhada com outra conta.

> **Note**
>
> Esse recurso não está disponível nas regiões Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Melbourne), Ásia-Pacífico (Osaka), África (Cidade do Cabo), Oriente Médio (Bahrein), Oriente Médio (EAU), Europa (Espanha), Europa (Zurique), Israel (Tel Aviv) ou (). AWS GovCloud US-West Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline. Para considerações sobre essa ação na região Europa (Milão), consulte a nota em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

Para adicionar uma ação DevOps de origem do Azure CodePipeline, você pode escolher entre:

- Use o assistente de criação de pipeline do CodePipeline console ou a página Editar ação para escolher a opção de DevOps provedor do Azure. Consulte Criar uma conexão com o Bitbucket Cloud (console) para adicionar a ação. O console ajuda você a criar um recurso de conexão.
- Usar a CLI para adicionar a configuração da ação `CreateSourceConnection` com o provedor `Azure DevOps`:
  - Para criar seus recursos de conexão, consulte Crie uma conexão com o Azure DevOps (CLI) para criar um recurso de conexão com a CLI.
  - Use o exemplo de configuração da ação `CreateSourceConnection` em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com para adicionar sua ação, conforme mostrado em Criar um pipeline (CLI).

> **Note**
>
> É possível criar uma conexão por meio do console do Developer Tools em Configurações. Consulte Criar uma conexão.

Antes de começar

- Você deve ter criado uma conta com o provedor do repositório de terceiros, como o Azure DevOps

- Você já deve ter criado um repositório de código de terceiros, como um DevOps repositório do Azure.

> **Note**
>
> DevOps As conexões do Azure fornecem acesso somente aos repositórios pertencentes à DevOps conta do Azure que foi usada para criar a conexão.
> Para instalar a conexão, é necessário ter permissões Administrador para a conta do Azure.

Tópicos

- Criar uma conexão com o Azure DevOps (console)
- Crie uma conexão com o Azure DevOps (CLI)

Criar uma conexão com o Azure DevOps (console)

Use essas etapas para usar o CodePipeline console para adicionar uma ação de conexões ao seu DevOps repositório do Azure.

Etapa 1: Criar ou editar seu pipeline

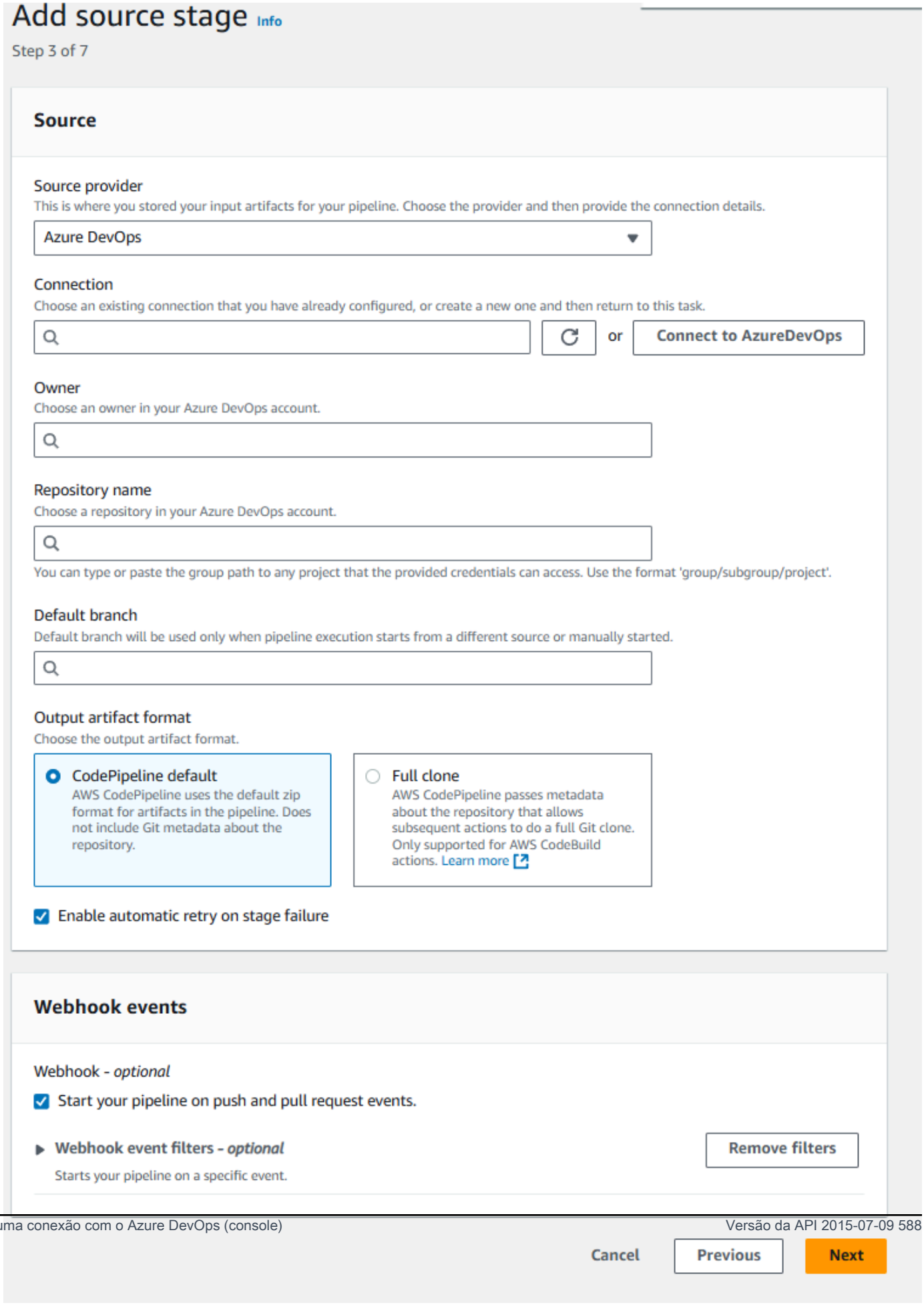
Para criar ou editar seu pipeline

1. Faça login no CodePipeline console.
2. Escolha uma das opções a seguir.
   - Opte por criar um pipeline. Siga as etapas em Criar um pipeline para concluir a primeira tela e escolha Próximo. Na página Fonte, em Provedor de Origem, escolha Azure DevOps.
   - Opte por editar um pipeline existente. Escolha Editar e, em seguida, escolha Editar estágio. Escolha adicionar ou editar sua ação de origem. Na página Editar ação, em Nome da ação, insira um nome para a ação. Em Provedor de ação, escolha Azure DevOps.
3. Execute um destes procedimentos:
   - Em Conexão, se você ainda não tiver criado uma conexão com seu provedor, escolha Conectar ao Azure DevOps. Vá para a Etapa 2: Criar uma conexão com o Azure DevOps.
   - Em Conexão, se você já tiver criado uma conexão com seu provedor, escolha a conexão. Vá para a Etapa 3: Salvar a ação de origem para sua conexão.

Etapa 2: criar uma conexão com o Azure DevOps

Para criar uma conexão com o Azure DevOps

1. Para criar uma conexão com um DevOps repositório do Azure, em Selecionar um provedor, escolha Azure DevOps. Em Connection name (Nome da conexão), digite o nome da conexão que você deseja criar. Escolha Connect to Azure DevOps e vá para a Etapa 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. Na página de DevOps configurações do Connect to Azure, o nome da sua conexão é exibido.

   Em DevOps Aplicativos do Azure, escolha uma instalação de aplicativo ou escolha Instalar um novo aplicativo para criar um.

   > **Note**
   >
   > Você só instala o aplicativo uma vez para cada DevOps conta do Azure. Se você já tiver instalado a aplicação do conector, escolha-a e vá até a última etapa nesta seção.

3. Se a página de login da Microsoft for exibida, faça login usando as credenciais e opte por continuar.
4. Na página de instalação do aplicativo, uma mensagem mostra que o aplicativo conector está tentando se conectar à sua DevOps conta do Azure.

   Escolha Conceder acesso.
5. Na página de conexão, o ID de conexão da nova instalação é exibido. Selecione Conectar. A conexão criada é exibida na lista de conexões.

Etapa 3: Salve sua ação de DevOps origem do Azure

Execute estas etapas no assistente ou na página Editar ação para salvar a ação de origem com as informações de conexão.

Para concluir e salvar a ação de origem com a conexão

1. Em Repository name (Nome do repositório), escolha o nome do repositório de terceiros.
2. Em Gatilhos do Pipeline, você pode adicionar gatilhos se sua ação for uma ação. AWS CodeConnections Para definir a configuração de gatilhos do pipeline e, se necessário, aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request.
3. Em Output artifact format (Formato de artefato de saída), você deve escolher o formato para seus artefatos.

   - Para armazenar artefatos de saída da DevOps ação do Azure usando o método padrão, escolha CodePipeline default. A ação acessa os arquivos no repositório e armazena os artefatos em um arquivo ZIP no armazenamento de artefatos do pipeline.

- Para armazenar um arquivo JSON que contém uma referência de URL ao repositório para que as ações downstream possam executar comandos Git diretamente, escolha Full clone (Clone completo). Essa opção só pode ser usada por ações CodeBuild posteriores.

  Se você escolher essa opção, precisará atualizar as permissões para sua função de serviço CodeBuild do projeto, conforme mostrado em[Adicione CodeBuild GitClone permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com](#).

4. Escolha Próximo no assistente ou Salvar na página Editar ação.

Crie uma conexão com o Azure DevOps (CLI)

Você pode usar o AWS Command Line Interface (AWS CLI) para criar uma conexão.

Para fazer isso, use o comando create-connection.

> **Important**
>
> Uma conexão criada por meio do AWS CLI ou AWS CloudFormation está no `PENDING` status por padrão. Depois de criar uma conexão com a CLI ou CloudFormation, use o console para editar a conexão e definir seu status. `AVAILABLE`

Para criar uma conexão

1. Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows). Use o AWS CLI para executar o create-connection comando, especificando `--provider-type` e `--connection-name` para sua conexão. Neste exemplo, o nome do provedor de terceiros é `AzureDevOps` e o nome da conexão especificada é `MyConnection`.

   ```
   aws codeconnections create-connection --provider-type AzureDevOps --connection-name
    MyConnection
   ```

   Se tiver êxito, esse comando gerará as informações do ARN de conexão semelhantes às seguintes.

   ```
   {
       "ConnectionArn": "arn:aws:codeconnections:us-west-2:account_id:connection/
   aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
   ```

```
}
```

2. Use o console para concluir a conexão. Para obter mais informações, consulte Atualizar uma conexão pendente.
3. O pipeline assume como padrão a detecção de alterações ao enviar o código por push ao repositório de origem da conexão. Para definir a configuração do gatilho do pipeline para liberação manual ou para tags Git, execute um dos seguintes procedimentos:

   - Para definir a configuração do gatilho do pipeline para início somente por meio de liberação manual, adicione a seguinte linha à configuração:

     ```
     "DetectChanges": "false",
     ```

   - Para definir a configuração de gatilhos do pipeline e aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request. Por exemplo, o trecho a seguir adiciona etiquetas Git ao nível do pipeline dentro da definição JSON do pipeline. Neste exemplo, `release-v0` e `release-v1` são as tags Git a serem incluídas, enquanto `release-v2` são as tags Git a serem excluídas.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
     ```

Conexões do Bitbucket Cloud

As conexões permitem que você autorize e estabeleça configurações que associem seu provedor terceirizado aos seus AWS recursos. Para associar seu repositório de terceiros como origem do pipeline, use uma conexão.

> **Note**
>
> Em vez de criar ou usar uma conexão existente na conta, você pode usar uma conexão compartilhada entre outra Conta da AWS. Consulte Usar uma conexão compartilhada com outra conta.

> **Note**
>
> Esse recurso não está disponível nas regiões Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Melbourne), Ásia-Pacífico (Osaka), África (Cidade do Cabo), Oriente Médio (Bahrein), Oriente Médio (EAU), Europa (Espanha), Europa (Zurique), Israel (Tel Aviv) ou (). AWS GovCloud US-West Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline. Para considerações sobre essa ação na região Europa (Milão), consulte a nota em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

Para adicionar uma ação de origem do Bitbucket Cloud CodePipeline, você pode escolher entre:

- Use o assistente de criação de pipeline do CodePipeline console ou a página Editar ação para escolher a opção de provedor do Bitbucket. Consulte Criar uma conexão com o Bitbucket Cloud (console) para adicionar a ação. O console ajuda você a criar um recurso de conexão.

  > **Note**
  >
  > Você pode criar conexões para um repositório do Bitbucket Cloud. Não há suporte a tipos de provedores instalados do Bitbucket, como o Bitbucket Server.

- Usar a CLI para adicionar a configuração da ação `CreateSourceConnection` com o provedor `Bitbucket`:

- Para criar seus recursos de conexão, consulte Criar uma conexão com o Bitbucket Cloud (CLI) para criar um recurso de conexão com a CLI.
- Use o exemplo de configuração da ação `CreateSourceConnection` em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com para adicionar sua ação, conforme mostrado em Criar um pipeline (CLI).

> **Note**
>
> É possível criar uma conexão por meio do console do Developer Tools em Configurações. Consulte Criar uma conexão.

Antes de começar

- Você deve ter criado uma conta com o provedor do repositório de terceiros, como o Bitbucket Cloud.
- Você já deve ter criado um repositório de código de terceiros, como um repositório do Bitbucket Cloud.

> **Note**
>
> As conexões do Bitbucket Cloud fornecem acesso somente aos repositórios pertencentes à conta do Bitbucket Cloud usada para criar a conexão.
> Se a aplicação estiver sendo instalada em um espaço de trabalho do Bitbucket Cloud, você precisará de permissões para Administrar o espaço de trabalho. Caso contrário, a opção de instalar a aplicação não será exibida.

Tópicos

- Criar uma conexão com o Bitbucket Cloud (console)
- Criar uma conexão com o Bitbucket Cloud (CLI)

Criar uma conexão com o Bitbucket Cloud (console)

Use essas etapas para usar o CodePipeline console para adicionar uma ação de conexões ao seu repositório Bitbucket.

> **Note**
>
> Você pode criar conexões para um repositório do Bitbucket Cloud. Não há suporte a tipos de provedores instalados do Bitbucket, como o Bitbucket Server.

Etapa 1: Criar ou editar seu pipeline

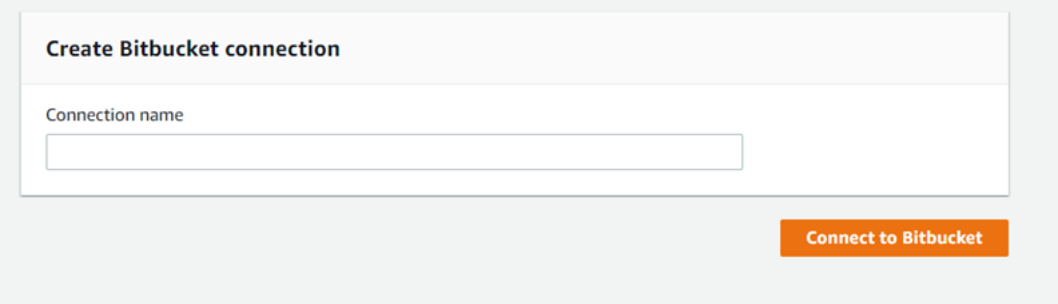
Para criar ou editar seu pipeline

1. Faça login no CodePipeline console.
2. Escolha uma das opções a seguir.
   - Opte por criar um pipeline. Siga as etapas em Criar um pipeline para concluir a primeira tela e escolha Próximo. Na página Origem, em Provedor de origem, escolha Bitbucket.
   - Opte por editar um pipeline existente. Escolha Editar e, em seguida, escolha Editar estágio. Escolha adicionar ou editar sua ação de origem. Na página Editar ação, em Nome da ação, insira um nome para a ação. Em Provedor de ação, selecione Bitbucket.
3. Execute um destes procedimentos:
   - Em Conexão, se você ainda não criou uma conexão com seu provedor, escolha Conectar ao Bitbucket. Vá para a Etapa 2: Criar uma conexão com o Bitbucket.
   - Em Conexão, se você ainda não criou uma conexão com seu provedor, escolha a conexão. Vá para a Etapa 3: Salve a ação de origem para sua conexão.

Etapa 2: Criar uma conexão com o Bitbucket Cloud

Para criar uma conexão com o Bitbucket Cloud

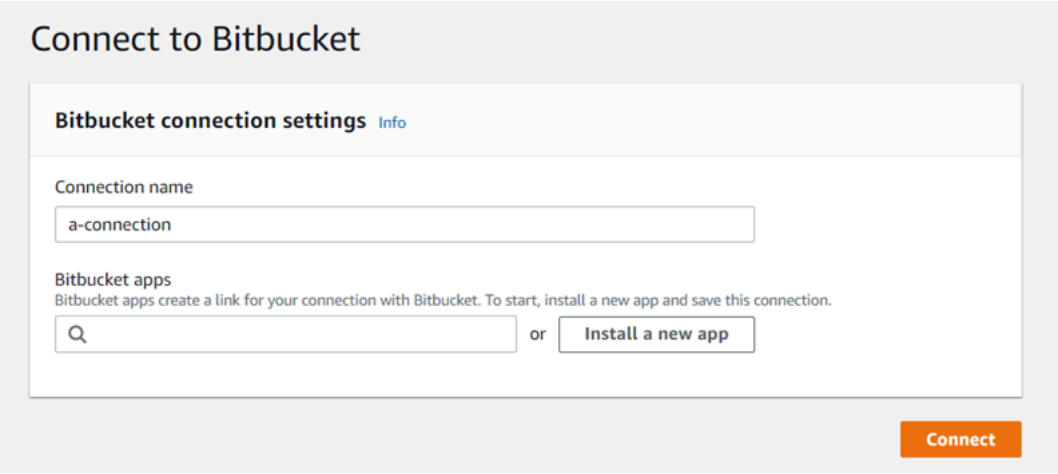
1. Na página de configurações Conectar ao Bitbucket, insira o nome da sua conexão e escolha Conectar ao Bitbucket.



O campo Aplicativos Bitbucket é exibido.

2. Em Bitbucket apps (Aplicações do Bitbucket), escolha uma instalação de aplicação ou Install a new app (Instalar uma nova aplicação) para criar uma.

> **Note**
>
> O aplicativo é instalado apenas uma vez para cada espaço de trabalho ou conta do Bitbucket. Se você já instalou o aplicativo Bitbucket, escolha-o e vá para a etapa 4.

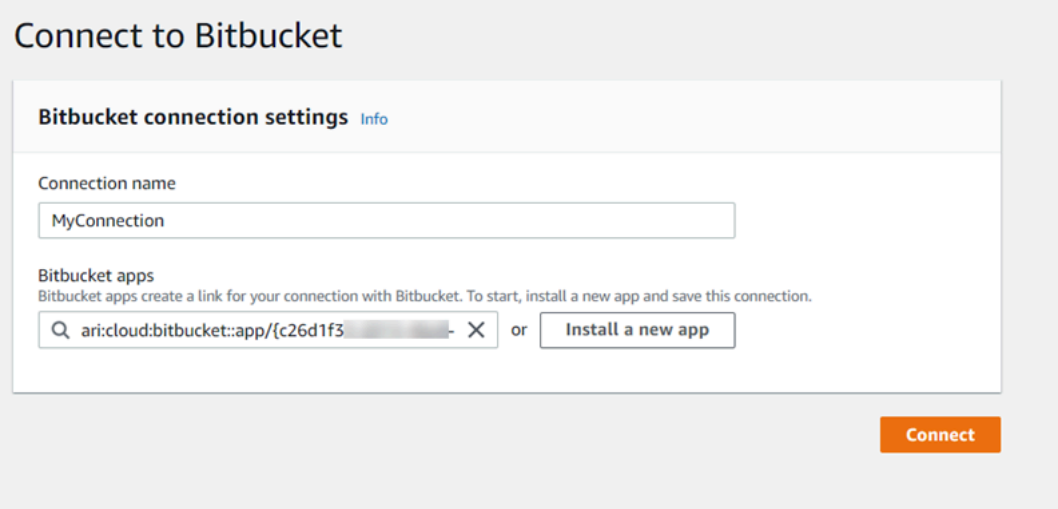


3. Se a página de login do Bitbucket Cloud for exibida, faça login com suas credenciais e escolha a opção de continuar.
4. Na página de instalação do aplicativo, uma mensagem mostra que o AWS CodeStar aplicativo está tentando se conectar à sua conta do Bitbucket.

   Se você estiver usando um espaço de trabalho do Bitbucket, altere a opção Authorize for (Autorizar para) do espaço de trabalho. Somente os espaços de trabalho nos quais você tem acesso de administrador serão exibidos.

Escolha Conceder acesso.

5. Em Bitbucket apps (Aplicações do Bitbucket), o ID de conexão para a nova instalação é exibido. Selecione Conectar. A conexão criada é exibida na lista de conexões.



Etapa 3: Salvar a ação de origem do Bitbucket Cloud

Execute estas etapas no assistente ou na página Editar ação para salvar a ação de origem com as informações de conexão.

Para concluir e salvar a ação de origem com a conexão

1. Em Repository name (Nome do repositório), escolha o nome do repositório de terceiros.
2. Em Gatilhos do Pipeline, você pode adicionar gatilhos se sua ação for uma ação. AWS CodeConnections Para definir a configuração de gatilhos do pipeline e, se necessário, aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request.
3. Em Output artifact format (Formato de artefato de saída), você deve escolher o formato para seus artefatos.

   - Para armazenar artefatos de saída da ação do Bitbucket Cloud usando o método padrão, escolha CodePipeline default. A ação acessa os arquivos do repositório do Bitbucket Cloud e armazena os artefatos em um arquivo ZIP no armazenamento de artefatos do pipeline.
   - Para armazenar um arquivo JSON que contém uma referência de URL ao repositório para que as ações downstream possam executar comandos Git diretamente, escolha Full clone (Clone completo). Essa opção só pode ser usada por ações CodeBuild posteriores.

Se você escolher essa opção, precisará atualizar as permissões para sua função de serviço CodeBuild do projeto, conforme mostrado emAdicione CodeBuild GitClone permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com.

4. Escolha Próximo no assistente ou Salvar na página Editar ação.

Criar uma conexão com o Bitbucket Cloud (CLI)

Você pode usar o AWS Command Line Interface (AWS CLI) para criar uma conexão.

> **Note**
>
> Você pode criar conexões para um repositório do Bitbucket Cloud. Não há suporte a tipos de provedores instalados do Bitbucket, como o Bitbucket Server.

Para fazer isso, use o comando create-connection.

> **Important**
>
> Uma conexão criada por meio do AWS CLI ou AWS CloudFormation está no `PENDING` status por padrão. Depois de criar uma conexão com a CLI ou CloudFormation, use o console para editar a conexão e definir seu status. `AVAILABLE`

Para criar uma conexão

1. Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows). Use o AWS CLI para executar o create-connection comando, especificando `--provider-type` e `--connection-name` para sua conexão. Neste exemplo, o nome do provedor de terceiros é `Bitbucket` e o nome da conexão especificada é `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type Bitbucket --connection-name MyConnection
   ```

   Se tiver êxito, esse comando gerará as informações do ARN de conexão semelhantes às seguintes.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Use o console para concluir a conexão. Para obter mais informações, consulte Atualizar uma conexão pendente.
3. O pipeline assume como padrão a detecção de alterações ao enviar o código por push ao repositório de origem da conexão. Para definir a configuração do gatilho do pipeline para liberação manual ou para tags Git, execute um dos seguintes procedimentos:

   - Para definir a configuração do gatilho do pipeline para início somente por meio de liberação manual, adicione a seguinte linha à configuração:

     ```
     "DetectChanges": "false",
     ```

   - Para definir a configuração de gatilhos do pipeline e aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request. Por exemplo, o trecho a seguir adiciona etiquetas Git ao nível do pipeline dentro da definição JSON do pipeline. Neste exemplo, `release-v0` e `release-v1` são as tags Git a serem incluídas, enquanto `release-v2` são as tags Git a serem excluídas.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
                            }
                        ]
                    }
     ```

```
            }
        ]
```

GitHub conexões

Você usa conexões para autorizar e estabelecer configurações que associam seu provedor terceirizado aos seus AWS recursos.

> **Note**
>
> Em vez de criar ou usar uma conexão existente na conta, você pode usar uma conexão compartilhada entre outra Conta da AWS. Consulte Usar uma conexão compartilhada com outra conta.

> **Note**
>
> Esse recurso não está disponível nas regiões Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Melbourne), Ásia-Pacífico (Osaka), África (Cidade do Cabo), Oriente Médio (Bahrein), Oriente Médio (EAU), Europa (Espanha), Europa (Zurique), Israel (Tel Aviv) ou (). AWS GovCloud US-West Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline. Para considerações sobre essa ação na região Europa (Milão), consulte a nota em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

Para adicionar uma ação de origem para seu repositório GitHub ou para o seu repositório do GitHub Enterprise Cloud CodePipeline, você pode escolher entre:

- Use o assistente de criação de pipeline do CodePipeline console ou a página Editar ação para escolher a opção de provedor GitHub (via GitHub aplicativo). Consulte Crie uma conexão com GitHub (console) para adicionar a ação. O console ajuda você a criar um recurso de conexão.

> **Note**
>
> Para ver um tutorial que explica como adicionar uma GitHub conexão e usar a opção de clonagem completa em seu pipeline para clonar metadados, consulte. Tutorial: use o clone completo com uma fonte de GitHub pipeline

- Usar a CLI para adicionar a configuração da ação `CodeStarSourceConnection` com o provedor `GitHub` executando as etapas da CLI em Criar um pipeline (CLI).

> **Note**
>
> É possível criar uma conexão por meio do console do Developer Tools em Configurações. Consulte Criar uma conexão.

Antes de começar

- Você deve ter criado uma conta com GitHub.
- Você já deve ter criado um repositório GitHub de código.
- Se sua função CodePipeline de serviço foi criada antes de 18 de dezembro de 2019, talvez seja necessário atualizar suas permissões `codestar-connections:UseConnection` para uso em AWS CodeStar conexões. Para instruções, consulte Adicionar permissões à função CodePipeline de serviço.

> **Note**
>
> Para criar a conexão, você deve ser o proprietário GitHub da organização. Para repositórios que não estão em uma organização, você deve ser o proprietário do repositório.

Tópicos

- Crie uma conexão com GitHub (console)
- Crie uma conexão com GitHub (CLI)

Crie uma conexão com GitHub (console)

Use essas etapas para usar o CodePipeline console para adicionar uma ação de conexões para seu repositório GitHub ou para o seu repositório do GitHub Enterprise Cloud.

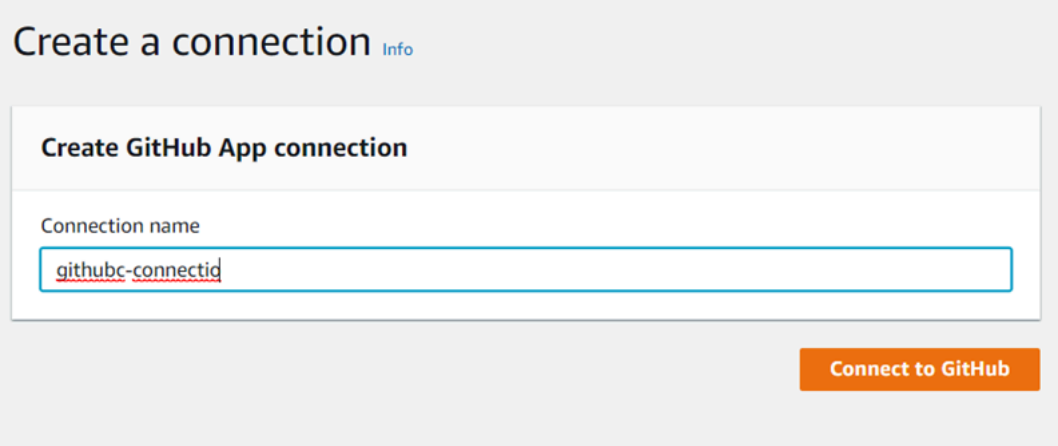
> **Note**
>
> Nestas etapas, você pode selecionar repositórios específicos em Acesso ao repositório. Quaisquer repositórios que não estejam selecionados não estarão acessíveis ou visíveis pelo CodePipeline.

Etapa 1: Criar ou editar seu pipeline

1. Faça login no CodePipeline console.
2. Escolha uma das opções a seguir.
   - Opte por criar um pipeline. Siga as etapas em Criar um pipeline para concluir a primeira tela e escolha Próximo. Na página Fonte, em Provedor de origem, escolha GitHub (via GitHub aplicativo).
   - Opte por editar um pipeline existente. Escolha Editar e, em seguida, escolha Editar estágio. Escolha adicionar ou editar sua ação de origem. Na página Editar ação, em Nome da ação, insira um nome para a ação. Em Provedor de ações, escolha GitHub (via GitHub aplicativo).
3. Execute um destes procedimentos:
   - Em Conexão, se você ainda não tiver criado uma conexão com seu provedor, escolha Conectar GitHub a. Prossiga para a Etapa 2: Crie uma conexão com GitHub.
   - Em Conexão, se você já tiver criado uma conexão com seu provedor, escolha a conexão. Vá para a Etapa 3: Salvar a ação de origem da sua conexão.

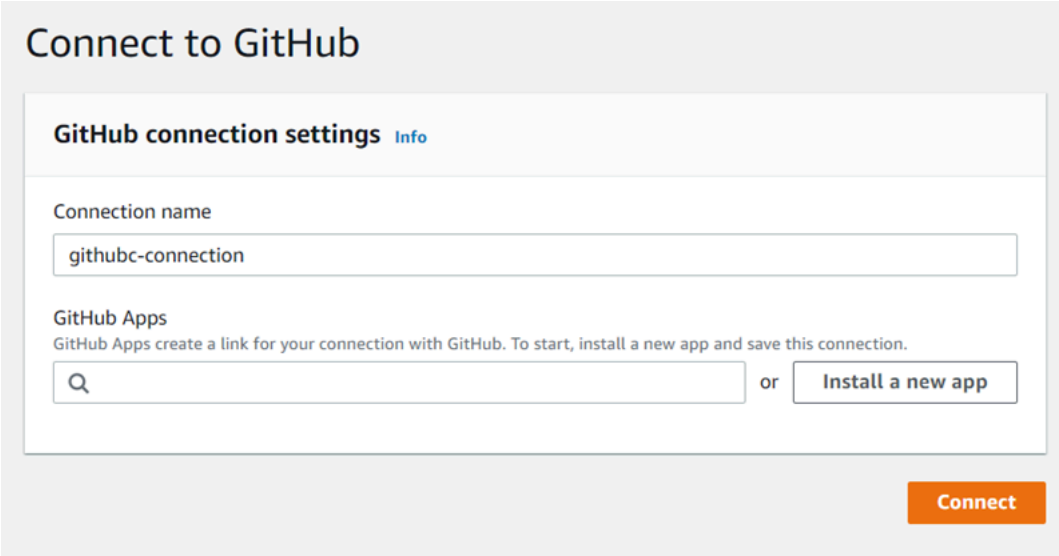
Etapa 2: criar uma conexão com GitHub

Depois de escolher criar a conexão, a GitHub página Connect to é exibida.



Para criar uma conexão com GitHub

1. Nas configurações de GitHub conexão, o nome da conexão aparece em Nome da conexão. Escolha Connect to GitHub. A página de solicitação de acesso será exibida.
2. Escolha Autorizar AWS conector para GitHub. A página de conexão é exibida e mostra o campo GitHub Aplicativos.



3. Em GitHub Aplicativos, escolha uma instalação de aplicativo ou escolha Instalar um novo aplicativo para criar um.

   Você instala uma aplicação para todas as suas conexões com um provedor específico. Se você já instalou o GitHub aplicativo AWS Connector for, escolha-o e pule esta etapa.

> **Note**
>
> Se você quiser criar um token de acesso de usuário, verifique se já instalou o AWS Connector for GitHub app e deixe o campo de instalação do aplicativo vazio. AWS CodeConnections usará o token de acesso do usuário para a conexão.

4. Na GitHub página Install AWS Connector for, escolha a conta na qual você deseja instalar o aplicativo.

> **Note**
>
> Você só instala o aplicativo uma vez para cada GitHub conta. Se você instalou a aplicação anteriormente, poderá escolher Configure (Configurar) para prosseguir para uma página de modificação para a instalação da aplicação ou usar o botão Back (Voltar) para retornar ao console.

5. Na GitHub página Install AWS Connector for, deixe os padrões e escolha Instalar.
6. Na GitHub página Connect to, o ID de conexão da sua nova instalação aparece em GitHub Apps. Selecione Conectar.

Etapa 3: Salve sua ação GitHub de origem

Execute estas etapas na página Editar ação para salvar a ação de origem com as informações de conexão.

Para salvar sua ação GitHub de origem

1. Em Repository name (Nome do repositório), escolha o nome do repositório de terceiros.
2. Em Gatilhos do Pipeline, você pode adicionar gatilhos se sua ação for uma ação. AWS CodeConnections Para definir a configuração de gatilhos do pipeline e, se necessário, aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request.
3. Em Output artifact format (Formato de artefato de saída), você deve escolher o formato para seus artefatos.

- Para armazenar artefatos de saída da GitHub ação usando o método padrão, escolha CodePipeline default. A ação acessa os arquivos do GitHub repositório e armazena os artefatos em um arquivo ZIP no repositório de artefatos do pipeline.
- Para armazenar um arquivo JSON que contém uma referência de URL ao repositório para que as ações downstream possam executar comandos Git diretamente, escolha Full clone (Clone completo). Essa opção só pode ser usada por ações CodeBuild posteriores.

  Se você escolher essa opção, precisará atualizar as permissões para sua função de serviço CodeBuild do projeto, conforme mostrado emAdicione CodeBuild GitClone permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com. Para assistir a um tutorial que mostre como usar a opção Clone completo, consulte Tutorial: use o clone completo com uma fonte de GitHub pipeline.

4. Escolha Próximo no assistente ou Salvar na página Editar ação.

Crie uma conexão com GitHub (CLI)

Você pode usar o AWS Command Line Interface (AWS CLI) para criar uma conexão.

Para fazer isso, use o comando create-connection.

> **Important**
>
> Uma conexão criada por meio do AWS CLI ou AWS CloudFormation está no `PENDING` status por padrão. Depois de criar uma conexão com a CLI ou CloudFormation, use o console para editar a conexão e definir seu status. `AVAILABLE`

Para criar uma conexão

1. Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows). Use o AWS CLI para executar o create-connection comando, especificando `--provider-type` e `--connection-name` para sua conexão. Neste exemplo, o nome do provedor de terceiros é `GitHub` e o nome da conexão especificada é `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type GitHub --connection-name
    MyConnection
   ```

Se tiver êxito, esse comando gerará as informações do ARN de conexão semelhantes às seguintes.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Use o console para concluir a conexão. Para obter mais informações, consulte Atualizar uma conexão pendente.
3. O pipeline assume como padrão a detecção de alterações ao enviar o código por push ao repositório de origem da conexão. Para definir a configuração do gatilho do pipeline para liberação manual ou para tags Git, execute um dos seguintes procedimentos:

   - Para definir a configuração do gatilho do pipeline para início somente por meio de liberação manual, adicione a seguinte linha à configuração:

     ```
     "DetectChanges": "false",
     ```

   - Para definir a configuração de gatilhos do pipeline e aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request. Por exemplo, o trecho a seguir é adicionado ao nível do pipeline na definição JSON do pipeline. Neste exemplo, `release-v0` e `release-v1` são as tags Git a serem incluídas, enquanto `release-v2` são as tags Git a serem excluídas.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
     ```

```
                    }
                ]
            }
        }
    ]
```

GitHub Conexões do Enterprise Server

As conexões permitem que você autorize e estabeleça configurações que associem seu provedor terceirizado aos seus AWS recursos. Para associar seu repositório de terceiros como origem do pipeline, use uma conexão.

> **Note**
>
> Em vez de criar ou usar uma conexão existente na conta, você pode usar uma conexão compartilhada entre outra Conta da AWS. Consulte Usar uma conexão compartilhada com outra conta.

> **Note**
>
> Esse recurso não está disponível nas regiões Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Melbourne), Ásia-Pacífico (Osaka), África (Cidade do Cabo), Oriente Médio (Bahrein), Oriente Médio (EAU), Europa (Espanha), Europa (Zurique), Israel (Tel Aviv) ou (). AWS GovCloud US-West Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline. Para considerações sobre essa ação na região Europa (Milão), consulte a nota em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

Para adicionar uma ação de origem do GitHub Enterprise Server CodePipeline, você pode escolher entre:

- Use o assistente de criação de pipeline do CodePipeline console ou a página Editar ação para escolher a opção de provedor do GitHub Enterprise Server. Consulte Crie uma conexão com o GitHub Enterprise Server (console) para adicionar a ação. O console ajuda você a criar um recurso de host e um recurso de conexão.

- Usar a CLI para adicionar a configuração da ação `CreateSourceConnection` com o provedor `GitHubEnterpriseServer` e criar seus recursos:
  - Para criar seus recursos de conexão, consulte Crie um host e uma conexão com o GitHub Enterprise Server (CLI) para criar um recurso de host e um recurso de conexão com a CLI.
  - Use o exemplo de configuração da ação `CreateSourceConnection` em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com para adicionar sua ação, conforme mostrado em Criar um pipeline (CLI).

> **Note**
>
> É possível criar uma conexão por meio do console do Developer Tools em Configurações. Consulte Criar uma conexão.

Antes de começar

- Você deve ter criado uma conta no GitHub Enterprise Server e instalado a instância do GitHub Enterprise Server em sua infraestrutura.

  > **Note**
  >
  > Cada VPC só pode ser associada a um host (instância do GitHub Enterprise Server) por vez.

- Você já deve ter criado um repositório de código com o GitHub Enterprise Server.

Tópicos

- Crie uma conexão com o GitHub Enterprise Server (console)
- Crie um host e uma conexão com o GitHub Enterprise Server (CLI)

Crie uma conexão com o GitHub Enterprise Server (console)

Use essas etapas para usar o CodePipeline console para adicionar uma ação de conexões ao seu repositório do GitHub Enterprise Server.

> **Note**
>
> GitHub As conexões do Enterprise Server fornecem acesso somente aos repositórios pertencentes à conta do GitHub Enterprise Server que foi usada para criar a conexão.

Antes de começar

Para uma conexão de host com o GitHub Enterprise Server, você deve ter concluído as etapas para criar um recurso de host para sua conexão. Consulte Gerenciar hosts para conexões.

Etapa 1: Criar ou editar seu pipeline

Para criar ou editar seu pipeline

1. Faça login no CodePipeline console.
2. Escolha uma das opções a seguir.
   - Opte por criar um pipeline. Siga as etapas em Criar um pipeline para concluir a primeira tela e escolha Próximo. Na página Origem, em Provedor de origem, escolha GitHub Enterprise Server.
   - Opte por editar um pipeline existente. Escolha Editar e, em seguida, escolha Editar estágio. Escolha adicionar ou editar sua ação de origem. Na página Editar ação, em Nome da ação, insira um nome para a ação. Em Provedor de ação, escolha GitHub Enterprise Server.
3. Execute um destes procedimentos:
   - Em Conexão, se você ainda não tiver criado uma conexão com seu provedor, escolha Conectar ao GitHub Enterprise Server. Prossiga para a Etapa 2: Criar uma conexão com o GitHub Enterprise Server.
   - Em Conexão, se você já tiver criado uma conexão com seu provedor, escolha a conexão. Vá para a Etapa 3: Salvar a ação de origem para sua conexão.

Crie uma conexão com o GitHub Enterprise Server

Depois de escolher criar a conexão, a página Connect to GitHub Enterprise Server é exibida.

> **Important**
>
> AWS CodeConnections não oferece suporte ao GitHub Enterprise Server versão 2.22.0 devido a um problema conhecido na versão. Para conectar, atualize para a versão 2.22.1 ou a versão mais recente disponível.

Para se conectar ao GitHub Enterprise Server

1. Em Connection name (Nome da conexão), informe um nome para a conexão.
2. Em URL, insira o endpoint do seu servidor.

   > **Note**
   >
   > Se a URL fornecida já tiver sido usada para configurar um GitHub Enterprise Server para uma conexão, você será solicitado a escolher o ARN do recurso de host que foi criado anteriormente para esse endpoint.

3. Se você tiver iniciado seu servidor em uma Amazon VPC e quiser se conectar à sua VPC, escolhaUse a VPC (Usar uma VPC) e conclua as operações a seguir.

   a. Em VPC ID (ID da VPC), escolha o ID da sua VPC. Certifique-se de escolher a VPC para a infraestrutura em que sua instância do GitHub Enterprise Server está instalada ou uma VPC com acesso à sua instância do GitHub Enterprise Server por meio de VPN ou Direct Connect.

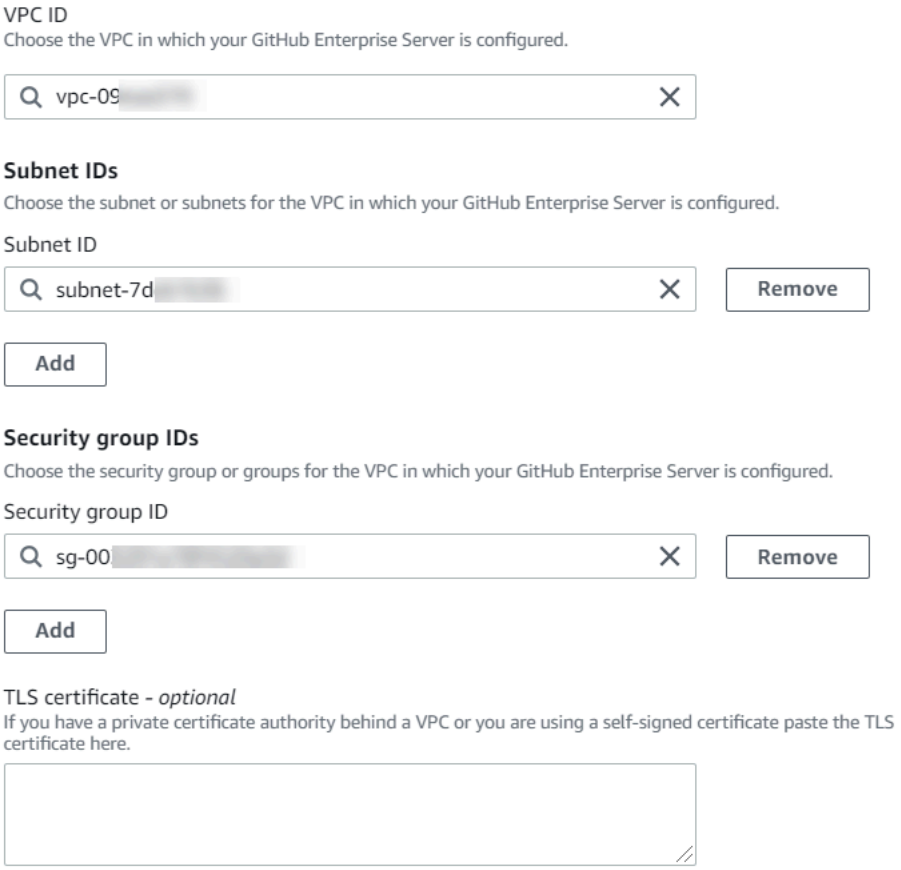
   b. Em Subnet ID (ID da sub-rede), escolha Add (Adicionar). No campo, escolha o ID da sub-rede que você deseja usar para seu host. Você pode escolher até 10 sub-redes.

      Certifique-se de escolher a sub-rede para a infraestrutura em que sua instância do GitHub Enterprise Server está instalada ou uma sub-rede com acesso à sua instância instalada do GitHub Enterprise Server por meio de VPN ou Direct Connect.

   c. Em Security group IDs (IDs de grupos de segurança), escolha Add (Adicionar). No campo, escolha o grupo de segurança que você deseja usar para seu host. Você pode criar até 10 grupos de segurança.

      Certifique-se de escolher o grupo de segurança para a infraestrutura em que sua instância do GitHub Enterprise Server está instalada ou um grupo de segurança com acesso à sua instância instalada do GitHub Enterprise Server por meio de VPN ou Direct Connect.

d. Se você tiver uma VPC privada configurada e tiver configurado sua instância do GitHub Enterprise Server para realizar a validação de TLS usando uma autoridade de certificação não pública, em Certificado TLS, insira seu ID do certificado. O valor do certificado TLS deve ser a chave pública do certificado.



4. Escolha Connect to GitHub Enterprise Server. A conexão criada é mostrada com um status Pending (Pendente). Um recurso de host é criado para a conexão com as informações do servidor fornecidas. Para o nome do host, o URL é usado.
5. Selecione Update pending connection (Atualizar conexão pendente).
6. Se solicitado, na página de login do GitHub Enterprise, faça login com suas credenciais do GitHub Enterprise.
7. Na página Criar GitHub aplicativo, escolha um nome para seu aplicativo.
8. <app-name>Na página de GitHub autorização, escolha Autorizar.
9. Na página de instalação da aplicação, uma mensagem informa que a aplicação de conector está pronta para ser instalada. Se você tiver várias organizações, poderá ser solicitado a escolher a organização onde deseja instalar a aplicação.

Escolha as configurações do repositório em que deseja instalar a aplicação. Escolha Instalar.

10. A página de conexão mostra a conexão criada em um status Available (Disponível).

Etapa 3: Salvar sua ação de origem do GitHub Enterprise Server

Execute estas etapas no assistente ou na página Editar ação para salvar a ação de origem com as informações de conexão.

Para concluir e salvar a ação de origem com a conexão

1. Em Repository name (Nome do repositório), escolha o nome do repositório de terceiros.
2. Em Gatilhos do Pipeline, você pode adicionar gatilhos se sua ação for uma ação. AWS CodeConnections Para definir a configuração de gatilhos do pipeline e, se necessário, aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request.
3. Em Output artifact format (Formato de artefato de saída), você deve escolher o formato para seus artefatos.
   - Para armazenar artefatos de saída da ação do GitHub Enterprise Server usando o método padrão, escolha CodePipelinedefault. A ação acessa os arquivos do repositório do GitHub Enterprise Server e armazena os artefatos em um arquivo ZIP no repositório de artefatos do pipeline.
   - Para armazenar um arquivo JSON que contém uma referência de URL ao repositório para que as ações downstream possam executar comandos Git diretamente, escolha Full clone (Clone completo). Essa opção só pode ser usada por ações CodeBuild posteriores.
4. Escolha Próximo no assistente ou Salvar na página Editar ação.

Crie um host e uma conexão com o GitHub Enterprise Server (CLI)

Você pode usar o AWS Command Line Interface (AWS CLI) para criar uma conexão.

Para fazer isso, use o comando create-connection.

> **Important**
>
> Uma conexão criada por meio do AWS CLI ou AWS CloudFormation está no `PENDING` status por padrão. Depois de criar uma conexão com a CLI ou CloudFormation, use o console para editar a conexão e definir seu status. `AVAILABLE`

Você pode usar o AWS Command Line Interface (AWS CLI) para criar um host para conexões instaladas.

> **Note**
>
> Você só cria um host uma vez por conta do GitHub Enterprise Server. Todas as suas conexões com uma conta específica do GitHub Enterprise Server usarão o mesmo host.

Você usa um host para representar o endpoint da infraestrutura em que seu provedor de terceiros está instalado. Após concluir a criação do host com a CLI, o host fica no status Pendente. Então, você configura ou registra o host para movê-lo para o status Disponível. Depois que o host estiver disponível, conclua as etapas para criar uma conexão.

Para fazer isso, use o comando create-host.

> **Important**
>
> Um host criado por meio do `Pending` status AWS CLI is in por padrão. Após criar um host com a CLI, use o console ou a CLI para configurar o host e tornar seu status `Available`.

Para criar um host

1. Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows). Use o AWS CLI para executar o create-host comando, especificando o `--name--provider-type`, e `--provider-endpoint` para sua conexão. Neste exemplo, o nome do provedor de terceiros é `GitHubEnterpriseServer` e o endpoint é `my-instance.dev`.

   ```
   aws codestar-connections create-host --name MyHost --provider-type
    GitHubEnterpriseServer --provider-endpoint "https://my-instance.dev"
   ```

Se o comando for bem-sucedido, ele retornará as informações de nome do recurso da Amazon (ARN) do host semelhantes às mostradas a seguir.

```
{
    "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
Host-28aef605"
}
```

Após esta etapa, o host estará no status `PENDING`.

2. Use o console para concluir a configuração do host e mova o host para um status `Available`.

Para criar uma conexão com o GitHub Enterprise Server

1. Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows). Use o AWS CLI para executar o create-connection comando, especificando `--host-arn` e `--connection-name` para sua conexão.

```
aws codestar-connections create-connection --host-arn arn:aws:codestar-
connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
 MyConnection
```

Se tiver êxito, esse comando gerará as informações do ARN de conexão semelhantes às seguintes.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad"
}
```

2. Use o console para configurar a conexão pendente.
3. O pipeline assume como padrão a detecção de alterações ao enviar o código por push ao repositório de origem da conexão. Para definir a configuração do gatilho do pipeline para liberação manual ou para tags Git, execute um dos seguintes procedimentos:

   - Para definir a configuração do gatilho do pipeline para início somente por meio de liberação manual, adicione a seguinte linha à configuração:

```
"DetectChanges": "false",
```

- Para definir a configuração de gatilhos do pipeline e aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request. Por exemplo, o trecho a seguir é adicionado ao nível do pipeline na definição JSON do pipeline. Neste exemplo, `release-v0` e `release-v1` são as tags Git a serem incluídas, enquanto `release-v2` são as tags Git a serem excluídas.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "tags": {
                                "includes": [
                                    "release-v0", "release-v1"
                                ],
                                "excludes": [
                                    "release-v2"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
```

GitLab.com conexões

As conexões permitem que você autorize e estabeleça configurações que associem seu provedor terceirizado aos seus AWS recursos. Para associar seu repositório de terceiros como origem do pipeline, use uma conexão.

> **Note**
>
> Em vez de criar ou usar uma conexão existente na conta, você pode usar uma conexão compartilhada entre outra Conta da AWS. Consulte Usar uma conexão compartilhada com outra conta.

> **Note**
>
> Esse recurso não está disponível nas regiões Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Melbourne), Ásia-Pacífico (Osaka), África (Cidade do Cabo), Oriente Médio (Bahrein), Oriente Médio (EAU), Europa (Espanha), Europa (Zurique), Israel (Tel Aviv) ou (). AWS GovCloud US-West Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline. Para considerações sobre essa ação na região Europa (Milão), consulte a nota em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

Para adicionar uma ação de GitLab.com origem CodePipeline, você pode escolher entre:

- Use o assistente de criação de pipeline do CodePipeline console ou a página Editar ação para escolher a opção GitLabde provedor. Consulte Crie uma conexão com GitLab.com (console) para adicionar a ação. O console ajuda você a criar um recurso de conexão.
- Usar a CLI para adicionar a configuração da ação `CreateSourceConnection` com o provedor `GitLab`:
  - Para criar seus recursos de conexão, consulte Crie uma conexão com GitLab.com (CLI) para criar um recurso de conexão com a CLI.
  - Use o exemplo de configuração da ação `CreateSourceConnection` em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com para adicionar sua ação, conforme mostrado em Criar um pipeline (CLI).

> **Note**
>
> É possível criar uma conexão por meio do console do Developer Tools em Configurações. Consulte Criar uma conexão.

> **Note**
>
> Ao autorizar a instalação dessa conexão em GitLab.com, você concede ao nosso serviço permissões para processar seus dados acessando sua conta e pode revogar as permissões a qualquer momento desinstalando o aplicativo.

Antes de começar

- Você já deve ter criado uma conta com GitLab.com.

  > **Note**
  >
  > As conexões fornecem acesso somente a repositórios pertencentes à conta usada para criar e autorizar a conexão.

  > **Note**
  >
  > Você pode criar conexões com um repositório no qual você tem a função de Proprietário e GitLab, em seguida, a conexão pode ser usada com o repositório com recursos como. CodePipeline Para repositórios em grupos, você não precisa ser o proprietário do grupo.

- Para especificar uma origem para seu pipeline, é necessário que você já tenha criado um repositório no gitlab.com.

Tópicos

- Crie uma conexão com GitLab.com (console)
- Crie uma conexão com GitLab.com (CLI)

Crie uma conexão com GitLab.com (console)

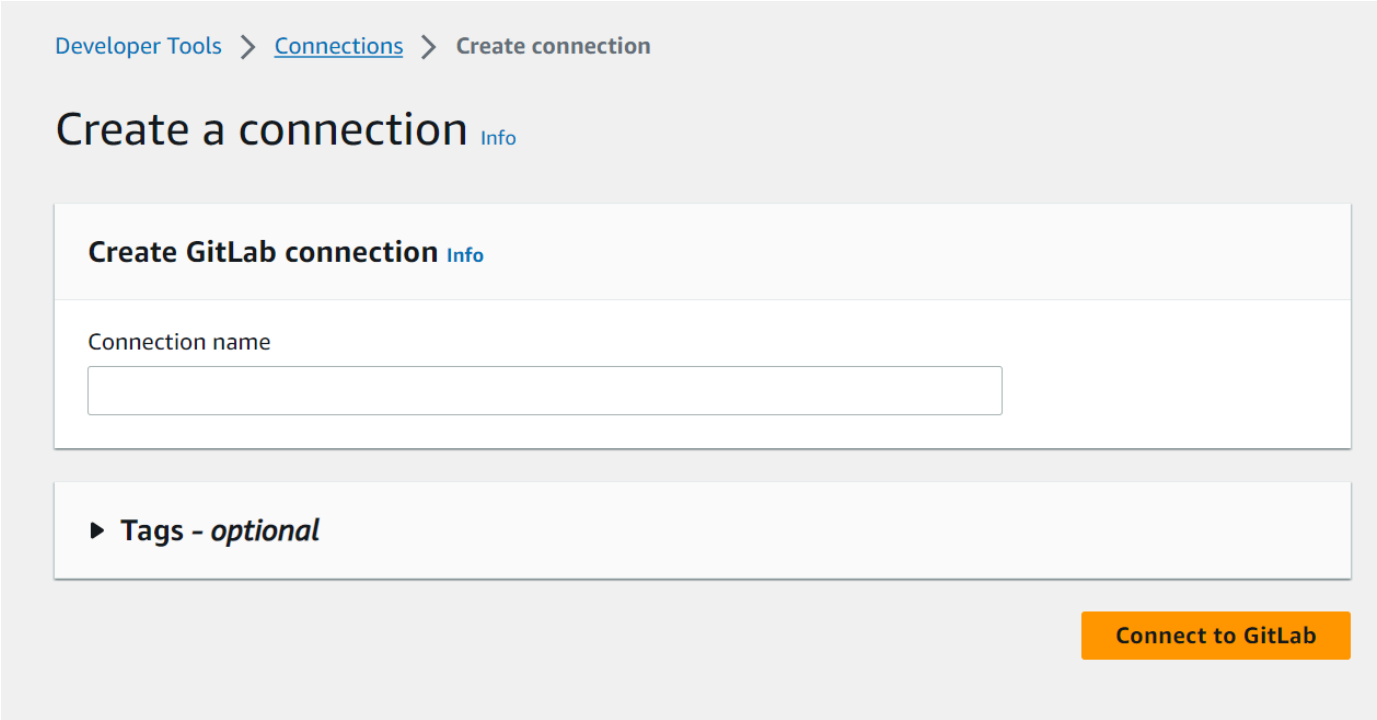
Use essas etapas para usar o CodePipeline console para adicionar uma ação de conexões para seu projeto (repositório) no GitLab.

Para criar ou editar seu pipeline

1. Faça login no CodePipeline console.
2. Escolha uma das opções a seguir.
   - Opte por criar um pipeline. Siga as etapas em Criar um pipeline para concluir a primeira tela e escolha Próximo. Na página Fonte, em Provedor de Origem, escolha GitLab.
   - Opte por editar um pipeline existente. Escolha Editar e, em seguida, escolha Editar estágio. Escolha adicionar ou editar sua ação de origem. Na página Editar ação, em Nome da ação, insira um nome para a ação. Em Provedor de ação, selecione GitLab.
3. Execute um destes procedimentos:
   - Em Conexão, se você ainda não tiver criado uma conexão com seu provedor, escolha Conectar GitLab a. Vá para a Etapa 4 para criar uma conexão.
   - Em Conexão, se você ainda não criou uma conexão com seu provedor, escolha a conexão. Vá para a etapa 9.

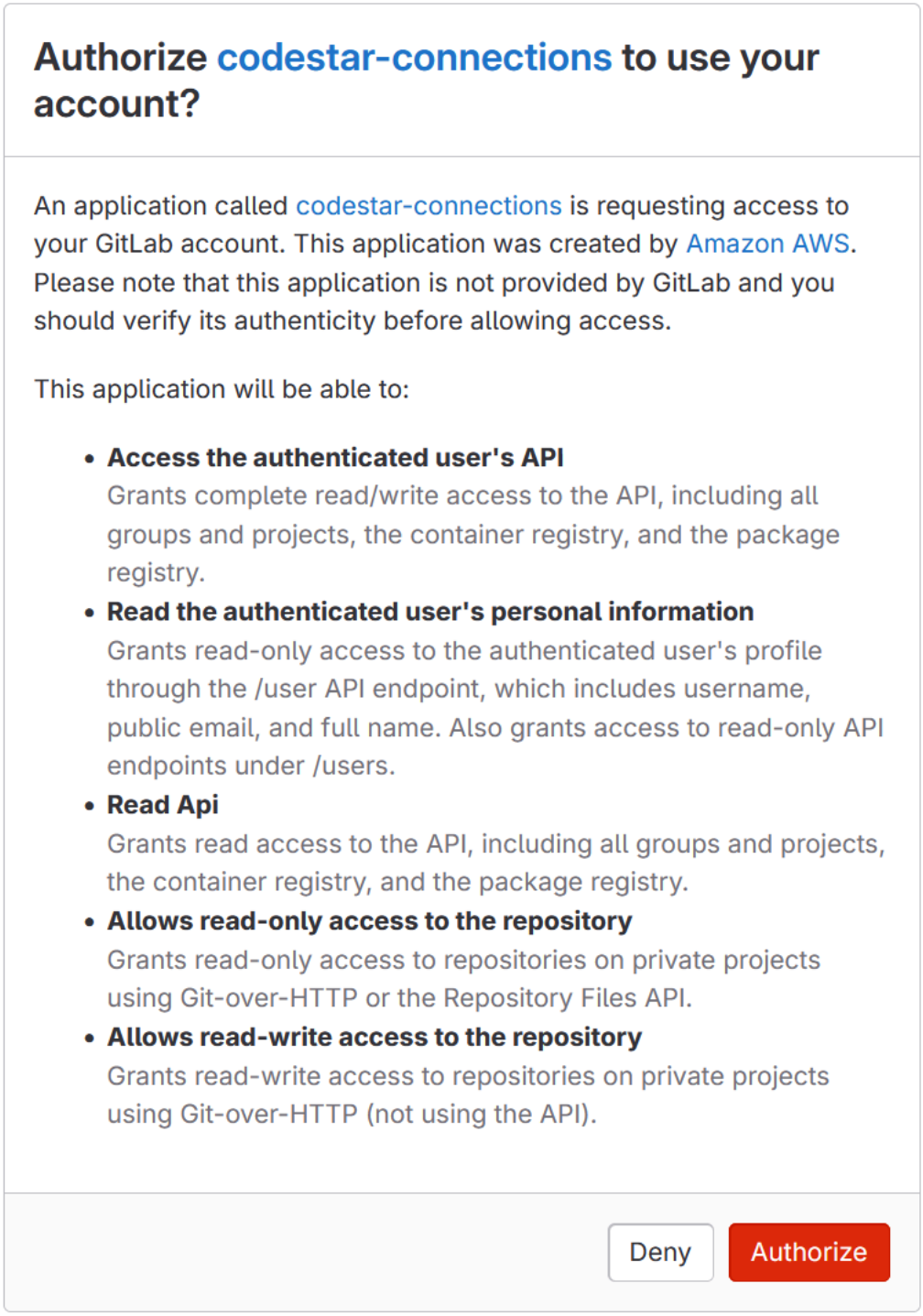
   > **Note**
   >
   > Se você fechar a janela pop-up antes de criar uma GitLab.com conexão, precisará atualizar a página.

4. Para criar uma conexão com um GitLab.com repositório, em Selecionar um provedor, escolha GitLab. Em Connection name (Nome da conexão), digite o nome da conexão que você deseja criar. Escolha Connect to GitLab.



5. Quando a página de login for GitLab.com exibida, faça login com suas credenciais e escolha Entrar.
6. Se esta for a primeira vez que você autoriza a conexão, uma página de autorização será exibida com uma mensagem solicitando autorização para que a conexão acesse sua GitLab.com conta.

   Escolha Authorize.



7. O navegador retorna à página do console de conexões. Em Criar GitLab conexão, a nova conexão é mostrada em Nome da conexão.
8. Escolha Connect to GitLab.

   Você retornará ao CodePipeline console.

> **Note**
>
> Depois que uma GitLab.com conexão for criada com sucesso, um banner de sucesso será exibido na janela principal.
> Se você não tiver feito login anteriormente GitLab na máquina atual, precisará fechar manualmente a janela pop-up.

9. Em Nome do repositório, escolha o nome do seu projeto em GitLab especificando o caminho do projeto com o namespace. Por exemplo, para um repositório em nível de grupo, insira o nome do repositório no seguinte formato: `group-name/repository-name`. Para obter mais informações sobre o caminho e o namespace, consulte o `path_with_namespace` campo em https://docs.gitlab.com/ee/api/projects.html#get -single-project. Para obter mais informações sobre o namespace em GitLab, consulte. https://docs.gitlab.com/ee/user/namespace/

> **Note**
>
> Para grupos em GitLab, você deve especificar manualmente o caminho do projeto com o namespace. Por exemplo, para um repositório `myrepo` em um grupo `mygroup`, insira o seguinte: `mygroup/myrepo`. Você pode encontrar o caminho do projeto com o namespace na URL em. GitLab

10. Em Gatilhos do Pipeline, você pode adicionar gatilhos se sua ação for uma ação. AWS CodeConnections Para definir a configuração de gatilhos do pipeline e, se necessário, aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request.
11. Em Branch name (Nome da ramificação), escolha a ramificação onde deseja que o pipeline detecte alterações de origem.

> **Note**
>
> Se o nome da ramificação não for preenchido automaticamente, você não terá acesso de Proprietário ao repositório. Ou o nome do projeto não é válido ou a conexão usada não tem acesso ao project/repository.

12. Em Output artifact format (Formato de artefato de saída), você deve escolher o formato para seus artefatos.

- Para armazenar artefatos de saída da GitLab.com ação usando o método padrão, escolha CodePipeline default. A ação acessa os arquivos do GitLab.com repositório e armazena os artefatos em um arquivo ZIP no repositório de artefatos do pipeline.
- Para armazenar um arquivo JSON que contém uma referência de URL ao repositório para que as ações downstream possam executar comandos Git diretamente, escolha Full clone (Clone completo). Esta opção só pode ser usada por ações downstream do CodeBuild .

  Se você escolher essa opção, precisará atualizar as permissões para sua função de serviço CodeBuild do projeto, conforme mostrado emAdicione CodeBuild GitClone permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com. Para assistir a um tutorial que mostre como usar a opção Clone completo, consulte Tutorial: use o clone completo com uma fonte de GitHub pipeline.

13. Opte por salvar a ação de origem e continuar.

Crie uma conexão com GitLab.com (CLI)

Você pode usar o AWS Command Line Interface (AWS CLI) para criar uma conexão.

Para fazer isso, use o comando create-connection.

> **Important**
>
> Uma conexão criada por meio do AWS CLI ou AWS CloudFormation está no `PENDING` status por padrão. Depois de criar uma conexão com a CLI ou CloudFormation, use o console para editar a conexão e definir seu status. `AVAILABLE`

Para criar uma conexão

1. Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows). Use o AWS CLI para executar o create-connection comando, especificando `--provider-type` e `--connection-name` para sua conexão. Neste exemplo, o nome do provedor de terceiros é `GitLab` e o nome da conexão especificada é `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type GitLab --connection-name
    MyConnection
   ```

Se tiver êxito, esse comando gerará as informações do ARN de conexão semelhantes às seguintes.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Use o console para concluir a conexão. Para obter mais informações, consulte Atualizar uma conexão pendente.
3. O pipeline assume como padrão a detecção de alterações ao enviar o código por push ao repositório de origem da conexão. Para definir a configuração do gatilho do pipeline para liberação manual ou para tags Git, execute um dos seguintes procedimentos:

   - Para definir a configuração do gatilho do pipeline para início somente por meio de liberação manual, adicione a seguinte linha à configuração:

     ```
     "DetectChanges": "false",
     ```

   - Para definir a configuração de gatilhos do pipeline e aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request. Por exemplo, o trecho a seguir é adicionado ao nível do pipeline na definição JSON do pipeline. Neste exemplo, `release-v0` e `release-v1` são as tags Git a serem incluídas, enquanto `release-v2` são as tags Git a serem excluídas.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
     ```

```
                    }
                ]
            }
        }
    ]
```

Conexões para GitLab autogerenciamento

As conexões permitem que você autorize e estabeleça configurações que associem seu provedor terceirizado aos seus AWS recursos. Para associar seu repositório de terceiros como origem do pipeline, use uma conexão.

> **Note**
>
> Em vez de criar ou usar uma conexão existente na conta, você pode usar uma conexão compartilhada entre outra Conta da AWS. Consulte Usar uma conexão compartilhada com outra conta.

> **Note**
>
> Esse recurso não está disponível nas regiões Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Melbourne), Ásia-Pacífico (Osaka), África (Cidade do Cabo), Oriente Médio (Bahrein), Oriente Médio (EAU), Europa (Espanha), Europa (Zurique), Israel (Tel Aviv) ou (). AWS GovCloud US-West Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline. Para considerações sobre essa ação na região Europa (Milão), consulte a nota em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

Para adicionar uma ação de origem GitLab autogerenciada CodePipeline, você pode escolher entre:

- Use o assistente de criação de pipeline do CodePipeline console ou a página Editar ação para escolher a opção de provedor GitLabautogerenciado. Consulte Crie uma conexão com o GitLab autogerenciado (console) para adicionar a ação. O console ajuda você a criar um recurso de host e um recurso de conexão.

- Usar a CLI para adicionar a configuração da ação `CreateSourceConnection` com o provedor `GitLabSelfManaged` e criar seus recursos:
  - Para criar seus recursos de conexão, consulte Crie um host e uma conexão com o GitLab autogerenciado (CLI) para criar um recurso de host e um recurso de conexão com a CLI.
  - Use o exemplo de configuração da ação `CreateSourceConnection` em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com para adicionar sua ação, conforme mostrado em Criar um pipeline (CLI).

> **Note**
>
> É possível criar uma conexão por meio do console do Developer Tools em Configurações. Consulte Criar uma conexão.

Antes de começar

- Você já deve ter criado uma conta GitLab e ter a GitLab Enterprise Edition ou a GitLab Community Edition com uma instalação autogerenciada. Para obter mais informações, consulte https://docs.gitlab.com/ee/subscriptions/self_managed/.

  > **Note**
  >
  > As conexões fornecem acesso à conta usada para criar e autorizar a conexão.

  > **Note**
  >
  > Você pode criar conexões com um repositório no qual você tem a função de Proprietário e GitLab, em seguida, a conexão pode ser usada com recursos como CodePipeline. Para repositórios em grupos, você não precisa ser o proprietário do grupo.

- Você já deve ter criado um token de acesso GitLab pessoal (PAT) com o `api` escopo. Se o Modo Administrativo estiver ativado na sua GitLab instância, o PAT também deverá incluir o `admin_mode` escopo. Para obter mais informações, consulte Tokens de acesso pessoal e Escopos de token de acesso no GitLab site. Para criar e utilizar o PAT, é necessário ser administrador.

> **Note**
>
> O PAT é usado para autorizar o host e não é armazenado ou usado pelas conexões. Para configurar um host, é possível criar um PAT temporário e, depois de configurar o host, você pode excluir o PAT.

- É possível optar por configurar o host com antecedência. É possível configurar um host com ou sem uma VPC. Para obter detalhes sobre a configuração da VPC e informações adicionais sobre a criação de um host, consulte Criar um host.

Tópicos

- Crie uma conexão com o GitLab autogerenciado (console)
- Crie um host e uma conexão com o GitLab autogerenciado (CLI)

Crie uma conexão com o GitLab autogerenciado (console)

Use essas etapas para usar o CodePipeline console para adicionar uma ação de conexões ao seu repositório GitLab autogerenciado.

> **Note**
>
> GitLab as conexões autogerenciadas fornecem acesso somente aos repositórios pertencentes à conta GitLab autogerenciada que foi usada para criar a conexão.

Antes de começar

Para que uma conexão de host seja GitLab autogerenciada, você deve ter concluído as etapas para criar um recurso de host para sua conexão. Consulte Gerenciar hosts para conexões.

Etapa 1: Criar ou editar seu pipeline

Para criar ou editar seu pipeline

1. Faça login no CodePipeline console.
2. Escolha uma das opções a seguir.

- Opte por criar um pipeline. Siga as etapas em Criar um pipeline para concluir a primeira tela e escolha Próximo. Na página Fonte, em Provedor de origem, escolha GitLab autogerenciado.
- Opte por editar um pipeline existente. Escolha Editar e, em seguida, escolha Editar estágio. Escolha adicionar ou editar sua ação de origem. Na página Editar ação, em Nome da ação, insira um nome para a ação. Em Provedor de ações, escolha GitLab autogerenciado.

3. Execute um destes procedimentos:

- Em Conexão, se você ainda não tiver criado uma conexão com seu provedor, escolha Conectar ao GitLab autogerenciado. Prossiga para a Etapa 2: Criar uma conexão com o GitLab autogerenciado.
- Em Conexão, se você já tiver criado uma conexão com seu provedor, escolha a conexão e vá para a Etapa 3: Salvar sua ação de origem GitLab autogerenciada.

Etapa 2: criar uma conexão com o GitLab autogerenciado

Depois de escolher criar a conexão, a página Connect to GitLab self-managed é exibida.

Para se conectar ao GitLab autogerenciado

1. Em Connection name (Nome da conexão), informe um nome para a conexão.
2. Em URL, insira o endpoint do seu servidor.

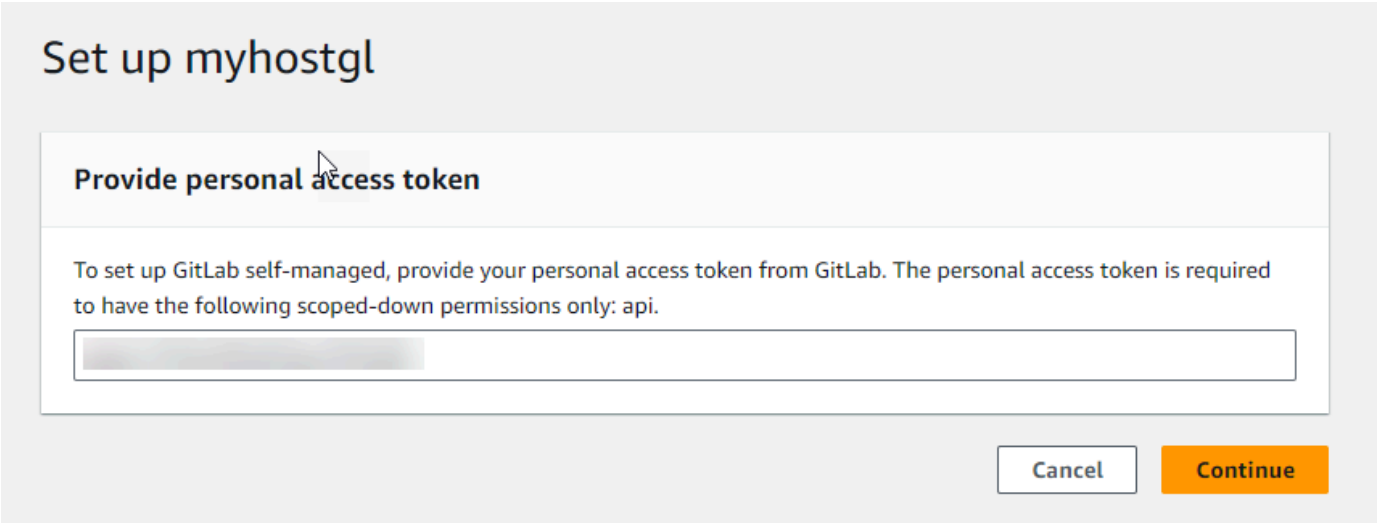
> **Note**
>
> Se o URL fornecido já tiver sido usado para configurar um host para uma conexão, você será solicitado a escolher o ARN de recurso de host criado anteriormente para esse endpoint.

3. Se você tiver iniciado o servidor em uma Amazon VPC e quiser se conectar à VPC, selecione Usar uma VPC e preencha as informações da VPC.
4. Escolha Conectar ao GitLab autogerenciado. A conexão criada é mostrada com um status Pending (Pendente). Um recurso de host é criado para a conexão com as informações do servidor fornecidas. Para o nome do host, o URL é usado.
5. Selecione Update pending connection (Atualizar conexão pendente).
6. Se uma página for aberta com uma mensagem de redirecionamento confirmando que você deseja continuar com o provedor, selecione Continuar. Insira a autorização para o provedor.

7. Uma *host_name* página de configuração é exibida. Em Fornecer token de acesso pessoal, forneça GitLab ao PAT o `api` escopo. Se o Modo Administrativo estiver ativado na sua GitLab instância, o PAT também deverá incluir o `admin_mode` escopo. Para obter mais informações, consulte Escopos do token de acesso no GitLab site.

   > **Note**
   >
   > Apenas administradores podem criar e utilizar o PAT.

   Escolha Continuar.



8. A página de conexão mostra a conexão criada em um status Available (Disponível).

Etapa 3: Salve sua GitLab ação de origem autogerenciada

Execute estas etapas no assistente ou na página Editar ação para salvar a ação de origem com as informações de conexão.

Para concluir e salvar a ação de origem com a conexão

1. Em Repository name (Nome do repositório), escolha o nome do repositório de terceiros.
2. Em Gatilhos do Pipeline, você pode adicionar gatilhos se sua ação for uma ação. AWS CodeConnections Para definir a configuração de gatilhos do pipeline e, se necessário, aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request.

3. Em Output artifact format (Formato de artefato de saída), você deve escolher o formato para seus artefatos.

   - Para armazenar artefatos de saída da ação GitLab autogerenciada usando o método padrão, escolha CodePipeline default. A ação acessa os arquivos no repositório e armazena os artefatos em um arquivo ZIP no armazenamento de artefatos do pipeline.
   - Para armazenar um arquivo JSON que contém uma referência de URL ao repositório para que as ações downstream possam executar comandos Git diretamente, escolha Full clone (Clone completo). Essa opção só pode ser usada por ações CodeBuild posteriores.

4. Escolha Próximo no assistente ou Salvar na página Editar ação.

Crie um host e uma conexão com o GitLab autogerenciado (CLI)

Você pode usar o AWS Command Line Interface (AWS CLI) para criar uma conexão.

Para fazer isso, use o comando create-connection.

> **Important**
>
> Uma conexão criada por meio do AWS CLI ou AWS CloudFormation está no `PENDING` status por padrão. Depois de criar uma conexão com a CLI ou CloudFormation, use o console para editar a conexão e definir seu status. `AVAILABLE`

Você pode usar o AWS Command Line Interface (AWS CLI) para criar um host para conexões instaladas.

Você usa um host para representar o endpoint da infraestrutura em que seu provedor de terceiros está instalado. Após concluir a criação do host com a CLI, o host fica no status Pendente. Então, você configura ou registra o host para movê-lo para o status Disponível. Depois que o host estiver disponível, conclua as etapas para criar uma conexão.

Para fazer isso, use o comando create-host.

> **Important**
>
> Um host criado por meio do `Pending` status AWS CLI is in por padrão. Após criar um host com a CLI, use o console ou a CLI para configurar o host e tornar seu status `Available`.

Para criar um host

1. Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows). Use o AWS CLI para executar o create-host comando, especificando o `--name`-`-provider-type`, e `--provider-endpoint` para sua conexão. Neste exemplo, o nome do provedor de terceiros é `GitLabSelfManaged` e o endpoint é `my-instance.dev`.

   ```
   aws codestar-connections create-host --name MyHost --provider-type
    GitLabSelfManaged --provider-endpoint "https://my-instance.dev"
   ```

   Se o comando for bem-sucedido, ele retornará as informações de nome do recurso da Amazon (ARN) do host semelhantes às mostradas a seguir.

   ```
   {
       "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
   Host-28aef605"
   }
   ```

   Após esta etapa, o host estará no status `PENDING`.

2. Use o console para concluir a configuração do host e mova o host para um status `Available`.

Para criar uma conexão com o GitLab autogerenciado

1. Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows). Use o AWS CLI para executar o create-connection comando, especificando `--host-arn` e `--connection-name` para sua conexão.

   ```
   aws codestar-connections create-connection --host-arn arn:aws:codestar-
   connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
    MyConnection
   ```

   Se tiver êxito, esse comando gerará as informações do ARN de conexão semelhantes às seguintes.

   ```
   {
       "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
   aEXAMPLE-8aad"
   }
   ```

2. Use o console para configurar a conexão pendente.
3. O pipeline assume como padrão a detecção de alterações ao enviar o código por push ao repositório de origem da conexão. Para definir a configuração do gatilho do pipeline para liberação manual ou para tags Git, execute um dos seguintes procedimentos:

   - Para definir a configuração do gatilho do pipeline para início somente por meio de liberação manual, adicione a seguinte linha à configuração:

     ```
     "DetectChanges": "false",
     ```

   - Para definir a configuração de gatilhos do pipeline e aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request. Por exemplo, o trecho a seguir é adicionado ao nível do pipeline na definição JSON do pipeline. Neste exemplo, `release-v0` e `release-v1` são as tags Git a serem incluídas, enquanto `release-v2` são as tags Git a serem excluídas.

     ```
     "triggers": [
                 {
                     "providerType": "CodeStarSourceConnection",
                     "gitConfiguration": {
                         "sourceActionName": "Source",
                         "push": [
                             {
                                 "tags": {
                                     "includes": [
                                         "release-v0", "release-v1"
                                     ],
                                     "excludes": [
                                         "release-v2"
                                     ]
                                 }
                             }
                         ]
                     }
                 }
             ]
     ```

Usar uma conexão compartilhada com outra conta

Você pode criar e gerenciar uma conexão compartilhada usando AWS RAM. Isso permite que as conexões sejam compartilhadas entre Contas da AWS para acesso a repositórios de terceiros. Isso permite que uma única conexão seja usada em pipelines do CodePipeline em todas as contas, ao mesmo tempo em que reduz a necessidade de usuários gerenciarem e administrarem conexões à parte em cada conta.

Para usar conexões compartilhadas no CodePipeline, faça o seguinte.

- Crie uma conexão usando o console do Developer Tools em Configurações. Consulte Criar uma conexão.
- Configure o compartilhamento de recursos usando AWS RAM. Consulte Compartilhar conexões com Contas da AWS.
- Ao usar o assistente Criar pipeline do console do CodePipeline ou a página Editar ação para escolher o provedor de conexão, como a opção do provedor Bitbucket, você pode escolher a conexão que foi compartilhada com a conta de destino.

Automatizar a inicialização de pipelines usando gatilhos e filtragem

Os gatilhos permitem configurar o pipeline para iniciar em um determinado tipo de evento ou tipo de evento filtrado, como quando uma alteração em uma ramificação específica ou solicitação pull é detectada. Os acionadores são configuráveis para ações de origem com conexões que usam a `CodeStarSourceConnection` ação em CodePipeline GitHub, como Bitbucket e. GitLab Para ter mais informações sobre ações de origem que usam conexões, consulte Adicione provedores de origem terceirizados aos pipelines usando AWS CodeConnections.

As ações de origem, como CodeCommit e S3, usam a detecção automática de alterações para iniciar os pipelines quando uma alteração é feita. Para obter mais informações, consulte CodeCommit ações de origem e EventBridge.

Você especifica gatilhos usando o console ou a CLI.

Você especifica os tipos de filtro da seguinte forma:

- Sem filtro

  Esta configuração de gatilho inicia o pipeline em qualquer envio para a ramificação padrão especificada como parte da configuração da ação.
- Especificar filtro

  Adicione um filtro que inicia o pipeline em um filtro específico, como em nomes de ramificações para um push de código, e busca a confirmação exata. Isso também configura o pipeline para não iniciar automaticamente em nenhuma alteração.
  - Push
    - As combinações de filtro válidas são:
      - Tags do Git

        Incluir ou excluir
      - filiais

        Incluir ou excluir
      - ramificações + caminhos de arquivo

Incluir ou excluir

- Solicitação pull
  - As combinações de filtro válidas são:
    - filiais

      Incluir ou excluir
    - ramificações + caminhos de arquivo

      Incluir ou excluir
- Não detecte alterações

  Isso não adiciona um gatilho, e o pipeline não inicia automaticamente em nenhuma alteração.

A tabela a seguir fornece opções de filtro válidas para cada tipo de evento. A tabela também mostra quais configurações de gatilho são padronizadas como verdadeiras ou falsas para detecção automática de alterações na configuração da ação.

Configurações do gatilho	Tipo de evento	Opções de filtro	Detectar alterações
Adicionar um gatilho: sem filtro	nenhuma	nenhuma	true
Adicionar um gatilho: filtro no envio de código	evento push	Etiquetas Git, ramificações, caminhos de arquivo	false
Adicionar um gatilho: filtro para solicitações pull	solicitações pull	ramificações, caminhos de arquivo	false
Sem gatilho: não detectar	nenhuma	nenhuma	false

> **Note**
>
> Este tipo de gatilho usa detecção automatizada de alterações (como o tipo de gatilho `Webhook`). Os provedores de ação de origem que usam esse tipo de gatilho são conexões configuradas para envio de código (Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com e GitLab autogerenciadas).

Para obter definições de campo e referência adicional para acionadores, consulte

Para conferir uma lista de definições de campo na estrutura JSON, consulte `triggers`.

Para filtragem, padrões de expressão regular no formato glob são compatíveis conforme detalhado em Trabalhar com padrões glob na sintaxe.

> **Note**
>
> Em certos casos, para pipelines com gatilhos filtrados por caminhos de arquivos, o pipeline pode não iniciar quando uma ramificação com um filtro de caminho de arquivo é criada pela primeira vez. Para obter mais informações, consulte Pipelines configurados com conexões que filtram gatilhos por caminhos de arquivos podem falhar ao iniciar durante a criação da ramificação..

Considerações sobre filtros de gatilho

As seguintes considerações se aplicam ao usar gatilhos.

- Você não pode adicionar mais de um acionador por ação de origem.
- Você pode adicionar vários tipos de filtro a um acionador. Para ver um exemplo, consulte 4: um acionador com dois tipos de filtro push com inclusões e exclusões conflitantes.
- Em gatilhos com filtros de ramificação e caminhos de arquivos, ao enviar a ramificação pela primeira vez, o pipeline não será acionado devido à indisponibilidade da lista de arquivos alterados na nova ramificação.
- A mesclagem de um pull request pode acionar duas execuções de pipeline, nos casos em que as configurações de acionador push (filtro de ramificações) e pull request (filtro de ramificações) se cruzam.

- Para um filtro que acione o pipeline em eventos pull request, para o tipo de evento pull request Fechado, o provedor de repositório de terceiros da conexão pode ter um status à parte para um evento de mesclagem. Por exemplo, no Bitbucket, o evento Git de uma mesclagem não é um evento de encerramento pull request. No entanto, em GitHub, mesclar uma pull request é um evento de encerramento. Para obter mais informações, consulte Eventos pull request para acionadores por provedor.
- Quando várias ações de origem em um pipeline fazem referência a diferentes ramificações do mesmo repositório por meio de uma conexão, somente uma ramificação aciona o pipeline de forma confiável. A assinatura do webhook da conexão é registrada para a combinação de pipeline e repositório, não por ramificação. Como solução alternativa, use um pipeline separado para cada ramificação.

Eventos pull request para acionadores por provedor

A tabela a seguir apresenta um resumo dos eventos do Git, como o encerramento de pull request, que resultem em tipos de evento pull request por provedor.

	Provedor de repositório para a conexão			
Evento PR para acionador	Bitbucket	GitHub	GHES	GitLab
Abrir: esta opção aciona o pipeline quando uma pull request é criada para o caminho de ramificação/ arquivo.	A criação de um pull request resulta em um evento do Git Aberto.	A criação de um pull request resulta em um evento do Git Aberto.	A criação de um pull request resulta em um evento do Git Aberto.	A criação de um pull request resulta em um evento do Git Aberto.
Atualizar: esta opção aciona o pipeline quando uma revisão pull request é publicada para o caminho de	A publicação de uma atualizaç ão resulta em um evento do Git Atualizado.	A publicação de uma atualizaç ão resulta em um evento do Git Atualizado.	A publicação de uma atualizaç ão resulta em um evento do Git Atualizado.	A publicação de uma atualizaç ão resulta em um evento do Git Atualizado.

	Provedor de repositório para a conexão			
Evento PR para acionador	Bitbucket	GitHub	GHES	GitLab
ramificação/ arquivo.				
Fechado - Essa opção aciona o pipeline quando uma pull request é fechada para o branch/file caminho.	A mesclagem de uma pull request no Bitbucket resulta em um evento do Git Fechado. Importante: o fechamento manual de uma pull request no Bitbucket sem mesclar não resulta em um evento do Git Fechado.	A mesclagem ou o fechamento manual de uma pull request resulta em um evento do Git Fechado.	A mesclagem ou o fechamento manual de uma pull request resulta em um evento do Git Fechado.	A mesclagem ou o fechamento manual de uma pull request resulta em um evento do Git Fechado.

Exemplos de filtros de gatilho

Para uma configuração do Git com filtros para os tipos de eventos push e solicitação pull, os filtros especificados podem entrar em conflito entre si. Veja a seguir exemplos de combinações de filtros válidas para eventos push e solicitação pull. Um acionador pode conter vários tipos de filtro, como dois tipos de filtro push na configuração do acionador, e os tipos de filtro push e pull request usarão uma operação OR entre eles, o que significa que qualquer correspondência iniciará o pipeline. Da mesma forma, cada tipo de filtro pode incluir vários filtros, como filePaths e ramificações; esses filtros usarão um operador AND, o que significa que somente uma correspondência completa iniciará o pipeline. Cada tipo de filtro pode conter inclusões e exclusões, onde as exclusões têm precedência sobre as inclusões. Se uma ramificação ou caminho de arquivo corresponder a um padrão de exclusão, ele não acionará o pipeline, mesmo que também corresponda a um padrão de inclusão. Quando um commit altera vários arquivos, cada arquivo é avaliado de forma independente

em relação ao filtro; se algum arquivo alterado for aprovado (corresponder a uma inclusão e não corresponder a uma exclusão), o pipeline será iniciado. Os nomes dentro da inclusão/exclusão, como nomes de ramificação, usam uma operação OR. A lista a seguir resume as operações de cada parte do objeto de configuração do Git.

Para obter uma lista de definições de campo na estrutura JSON e uma referência detalhada para inclusões e exclusões, consulte `triggers`.

Example 1: um tipo de filtro com filtros para ramificações e caminhos de arquivo (operação AND)

Para um único tipo de filtro, como pull request, você pode combinar filtros e esses filtros usarão um operador AND, o que significa que somente uma correspondência completa iniciará o pipeline. O exemplo a seguir mostra uma configuração do Git para um tipo de evento push com dois filtros diferentes (`filePaths` e `branches`). No exemplo a seguir, `filePaths` será usado com o operador E e com `branches`:

```
{
  "filePaths": {
    "includes": ["common/**/*.js"]
  },
  "branches": {
    "includes": ["feature/**"]
  }
}
```

Com a configuração do Git acima, este exemplo mostra um evento que iniciará a execução do pipeline porque o uso do operador E foi bem-sucedido. Em outras palavras, o caminho do arquivo `common/app.js` é incluído para o filtro, que inicia o pipeline como AND mesmo que a ramificação `refs/heads/feature/triggers` especificada não tenha tido impacto.

```
{
  "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "common/app.js"
      ]
      ...
    }
```

```
  ]
}
```

O exemplo a seguir mostra um evento de um acionador com a configuração acima que não iniciará a execução do pipeline porque a ramificação consegue filtrar, mas o caminho do arquivo não.

```
{
   "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "src/Main.java"
      ]
      ...
    }
  ]
}
```

Example 2: As exclusões têm precedência sobre as inclusões

Em um único filtro, as exclusões têm precedência sobre as inclusões. O exemplo a seguir mostra uma configuração do Git com um único filtro (`branches`) dentro do objeto de configuração. Isso significa que se uma ramificação corresponder a um padrão de exclusão (`feature-branch`no exemplo), o pipeline não será acionado, mesmo que também corresponda a um padrão de inclusão. Se o padrão de inclusão corresponder e nenhum padrão de exclusão corresponder, como para a `main` ramificação, o pipeline será acionado.

Para o seguinte JSON de exemplo:

- O envio de uma confirmação para a ramificação `main` acionará o pipeline
- O envio de uma confirmação para a ramificação `feature-branch` não acionará o pipeline.

```
{
  "branches": {
      "Includes": [
          "main"
      ],
      "Excludes": [
```

```
            "feature-branch"
        ]
    }
```

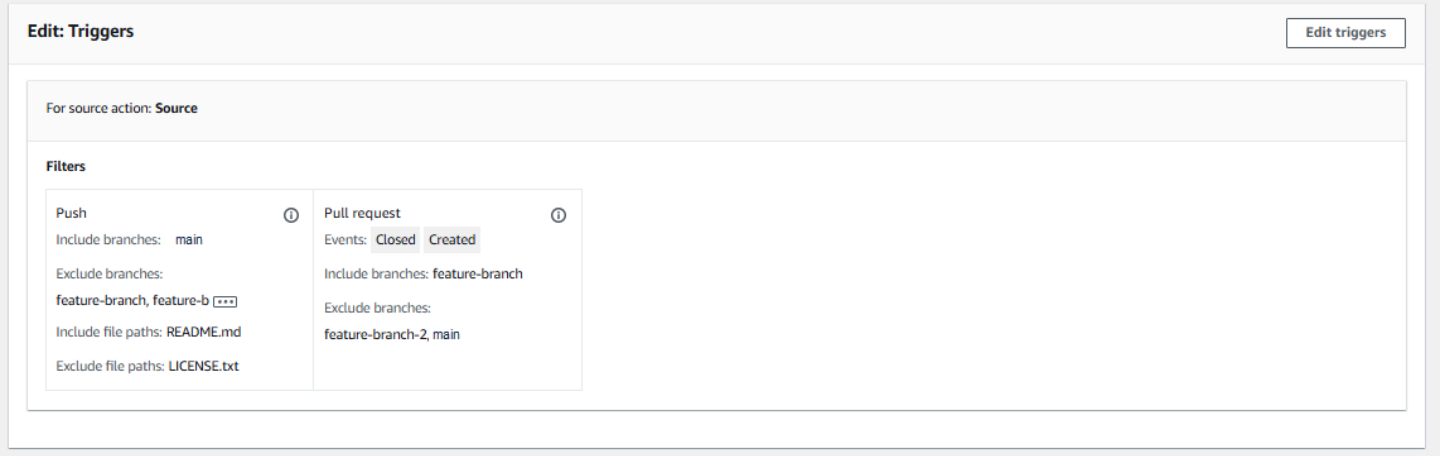
Example 3: Um gatilho com tipos de filtro de solicitação push e pull (operação OR), filtros para caminhos e ramificações de arquivos (operação AND) e includes/excludes (as exclusões têm precedência)

Os objetos de configuração do acionador, como um acionador que contém um tipo de evento push e um tipo de evento pull request, usam uma operação OR entre os dois tipos de evento. O exemplo a seguir mostra uma configuração de acionador com um tipo de evento push com a ramificação `main` incluída e um tipo de evento pull request com a mesma ramificação `main` excluída. Além disso, o tipo de evento push tem um caminho de arquivo `LICENSE.txt` excluído e um caminho de arquivo `README.MD` incluído. Para o segundo tipo de evento, uma pull request que está na ramificação `Closed` ou `Created` na `feature-branch` ramificação (incluída) inicia o pipeline, e o pipeline não inicia ao criar ou fechar uma pull request na ramificação `feature-branch-2` ou nas `main` ramificações (excluídas). Dentro de cada tipo de evento, as exclusões têm precedência sobre as inclusões. Por exemplo, para um evento de pull request na `feature-branch` ramificação (incluído na pull request), o tipo de evento push exclui a `feature-branch` ramificação, portanto, o push não acionará o pipeline.

Para o exemplo a seguir,

- O envio de uma confirmação para a ramificação `main` (incluída) do caminho do arquivo `README.MD` (incluído) acionará o pipeline.
- Na ramificação `feature-branch` (excluída), o envio de uma confirmação não acionará o pipeline.
- Na ramificação incluída, a edição do caminho do arquivo `README.MD` (incluído) aciona o pipeline.
- Na ramificação incluída, editar somente o caminho do `LICENSE.TXT` arquivo (excluído) não aciona o pipeline. No entanto, se o mesmo commit também mudar `README.MD` (incluído), o pipeline será acionado porque cada arquivo é avaliado de forma independente.
- Na `feature-branch` ramificação, fechar uma pull request acionará o pipeline porque `feature-branch` está incluído no tipo de evento do pull request e no tipo de evento CLOSED corresponde.

A imagem a seguir mostra a configuração.



Este é o JSON exemplo da configuração.

```
"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"
                            ]
                        }
                    }
                ]
            }
        }
    ]
},
```

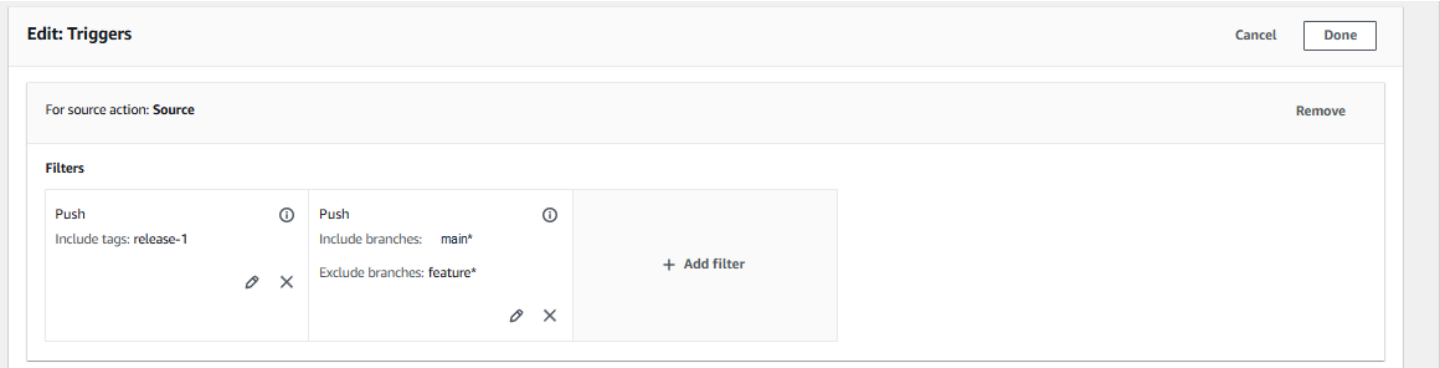
Example 4: um acionador com dois tipos de filtro push com inclusões e exclusões conflitantes

A imagem a seguir mostra um tipo de filtro push especificando o filtro na tag `release-1` (incluída). Um segundo tipo de filtro por push é adicionado especificando a filtragem na ramificação `main` (incluída) e não iniciar um envio por push para as ramificações `feature*` (excluídas).

Para o seguinte exemplo:

- Pressionar uma liberação da etiqueta `release-1` (incluída para o primeiro filtro de pressão) na `feature-branch` ramificação (excluída do segundo filtro de pressão) acionará a tubulação porque os dois tipos de filtro de pressão usam uma operação OR entre eles e o primeiro filtro de pressão (etiqueta`release-1`) corresponde. `feature*`
- O envio por push uma liberação da ramificação `main` (incluída para o segundo filtro Push) iniciará o pipeline.

O exemplo a seguir da página Editar mostra os dois tipos de filtros push e as configurações para inclusões e exclusões.



Este é o JSON exemplo da configuração.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "tags": {
                                "includes": [
                                    "release-1"
                                ]
                            }
                        },
                        {
                            "branches": {
                                "includes": [
                                    "main*"
                                ],
                                "excludes": [
                                    "feature*"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    },
```

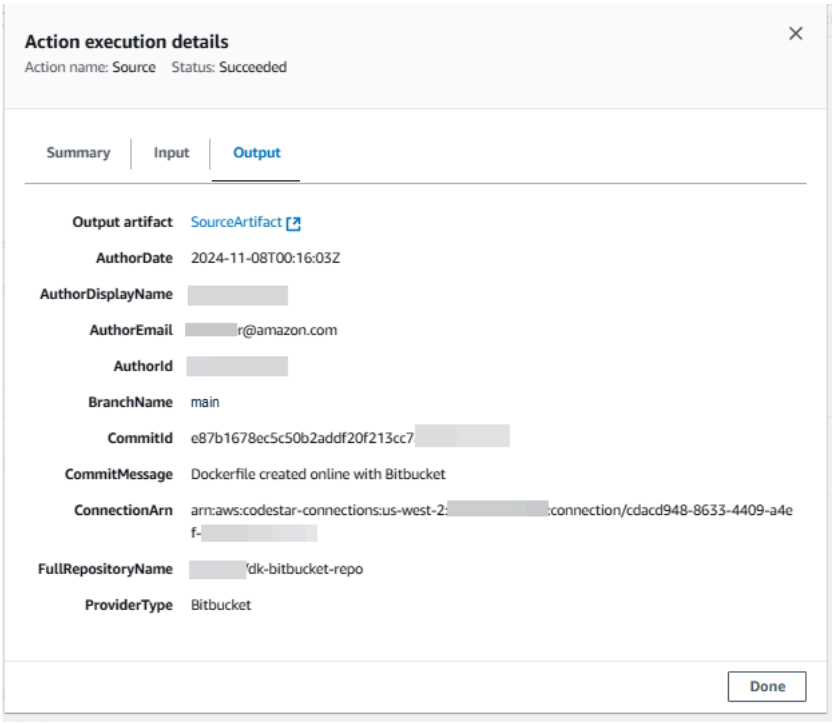
Example 5: Acionador configurado enquanto a configuração de ação padrão BranchName é usada para uma inicialização manual

O campo `BranchName` padrão de configuração da ação define uma única ramificação que será usada quando o pipeline for iniciado manualmente, e os gatilhos com filtros podem ser usados para qualquer ramificação ou ramificações especificadas por você.

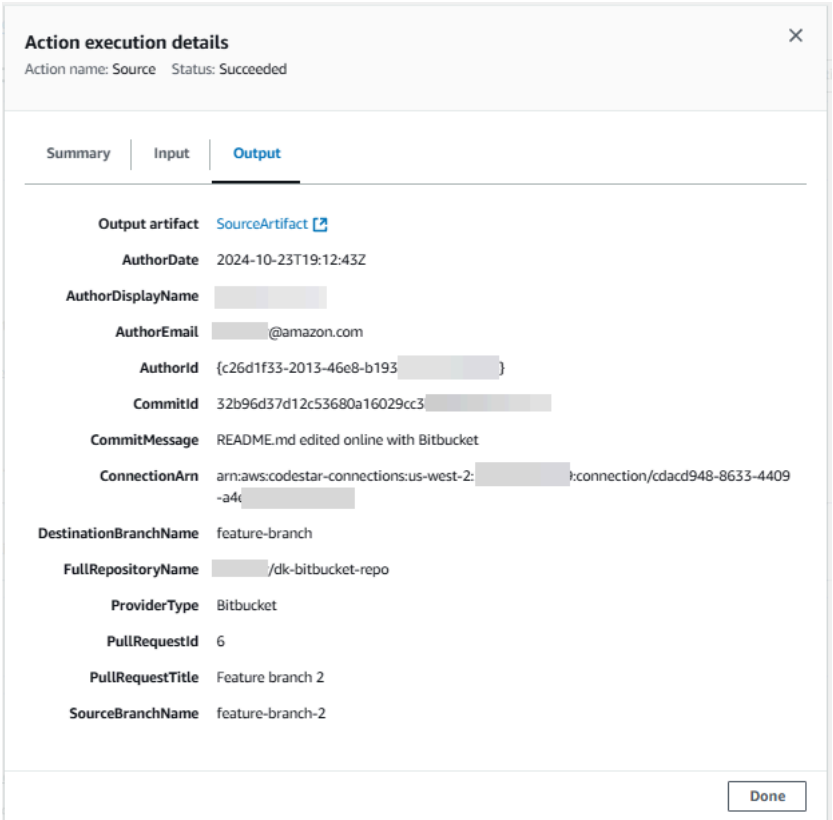
Este é o JSON de exemplo para a configuração de ação que mostra o campo `BranchName`.

```
{
                "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"
                    }
                ],
```

A saída da ação de exemplo a seguir mostra que a ramificação principal padrão foi usada quando o pipeline foi iniciado manualmente.



A saída da ação de exemplo a seguir mostra a pull request e a ramificação que foi usada para o acionador quando filtrado por pull request.



Adicionar gatilho em code push sem filtro

Os gatilhos permitem configurar o pipeline para iniciar em um determinado tipo de evento, como code push ou pull request. Os acionadores são configuráveis para ações de origem com conexões que usam a CodeStarSourceConnection ação em CodePipeline GitHub, como Bitbucket e. GitLab

Adição de um gatilho (console)

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon. com/codesuite/codepipeline/home.

   Os nomes e o status de todos os pipelines associados à sua AWS conta são exibidos.
2. Em Nome, selecione o nome do pipeline que você deseja editar. Caso contrário, use essas etapas no assistente de criação de pipeline.
3. Na página de detalhes do pipeline, selecione Editar.
4. Na página Editar, escolha a ação de origem que você deseja editar. Escolha Editar gatilhos. Opte por adicionar um acionador.
5. Em Tipo de acionador, escolha Sem filtro.

Adicionar gatilho com tipos de eventos code push ou pull request

Você pode configurar filtros para gatilhos de pipeline para que as execuções de pipeline sejam iniciadas para diferentes eventos Git, como push de etiqueta ou ramificação, alterações em caminhos de arquivo específicos, uma solicitação pull aberta em uma ramificação específica e assim por diante. Você pode usar o AWS CodePipeline console ou os update-pipeline comandos create-pipeline e no AWS CLI para configurar filtros de gatilho.

> **Note**
>
> O campo `BranchName` de configuração da ação define uma única ramificação, e os gatilhos com filtros podem ser usados para qualquer ramificação ou ramificações especificadas por você. Para um pipeline no qual os gatilhos sejam usados para filtrar ramificações por push ou pull request, o pipeline não usará a ramificação do campo `BranchName` padrão na configuração da ação. No entanto, a ramificação no campo `BranchName` na configuração da ação é o padrão quando o pipeline é iniciado manualmente. Para ver um exemplo, consulte

5: Acionador configurado enquanto a configuração de ação padrão BranchName é usada para uma inicialização manual.

Você pode especificar filtros para os seguintes tipos de gatilho:

- Push

  Um gatilho push inicia um pipeline quando uma alteração é enviada ao repositório de origem. A execução usará a confirmação da ramificação para a qual você está enviando (ou seja, a ramificação de destino). Você pode filtrar gatilho push em ramificações, caminhos de arquivos ou etiquetas Git.

- Solicitação pull

  Um gatilho de solicitação pull inicia um pipeline quando uma solicitação pull é aberta, atualizada ou fechada no repositório de origem. A execução usará a confirmação da ramificação de origem da qual você está extraindo (ou seja, a ramificação de origem). Você pode filtrar gatilhos de solicitação pull em ramificações e caminhos de arquivo.

  Os tipos de evento compatíveis para pull requests são os seguintes. Todos os outros eventos de solicitação pull são ignorados.

  - Aberto
  - Atualização
  - Fechado (mesclado)

> **Note**
>
> Determinados comportamentos do evento pull request podem mudar de acordo com o provedor. Para obter detalhes, consulte Eventos pull request para acionadores por provedor.

A definição do pipeline permite combinar filtros diferentes na mesma configuração de push de gatilho. Para saber mais sobre a definição do pipeline, consulte Adicionar filtros para tipos de evento push e pull request (CLI). Para obter uma lista das definições de campo, consulte gatilhos na referência de estrutura do Pipeline neste guia.

Tópicos

- Adicione filtros para tipos de evento push e pull request (console)
- Adicionar filtros para tipos de evento push e pull request (CLI)
- Adicione filtros para os tipos de eventos de push e pull request (CloudFormation modelos)

Adicione filtros para tipos de evento push e pull request (console)

Você pode usar o console para adicionar filtros para eventos push e incluir ou excluir ramificações ou caminhos de arquivo.

Adicionar filtros (console)

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes e o status de todos os canais associados à sua AWS conta são exibidos.
2. Em Nome, selecione o nome do pipeline que você deseja editar. Caso contrário, use essas etapas no assistente de criação de pipeline.
3. Na página de detalhes do pipeline, selecione Editar.
4. Na página Editar, escolha a ação de origem que você deseja editar. Escolha Editar gatilhos. Escolha Especificar filtro.
5. Em Tipo de evento, escolha Push entre as opções a seguir.

   - Escolha Push para iniciar o pipeline quando uma alteração for enviada para o repositório de origem. Essa escolha permite que os campos especifiquem filtros para ramificações e caminhos de arquivo ou etiquetas Git.
   - Escolha Solicitação pull para iniciar o pipeline quando uma solicitação pull for aberta, atualizada ou fechada no repositório de origem. Essa escolha permite que os campos especifiquem filtros para ramificações de destino e caminhos de arquivo.
6. Em Push, em Tipo de filtro, escolha uma das opções a seguir.

   - Escolha Ramificação para especificar as ramificações no repositório de origem que o gatilho monitora para saber quando iniciar a execução de um fluxo de trabalho. Em Incluir, insira padrões para nomes de ramificações no formato glob que você deseja especificar para a configuração do gatilho a fim de iniciar o pipeline em alterações nas ramificações especificadas. Em Excluir, insira os padrões de regex para nomes de ramificações no formato glob que você deseja especificar para a configuração do gatilho ignorar e não iniciar seu

pipeline em alterações nas ramificações especificadas. Consulte Trabalhar com padrões glob na sintaxe para obter mais informações.

> **Note**
>
> Se a inclusão e a exclusão tiverem o mesmo padrão de etiqueta, o padrão será excluído.

Você pode usar padrões glob para definir os nomes das ramificações. Por exemplo, use `main*` para combinar todas as ramificações que começam com `main`. Consulte Trabalhar com padrões glob na sintaxe para obter mais informações.

Para um gatilho de push, especifique as ramificações para as quais você está enviando, ou seja, as ramificações de destino. Para um gatilho de pull request, especifique as ramificações de destino para as quais você está abrindo uma pull request.

- (Opcional) Em Caminhos de arquivo, especifique caminhos de arquivo para o gatilho. Insira os nomes em Incluir e Excluir, conforme apropriado.

  Você pode usar padrões glob para definir os nomes de caminhos de arquivos. Por exemplo, use `prod*` para combinar todos os caminhos de arquivo que começam com `prod`. Consulte Trabalhar com padrões glob na sintaxe para obter mais informações.

- Selecione Etiquetas a fim de configurar o gatilho do pipeline para começar com etiquetas Git. Em Incluir, insira padrões para nomes de etiquetas no formato glob que você deseja especificar para a configuração do gatilho a fim de iniciar o pipeline na liberação das etiquetas especificadas. Em Excluir, insira os padrões de regex para nomes de etiquetas no formato glob que você deseja especificar para a configuração do gatilho ignorar e não iniciar o pipeline na liberação das etiquetas especificadas. Se a inclusão e a exclusão tiverem o mesmo padrão de tag, o padrão será excluir o padrão de tag.

7. Em Push, em Tipo de filtro, escolha uma das opções a seguir.

   - Escolha Ramificação para especificar as ramificações no repositório de origem que o gatilho monitora para saber quando iniciar a execução de um fluxo de trabalho. Em Incluir, insira padrões para nomes de ramificações no formato glob que você deseja especificar para a configuração do gatilho a fim de iniciar o pipeline em alterações nas ramificações especificadas. Em Excluir, insira os padrões de regex para nomes de ramificações no formato glob que você deseja especificar para a configuração do gatilho ignorar e não iniciar seu

pipeline em alterações nas ramificações especificadas. Consulte Trabalhar com padrões glob na sintaxe para obter mais informações.

> **Note**
>
> Se a inclusão e a exclusão tiverem o mesmo padrão de etiqueta, o padrão será excluído.

Você pode usar padrões glob para definir os nomes das ramificações. Por exemplo, use `main*` para combinar todas as ramificações que começam com `main`. Consulte Trabalhar com padrões glob na sintaxe para obter mais informações.

Para um gatilho de push, especifique as ramificações para as quais você está enviando, ou seja, as ramificações de destino. Para um gatilho de pull request, especifique as ramificações de destino para as quais você está abrindo uma pull request.

- (Opcional) Em Caminhos de arquivo, especifique caminhos de arquivo para o gatilho. Insira os nomes em Incluir e Excluir, conforme apropriado.

  Você pode usar padrões glob para definir os nomes de caminhos de arquivos. Por exemplo, use `prod*` para combinar todos os caminhos de arquivo que começam com `prod`. Consulte Trabalhar com padrões glob na sintaxe para obter mais informações.
- Escolha Solicitação de pull para definir a configuração de gatilho do pipeline para começar por eventos pull request especificados por você.

Adicionar filtros para tipos de evento push e pull request (CLI)

Você pode atualizar o JSON do pipeline para adicionar filtros para gatilhos.

Para usar o AWS CLI para criar ou atualizar seu pipeline, use o `update-pipeline` comando `create-pipeline` ou.

O exemplo de estrutura JSON a seguir fornece uma referência para as definições de campo em `create-pipeline`.

Para obter uma lista das definições de campo, consulte gatilhos na referência de estrutura do Pipeline neste guia.

```
{
    "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"
                        }
                    }
                ]
            }
        ]
```

```
    }
}
```

Adicione filtros para os tipos de eventos de push e pull request (CloudFormation modelos)

Você pode atualizar o recurso do pipeline CloudFormation para adicionar a filtragem de gatilhos.

O trecho de modelo de exemplo a seguir fornece uma referência YAML para definições de campos de gatilhos. Para obter uma lista das definições de campo, consulte gatilhos na referência de estrutura do Pipeline neste guia.

Para ver um exemplo de modelo completo para uma configuração de fonte de conexão e filtro de gatilho, consulte Pipeline com dois estágios e configuração de gatilho no Guia CloudFormation do usuário.

```
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
```

Adicionar acionador para desativar a detecção de alterações

Os gatilhos permitem configurar o pipeline para iniciar em um determinado tipo de evento, como code push ou pull request. Os gatilhos são configuráveis para ações de origem com conexões que usam a `CodeStarSourceConnection` ação em CodePipeline GitHub, como Bitbucket e. GitLab

Adição de um acionador para desativar a detecção de alterações (console)

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes e o status de todos os canais associados à sua AWS conta são exibidos.

2. Em Nome, selecione o nome do pipeline que você deseja editar. Caso contrário, use essas etapas no assistente de criação de pipeline.
3. Na página de detalhes do pipeline, selecione Editar.
4. Na página Editar, escolha a ação de origem que você deseja editar. Escolha Editar gatilhos. Opte por adicionar um acionador.
5. Em Tipo de acionador, escolha Não detectar alterações.

Iniciar e parar pipelines manualmente

Você pode usar o AWS CodePipeline console ou o AWS CLI para iniciar e parar tubulações manualmente. Existem métodos automatizados para iniciar pipelines com detecção de alterações baseada em eventos, conforme detalhado, ou configuração de gatilho, conforme detalhado.

Tópicos

- Inicie um funil em CodePipeline
- Interrompa a execução de um pipeline em CodePipeline

Inicie um funil em CodePipeline

Cada execução de pipeline pode ser iniciada com base em um gatilho diferente. Cada execução do pipeline pode ter um tipo diferente de gatilho, dependendo de como o pipeline é iniciado. O tipo de gatilho para cada execução é mostrado no histórico de execução de um pipeline. Os tipos de gatilho podem depender do provedor de ação de origem da seguinte forma:

> **Note**
>
> Você não pode especificar mais de um gatilho por ação de origem.

- Criação de pipeline: quando um pipeline é criado, uma execução de pipeline é iniciada automaticamente. Este é o tipo de gatilho `CreatePipeline` no Histórico de execução.
- Alterações nos objetos revisados: esta categoria representa o tipo de gatilho `PutActionRevision` no Histórico de execução.
- Detecção de alterações na ramificação e confirmação para um envio de código por push: esta categoria representa o tipo de gatilho `CloudWatchEvent` no Histórico de execução. Quando uma alteração é detectada em uma confirmação e ramificação de origem no repositório de origem, seu pipeline é iniciado. Este tipo de gatilho usa a detecção automatizada de alterações. Os provedores de ação de origem que usam esse tipo de acionador são S3 e. CodeCommit Este tipo também é usado para um agendamento que inicia seu pipeline. Consulte Iniciar um pipeline de acordo com uma programação.
- Sondagem de alterações de origem: esta categoria representa o tipo de gatilho `PollForSourceChanges` no Histórico de execução. Quando uma alteração é detectada em uma

confirmação e ramificação de origem no repositório de origem por meio da sondagem, seu pipeline é iniciado. Esse tipo de gatilho não é recomendado e deve ser migrado para usar a detecção automatizada de alterações. Os provedores de ação de origem que usam esse tipo de acionador são S3 e. CodeCommit

- Eventos Webhook para fontes de terceiros: esta categoria representa o tipo de gatilho `Webhook` no Histórico de execução. Quando uma alteração é detectada por um evento Webhook, seu pipeline é iniciado. Este tipo de gatilho usa a detecção automatizada de alterações. Os provedores de ação de origem que usam esse tipo de acionador são conexões configuradas para envio de código (Bitbucket Cloud GitHub, GitHub Enterprise Server e GitLab autogerenciado). GitLab.com
- Eventos WebhookV2 para fontes de terceiros: esta categoria representa o tipo de gatilho `WebhookV2` no Histórico de execução. Este tipo destina-se a execuções acionadas com base em gatilhos especificados na definição do pipeline. Quando uma versão com uma tag Git especificada é detectada, seu pipeline é iniciado. Você pode usar tags do Git para marcar uma confirmação com um nome ou outro identificador que ajude outros usuários do repositório a compreender a importância. Você também pode usar tags do Git para identificar uma confirmação específica no histórico de um repositório. Este tipo de gatilho desabilita a detecção automatizada de alterações. Os provedores de ação de origem que usam esse tipo de acionador são conexões configuradas para tags do Git (Bitbucket Cloud GitHub, GitHub Enterprise Server e GitLab.com).
- Iniciando manualmente um pipeline: esta categoria representa o tipo de gatilho `StartPipelineExecution` no Histórico de execução. Você pode usar o console ou o AWS CLI para iniciar um pipeline manualmente. Para mais informações, consulte Iniciar um pipeline manualmente.
- RollbackStage: essa categoria representa o tipo de `RollbackStage` gatilho no histórico de execução. Você pode usar o console ou o AWS CLI para reverter um estágio manual ou automaticamente. Para mais informações, consulte Configuração da reversão de estágio.

Quando você adiciona ao seu pipeline uma ação de origem que usa tipos de gatilho de detecção automatizada de alterações, as ações funcionam com recursos adicionais. A criação de cada ação de origem é detalhada em seções separadas devido a esses recursos adicionais para detecção de alterações. Para obter detalhes sobre cada provedor de origem e os métodos de detecção de alterações necessários à detecção automatizada de alterações, consulte Ações de origem e métodos de detecção de alterações.

Tópicos

- Iniciar um pipeline manualmente

- Iniciar um pipeline de acordo com uma programação
- Iniciar um pipeline com uma substituição da revisão de origem

Iniciar um pipeline manualmente

Por padrão, um pipeline inicia-se automaticamente quando é criado e sempre que uma alteração é feita em um repositório de origem. No entanto, pode ser que você queira executar novamente a revisão mais recente no pipeline. Você pode usar o CodePipeline console ou o start-pipeline-execution comando AWS CLI and para executar manualmente novamente a revisão mais recente por meio do seu pipeline.

Tópicos

- Iniciar um pipeline manualmente (console)
- Iniciar um pipeline manualmente (CLI)

Iniciar um pipeline manualmente (console)

Para iniciar manualmente um pipeline e executar a revisão mais recente por meio de um pipeline

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Em Name, escolha o nome do pipeline que você deseja iniciar.
3. Na página de detalhes do pipeline, escolha Lançar alteração. Se o pipeline estiver configurado para passar parâmetros (variáveis do pipeline), escolha Lançar alteração para abrir a janela Lançar alteração. Em Variáveis de pipeline, no(s) campo(s) das variáveis no nível do pipeline, insira o(s) valor(es) que você deseja passar para a execução desse pipeline. Para obter mais informações, consulte Referência de variáveis.

   Essa ação inicia a revisão mais recente disponível em cada local de origem especificado em uma ação de origem do pipeline.

Iniciar um pipeline manualmente (CLI)

Para iniciar manualmente um pipeline e executar a versão mais recente de um artefato por meio de um pipeline

1. Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use a AWS CLI para executar o comando start-pipeline-execution, especificando o nome do pipeline que você deseja iniciar. Por exemplo, para começar a executar a última alteração em um pipeline chamado *MyFirstPipeline*:

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

   Para iniciar um pipeline em que as variáveis são configuradas no nível do pipeline, use o comando start-pipeline-execution com o argumento --variables opcional para iniciar o pipeline e adicionar as variáveis que serão usadas na execução. Por exemplo, para adicionar uma variável `var1` com um valor 1, use o seguinte comando:

   ```
   aws codepipeline start-pipeline-execution --name MyFirstPipeline --variables
    name=var1,value=1
   ```

2. Para confirmar se tudo deu certo, visualize o objeto retornado. Este comando retorna um ID de execução semelhante a:

   ```
   {
       "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
   }
   ```

   > **Note**
   >
   > Depois de iniciar o pipeline, você pode monitorar seu progresso no CodePipeline console ou executando o get-pipeline-state comando. Para obter mais informações, consulte Visualizar pipelines (console) e Visualizar detalhes e histórico do pipeline (CLI).

Iniciar um pipeline de acordo com uma programação

Você pode configurar uma regra EventBridge para iniciar um funil de acordo com um cronograma.

Crie uma EventBridge regra que programe o início do funil (console)

Para criar uma EventBridge regra com um agendamento como fonte do evento

1. Abra o EventBridge console da Amazon em https://console.aws.amazon.com/events/.
2. No painel de navegação, escolha Regras.
3. Escolha Criar regra e, em Detalhes da regra, escolha Programação.
4. Configure a programação usando uma taxa fixa ou expressão. Para obter mais informações, consulte Schedule Expression for Rules.
5. Em Metas, escolha CodePipeline.
6. Insira o ARN do pipeline para a execução do pipeline de acordo com essa programação.

> **Note**
>
> Você pode encontrar o ARN do pipeline em Configurações no console. Consulte Visualizar o ARN do pipeline e o ARN do perfil de serviço (console).

7. Escolha uma das opções a seguir para criar ou especificar uma função de serviço do IAM que dê EventBridge permissões para invocar o destino associado à sua EventBridge regra (nesse caso, o destino é CodePipeline).
   - Escolha Criar uma nova função para esse recurso específico para criar uma função de serviço que conceda EventBridge permissões para iniciar suas execuções de pipeline.
   - Escolha Usar função existente para inserir uma função de serviço que conceda EventBridge permissões para iniciar suas execuções de pipeline.
8. Escolha Configure details (Configurar detalhes).
9. Na página Configure rule details (Configurar detalhes da regra), informe um nome e uma descrição para a regra e selecione State (Estado) para habilitá-la.
10. Se você estiver satisfeito com a regra, escolha Create rule.

Crie uma EventBridge regra que programe o início do pipeline (CLI)

Para usar o AWS CLI para criar uma regra, chame o put-rule comando, especificando:

- Um nome que identifique de forma exclusiva a regra que você está criando. Esse nome deve ser exclusivo em todos os funis que você cria CodePipeline associados à sua AWS conta.

- A expressão de programação para a regra.

Para criar uma EventBridge regra com um agendamento como fonte do evento

1. Use o comando put-rule e inclua os parâmetros `--name` e `--schedule-expression`.

   Exemplos:

   O exemplo de comando a seguir é usado --schedule-expression para criar uma regra chamada `MyRule2` que filtra EventBridge em um agendamento.

   ```
   aws events put-rule --schedule-expression 'cron(15 10 ? * 6L 2002-2005)' --name
    MyRule2
   ```

2. Para adicionar CodePipeline como destino, chame o put-targets comando e inclua os seguintes parâmetros:

   - O parâmetro `--rule` é usado com `rule_name` criado por meio de put-rule.
   - O parâmetro `--targets` é usado com o `Id` da lista do destino na lista de destinos e o `ARN` do pipeline de destino.

   O exemplo de comando a seguir especifica que, para a regra chamada `MyCodeCommitRepoRule`, o `Id` do destino é composto do número um, indicando que, em uma lista de destinos para a regra, esse é o destino 1. O exemplo de comando também especifica um exemplo `ARN` para o pipeline. O pipeline é iniciado quando uma alteração é feita no repositório.

   ```
   aws events put-targets --rule MyCodeCommitRepoRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

3. Conceda permissões EventBridge para usar CodePipeline para invocar a regra. Para obter mais informações, consulte Usando políticas baseadas em recursos para a Amazon. EventBridge

   a. Use o exemplo a seguir para criar a política de confiança que permita ao EventBridge assumir a função de serviço. Chame-o de `trustpolicyforEB.json`.

      JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Use o comando a seguir para criar a função `Role-for-MyRule` e anexar a política de confiança.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Crie o JSON de política de permissões, conforme mostrado neste exemplo, para o pipeline denominado `MyFirstPipeline`. Nomeie a politica de permissões `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Use o comando a seguir para anexar a nova política de permissões `CodePipeline-Permissions-Policy-for-EB` à função `Role-for-MyRule` criada.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforCWE.json
```

Iniciar um pipeline com uma substituição da revisão de origem

É possível usar substituições para iniciar um pipeline com um ID de revisão de origem específico fornecido para a execução do pipeline. Por exemplo, se você quiser iniciar um pipeline que processará um ID de confirmação específico de sua CodeCommit origem, você pode adicionar o ID de confirmação como uma substituição ao iniciar seu funil.

> **Note**
>
> Você também pode criar uma substituição de origem usando a entrada de transformação de entrada para usar o `revisionValue` in EventBridge para seu evento de pipeline, em que `revisionValue` é derivado da variável de evento de origem para sua chave de objeto, confirmação ou ID de imagem. Para saber mais, consulte a etapa opcional da entrada da transformação de entrada incluída nos procedimentos em Ações e recursos de origem do Amazon ECR EventBridge, Conexão a ações de origem do Amazon S3 com uma fonte habilitada para eventos ou CodeCommit ações de origem e EventBridge.

Há quatro tipos de revisão de origem para `revisionType`:

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

> **Note**
>
> Para os tipos de revisões de origem `COMMIT_ID` e `IMAGE_DIGEST`, o ID de revisão de origem se aplica a todo o conteúdo no repositório, em todas as ramificações.

> **Note**
>
> Para os `S3_OBJECT_KEY` tipos `S3_OBJECT_VERSION_ID` e de revisões de fonte, qualquer um dos tipos pode ser usado de forma independente ou pode ser usado em conjunto para substituir a fonte por um ID de versão específico ObjectKey . Para `S3_OBJECT_KEY`, o parâmetro de configuração `AllowOverrideForS3ObjectKey` precisa ser definido como `true`. Para ter mais informações sobre os parâmetros de configuração de origem do S3, consulte Parâmetros de configuração.

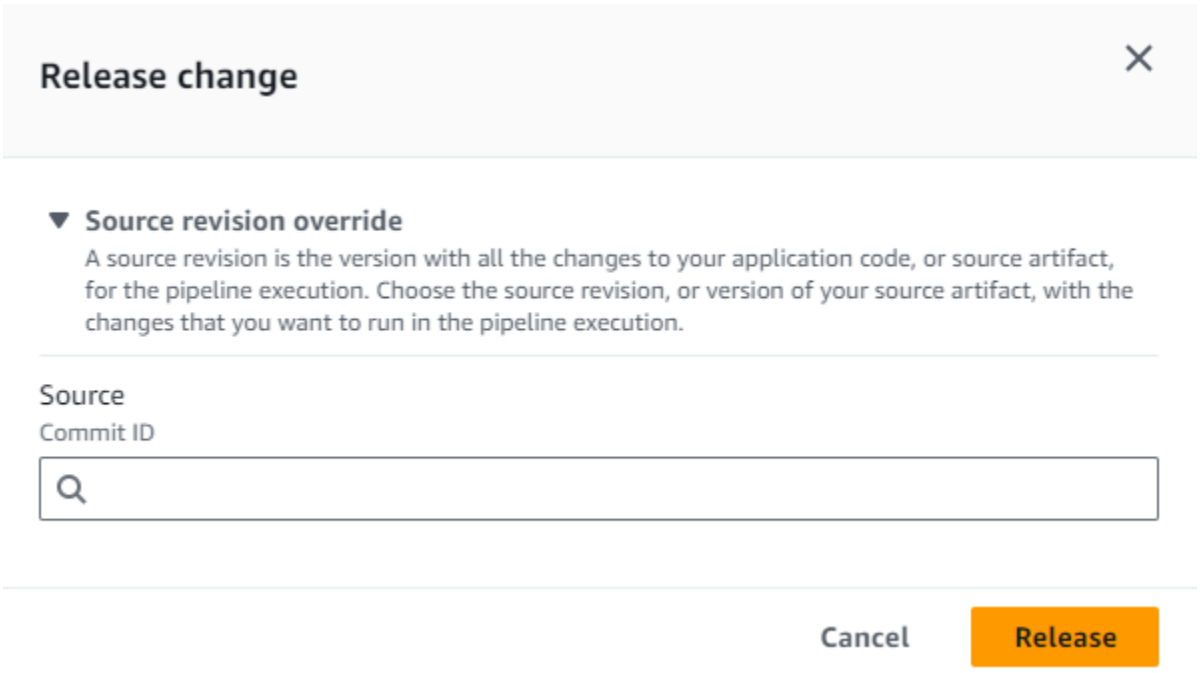
Tópicos

- Iniciar um pipeline com uma substituição de revisão de origem (console)
- Iniciar um pipeline com uma substituição da revisão de origem (CLI)

Iniciar um pipeline com uma substituição de revisão de origem (console)

Para iniciar manualmente um pipeline e executar a revisão mais recente por meio de um pipeline

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Em Name, escolha o nome do pipeline que você deseja iniciar.
3. Na página de detalhes do pipeline, escolha Lançar alteração. Selecionar Lançar alteração abre a janela Lançar alteração. Em Substituição da revisão de origem, selecione a seta para expandir o campo. Em Origem, insira o ID da revisão de origem. Por exemplo, se seu funil tiver uma CodeCommit origem, escolha o ID do commit no campo que você deseja usar.



Iniciar um pipeline com uma substituição da revisão de origem (CLI)

Como iniciar manualmente um pipeline e executar o ID de revisão de origem especificado para um artefato por meio de um pipeline

1. Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use a AWS CLI para executar o comando start-pipeline-execution, especificando o nome do pipeline que você deseja iniciar. Também é necessário o argumento --source-revisions para fornecer o ID da revisão de origem. A revisão de origem é composta por actionName, revisionType e revisionValue. Os valores válidos de revisionType são `COMMIT_ID | IMAGE_DIGEST | S3_OBJECT_VERSION_ID | S3_OBJECT_KEY`.

   No exemplo a seguir, para começar a executar a alteração especificada por meio de um pipeline chamadocodecommit-pipeline, o comando a seguir especifica um nome de ação de origem de Source, um tipo de `COMMIT_ID` revisão e um ID de commit `de78a25c18755ccac3f2a9eec099dEXAMPLE`.

   ```
   aws codepipeline start-pipeline-execution --name codecommit-pipeline --source-
   revisions
    actionName=Source,revisionType=COMMIT_ID,revisionValue=78a25c18755ccac3f2a9eec099dEXAMPLE
    --region us-west-1
   ```

2. Para confirmar se tudo deu certo, visualize o objeto retornado. Este comando retorna um ID de execução semelhante a:

```
{
    "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
}
```

> **Note**
>
> Depois de iniciar o pipeline, você pode monitorar seu progresso no CodePipeline console ou executando o get-pipeline-state comando. Para obter mais informações, consulte Visualizar pipelines (console) e Visualizar detalhes e histórico do pipeline (CLI).

Interrompa a execução de um pipeline em CodePipeline

Quando a execução de um pipeline começa, ela entra em um estágio de cada vez e bloqueia o estágio enquanto todas as execuções de ação no estágio estão em andamento. Essas ações em andamento devem ser manipuladas de forma que, quando a execução do pipeline for interrompida, as ações sejam concluídas ou abandonadas.

Há duas maneiras de interromper a execução de um pipeline:

- Parar e esperar: AWS CodePipeline espera para interromper a execução até que todas as ações em andamento sejam concluídas (ou seja, as ações tenham um `Failed` status `Succeeded` or). Essa opção preserva as ações em andamento. A execução fica em um estado `Stopping` até que as ações em andamento sejam concluídas. Depois, a execução fica em um estado `Stopped`. O estágio é desbloqueado após a conclusão das ações.

  Se optar por interromper e aguardar, e mudar de ideia enquanto a execução ainda está em um estado `Stopping`, você poderá optar por abandonar.
- Parar e abandonar: AWS CodePipeline interrompe a execução sem esperar que as ações em andamento sejam concluídas. A execução fica em um estado `Stopping` por um tempo muito curto enquanto as ações em andamento são abandonadas. Depois que a execução é interrompida, a execução da ação ficará em estado `Abandoned` enquanto a execução do pipeline estiver em estado `Stopped`. O estágio é desbloqueado.

Para a execução de um pipeline em um estado `Stopped`, as ações no estágio em que a execução foi interrompida podem ser repetidas.

> **Warning**
>
> Essa opção pode levar a tarefas com falha ou a tarefas fora de sequência.

Tópicos

- Interromper a execução de um pipeline (console)
- Interromper uma execução de entrada (console)
- Interromper a execução de um pipeline (CLI)
- Interromper uma execução de entrada (CLI)

Interromper a execução de um pipeline (console)

O console do pode ser usado para interromper a execução de um pipeline. Escolha uma execução e, em seguida, escolha o método para interromper a execução do pipeline.

> **Note**
>
> Você também pode interromper a execução de um pipeline que seja uma execução de entrada. Para saber mais sobre como interromper uma execução de entrada, consulte Interromper uma execução de entrada (console).

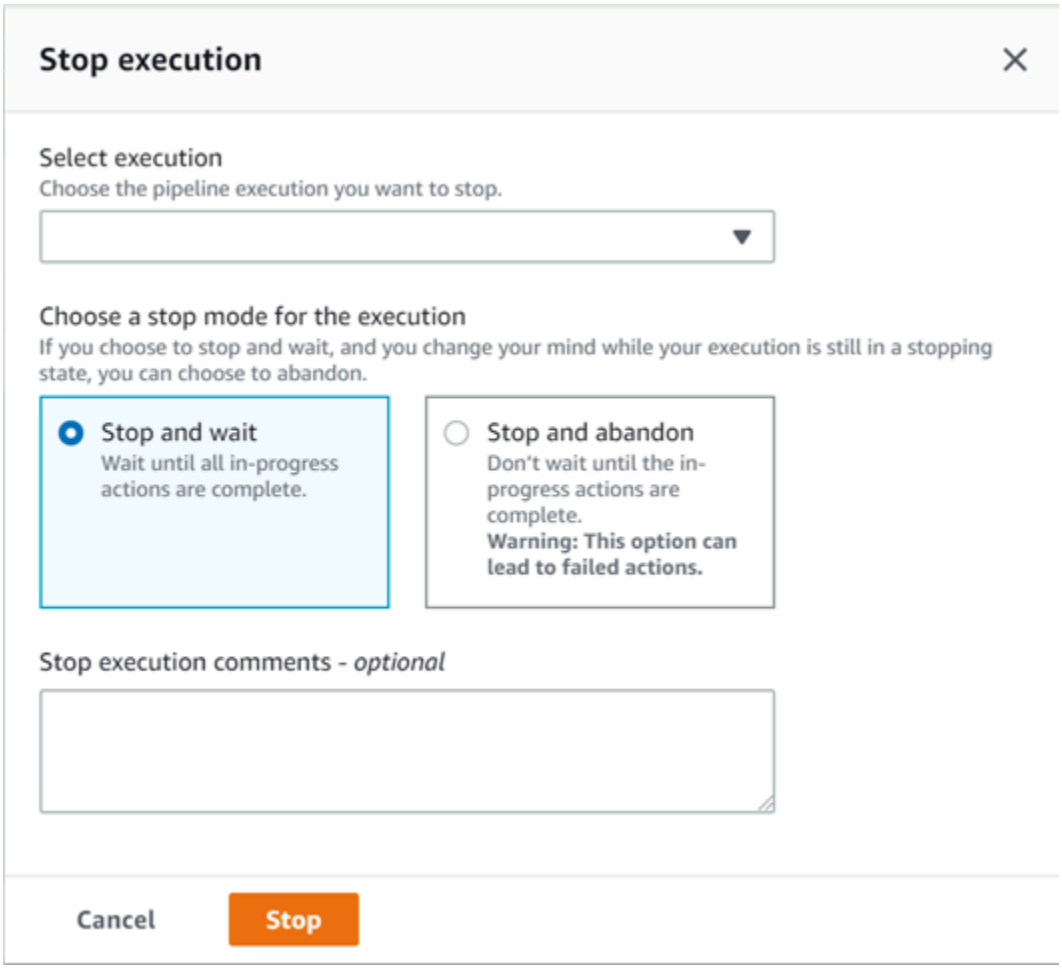
1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Execute um destes procedimentos:

> **Note**
>
> Antes de interromper uma execução, recomendamos que você desative a transição na frente do estágio. Dessa forma, quando o estágio for desbloqueado devido à execução interrompida, ele não aceitará a execução de um pipeline subsequente.

- Em Name (Nome), selecione o nome do pipeline com a execução que deseja interromper. Na página de detalhes do pipeline, selecione Stop execution (Interromper execução).
- Selecione Visualizar histórico. Na página do histórico, selecione Stop execution (Interromper execução).

3. Na página Stop execution (Interromper execução), em Select execution (Escolher execução), selecione a execução que deseja interromper.

> **Note**
>
> A execução só será exibida se ainda estiver em andamento. As execuções que já estão concluídas não serão exibidas.



4. Em Select an action to apply to execution (Selecionar uma ação a ser aplicada à execução), escolha uma das seguintes opções:

   - Para garantir que a execução não seja interrompida até que todas as ações em andamento sejam concluídas, selecione Stop and wait (Interromper e aguardar).

     > **Note**
     >
     > Não será possível optar por interromper e aguardar se a execução já estiver em um estado Stopping (Interrompendo) mas você poderá optar por interromper e abandonar.

   - Para interromper sem aguardar que as ações em andamento sejam concluídas, selecione Stop and abandon (Interromper e abandonar).

     > **Warning**
     >
     > Essa opção pode levar a tarefas com falha ou a tarefas fora de sequência.

5. (Opcional) Insira comentários. Esses comentários, além do status de execução, são exibidos na página do histórico da execução.
6. Escolha Parar.

   > **Important**
   >
   > Esta ação não pode ser desfeita.

7. Exiba o status de execução na visualização do pipeline da seguinte forma:

   - Se você optar por interromper e aguardar, a execução selecionada continuará até que as ações em andamento sejam concluídas.
     - A mensagem de banner de sucesso é exibida na parte superior do console.
     - No estágio atual, as ações em andamento continuam em um estado `InProgress`. Enquanto as ações estão em andamento, a execução do pipeline fica em um estado `Stopping`.

       Depois que as ações são concluídas (ou seja, a ação falha ou é bem-sucedida), a execução do pipeline muda para um estado `Stopped` e a ação muda para um estado `Failed` ou `Succeeded`. Também é possível visualizar o estado da ação na página de detalhes da

execução. É possível visualizar o status da execução na página do histórico da execução ou na página de detalhes da execução.

- A execução do pipeline muda para um estado `Stopping` brevemente e, depois, muda para um estado `Stopped`. É possível visualizar o status da execução na página do histórico da execução ou na página de detalhes da execução.

- Se você optar por interromper e abandonar, a execução não aguardará a conclusão das ações em andamento.
  - A mensagem de banner de sucesso é exibida na parte superior do console.
  - No estágio atual, as ações em andamento mudam para um status de `Abandoned`. Também é possível visualizar o status da ação na página de detalhes da execução.
  - A execução do pipeline muda para um estado `Stopping` brevemente e, depois, muda para um estado `Stopped`. É possível visualizar o status da execução na página do histórico da execução ou na página de detalhes da execução.

É possível visualizar o status de execução do pipeline na visualização do histórico de execução e na visualização detalhada do histórico.

Interromper uma execução de entrada (console)

O console do pode ser usado para interromper uma execução de entrada. Uma execução de entrada é uma execução de pipeline que está aguardando para entrar em um estágio em que a transição foi desabilitada. Quando a transição é habilitada, uma execução de entrada `InProgress` continua e entra no estágio. Uma execução de entrada `Stopped` não entra no estágio.

> **Note**
>
> Depois que uma execução de entrada for interrompida, ela não poderá ser repetida.

Se uma execução de entrada não for exibida, não haverá execuções pendentes em uma transição de estágio desabilitada.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes de todos os canais associados à sua AWS conta serão exibidos.

2. Escolha o nome do pipeline para o qual você deseja interromper a execução de entrada. Execute um destes procedimentos:

   - Na visualização Pipeline, escolha o ID de execução de entrada e, em seguida, opte por interromper a execução.
   - Escolha o pipeline e, em seguida, escolha Visualizar histórico. No histórico de execução, escolha o ID de execução de entrada e, em seguida, opte por interromper a execução.

3. No modal Interromper a execução, siga as etapas na seção acima para selecionar o ID de execução e especificar o método de interrupção.

   Use o comando get-pipeline-state para visualizar o status da execução de entrada.

Interromper a execução de um pipeline (CLI)

Para usar o AWS CLI para interromper manualmente um pipeline, use o stop-pipeline-execution comando com os seguintes parâmetros:

- ID de execução (obrigatório)
- Comentários (opcional)
- Nome do pipeline (obrigatório)
- Indicador Abandon (Abandonar) (opcional, o padrão é falso)

Formato do comando:

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --pipeline-
execution-id Execution_ID [--abandon | --no-abandon] [--reason STOP_EXECUTION_REASON]
```

1. Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows).
2. Para interromper a execução de um pipeline, escolha uma das seguintes opções:

   - Para garantir que a execução não seja interrompida até que todas as ações em andamento sejam concluídas, escolha interromper e aguardar. É possível fazer isso incluindo o parâmetro no-abandon. Se você não especificar o parâmetro, o comando padrão será interromper e aguardar. Use o AWS CLI para executar o stop-pipeline-execution comando, especificando o nome do pipeline e o ID de execução. Por exemplo, para interromper um pipeline chamado *MyFirstPipeline* com a opção interromper e aguardar especificada:

```
aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
pipeline-execution-id d-EXAMPLE --no-abandon
```

Por exemplo, para interromper um pipeline chamado *MyFirstPipeline*, assumindo como padrão a opção interromper e aguardar, e optar por incluir comentários:

```
aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
pipeline-execution-id d-EXAMPLE --reason "Stopping execution after the build
 action is done"
```

> **Note**
>
> Não é possível optar por interromper e aguardar se a execução já estiver em um estado Stopping (Interrompendo). É possível optar por interromper e abandonar uma execução que já está em um estado Stopping (Interrompendo).

- Para interromper sem esperar que as ações em andamento sejam concluídas, escolha interromper e abandonar. Inclua o parâmetro abandon. Use o AWS CLI para executar o stop-pipeline-execution comando, especificando o nome do pipeline e o ID de execução.

  Por exemplo, para interromper um pipeline chamado *MyFirstPipeline*, especificando a opção de abandonar e optando por incluir comentários:

```
aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
pipeline-execution-id d-EXAMPLE --abandon --reason "Stopping execution for a bug
 fix"
```

Interromper uma execução de entrada (CLI)

Você pode usar a CLI para interromper uma execução de entrada. Uma execução de entrada é uma execução de pipeline que está aguardando para entrar em um estágio em que a transição foi desabilitada. Quando a transição é habilitada, uma execução de entrada `InProgress` continua e entra no estágio. Uma execução de entrada `Stopped` não entra no estágio.

> **Note**
>
> Depois que uma execução de entrada for interrompida, ela não poderá ser repetida.

Se uma execução de entrada não for exibida, não haverá execuções pendentes em uma transição de estágio desabilitada.

Para usar o AWS CLI para interromper manualmente uma execução de entrada, use o stop-pipeline-execution comando com os seguintes parâmetros:

- ID de execução de entrada (obrigatório)
- Comentários (opcional)
- Nome do pipeline (obrigatório)
- Indicador Abandon (Abandonar) (opcional, o padrão é falso)

Formato do comando:

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --
pipeline-execution-id Inbound_Execution_ID [--abandon | --no-abandon] [--
reason STOP_EXECUTION_REASON]
```

Siga as etapas do procedimento acima para inserir o comando e especificar o método de interrupção.

Use o comando get-pipeline-state para visualizar o status da execução de entrada.

Exibir histórico e definir o modo para execuções de pipeline

Para analisar o progresso do pipeline, você pode visualizar o histórico de execução do pipeline e da ação. Para configurar o método de liberação para processar execuções de pipeline, defina o modo de execução.

Tópicos

- Exibir execuções em CodePipeline
- Definir ou alterar o modo de execução do pipeline

Exibir execuções em CodePipeline

Você pode usar o AWS CodePipeline console ou o AWS CLI para visualizar o status da execução, visualizar o histórico da execução e tentar novamente estágios ou ações que falharam.

Tópicos

- Visualizar o histórico de execução do pipeline (console)
- Visualizar o status de execução (console)
- Visualizar uma execução de entrada (console)
- Visualizar revisões de origem de execução do pipeline (console)
- Visualizar execuções da ação (console)
- Visualizar informações sobre artefatos e armazenamento de artefatos da ação (console)
- Visualizar detalhes e histórico do pipeline (CLI)

Visualizar o histórico de execução do pipeline (console)

Você pode usar o CodePipeline console para ver uma lista de todos os canais em sua conta. Você também pode ver os detalhes de cada pipeline, incluindo quando as ações foram executadas pela última vez no pipeline, se uma transição entre estágios está ativada ou desativada, se alguma ação falhou e outras informações. Você também pode visualizar uma página de histórico que mostra os detalhes de todas as execuções de pipeline para as quais o histórico foi registrado.

> **Note**
>
> Ao alternar entre modos de execução específicos, a visualização e o histórico do pipeline podem mudar. Para obter mais informações, consulte Definir ou alterar o modo de execução do pipeline.

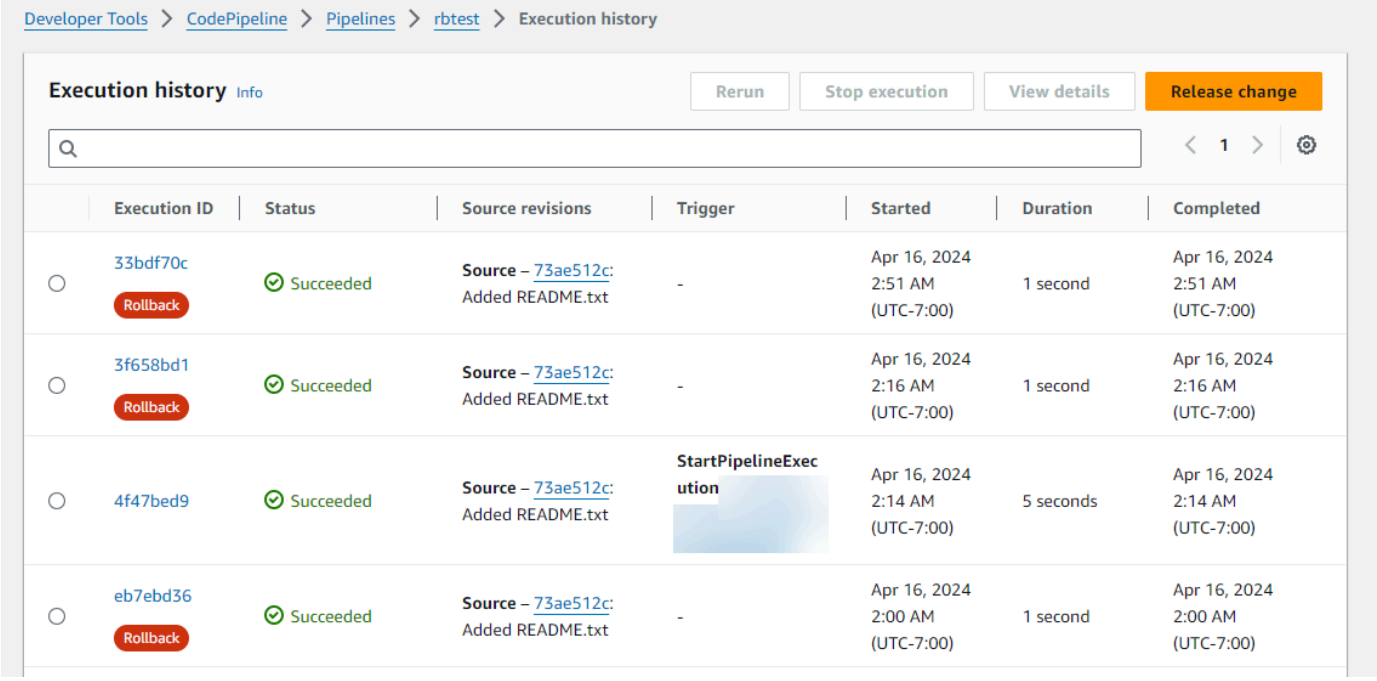
O histórico de execução é mantido por até 12 meses.

Você pode usar o console para visualizar o histórico de execuções em um pipeline, incluindo o status, as revisões de origem e os detalhes de tempo para cada execução.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes de todos os funis associados à sua AWS conta são exibidos, junto com seus status.
2. Em Nome, selecione o nome do pipeline.
3. Selecione Visualizar histórico.

> **Note**
>
> Em um pipeline no modo de execução PARALLEL, a visualização principal não mostra a estrutura do pipeline nem as execuções que estão em andamento. Em um pipeline no modo de execução PARALLEL, a estrutura do pipeline pode ser acessada selecionando o ID da execução desejada na página de histórico de execução. No painel de navegação à esquerda, selecione Histórico, escolha o ID da execução paralela e visualize o pipeline na guia Visualização.



4. Visualize o status, as revisões da origem, os detalhes das alterações e os triggers relacionados a cada execução do pipeline. As execuções de pipeline que foram revertidas exibirão o tipo de execução Reversão na tela de detalhes no console. Para a execução com falha que acionou a reversão automática, o ID da execução com falha é exibido.
5. Escolha uma execução. A visualização detalhada mostra detalhes da execução, a guia Linha do tempo, a guia Visualização e a guia Variáveis. Os valores das variáveis no nível do pipeline são resolvidos no momento da execução do pipeline e podem ser visualizados no histórico de cada execução.

> **Note**
>
> As variáveis de saída das ações do pipeline podem ser visualizadas na guia Variáveis de saída, abaixo do histórico de cada execução de ação.

Visualizar o status de execução (console)

Você pode visualizar o status do pipeline em Status na página do histórico de execução. Escolha um link de ID de execução e visualize o status da ação.

Veja a seguir os estados válidos para pipelines, estágios e ações:

> **Note**
>
> Os estados de pipeline a seguir também se aplicam a uma execução de pipeline, que é uma execução de entrada. Para visualizar uma execução de entrada e seu status, consulte Visualizar uma execução de entrada (console).

Pipeline-level estados

Estado do pipeline	Description
InProgress	O pipeline está em execução no momento.
Parando	A execução do pipeline está sendo interrompida devido a uma solicitação para interromper e aguardar ou interromper e abandonar a execução do pipeline.
Interrompido	O processo de interrupção é concluído e a execução do pipeline é interrompida.
Bem-sucedido	A execução do pipeline foi concluída com êxito.
Substituído	Embora a conclusão da execução desse pipeline estivesse programada para o estágio seguinte, uma nova execução se antecipou e prosseguiu pelo pipeline.
Failed	A execução do pipeline não foi concluída com êxito.

Stage-level estados

Estado do estágio	Description
InProgress	O estágio está em execução no momento.
Parando	A execução do estágio é interrompida devido a uma solicitação para interromper e aguardar ou interromper e abandonar a execução do pipeline.

Estado do estágio	Description
Interrompido	O processo de interrupção é concluído e a execução do estágio é interrompida.
Bem-sucedido	O estágio foi concluído com êxito.
Failed	O estágio não foi concluído com êxito.

Action-level estados

Estado da ação	Description
InProgress	A ação está em execução no momento.
Abandonado	A ação é abandonada devido a uma solicitação para interromper e abandonar a execução do pipeline.
Bem-sucedido	A ação foi concluída com êxito.
Failed	Para ações de aprovação, o estado FAILED significa que a ação foi rejeitada pelo revisor ou falhou devido a uma ação de configuração incorreta.

Visualizar uma execução de entrada (console)

Você pode usar o console do para visualizar o status e os detalhes de uma execução de entrada. Quando a transição é habilitada ou o estágio fica disponível, uma execução de entrada `InProgress` continua e entra no estágio. Uma execução de entrada com o `Stopped` status não entra no estágio. O status de execução de entrada será alterado para `Failed` se o pipeline for editado. Quando você edita um pipeline, todas as execuções em andamento são interrompidas e o status da execução é alterado para `Failed`.

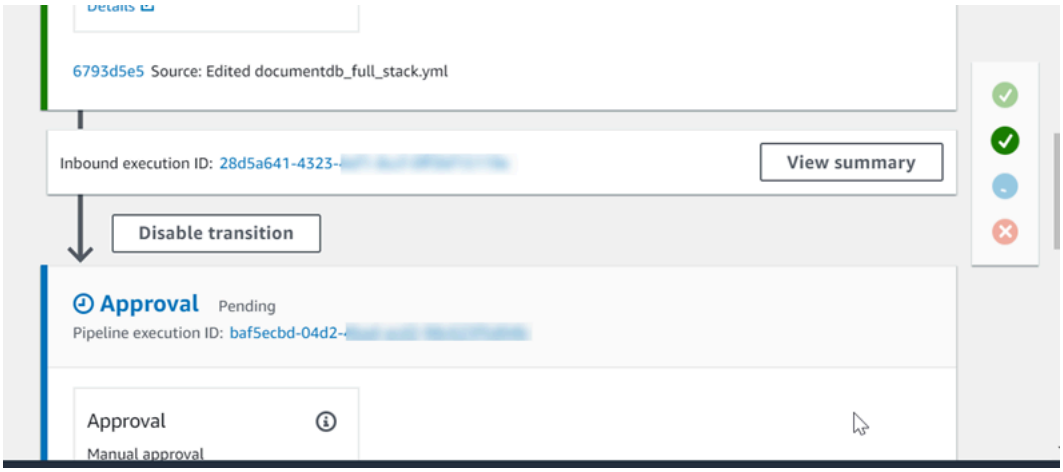
Se uma execução de entrada não é exibida, então não há execuções pendentes em uma transição de estágio desabilitada.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

Os nomes de todos os canais associados à sua AWS conta serão exibidos.

2. Escolha o nome do pipeline cuja execução de entrada você deseja visualizar. Siga um destes procedimentos:

   - Selecione Visualizar. No diagrama do pipeline, no campo ID de execução de entrada na frente da transição desabilitada, você verá o ID de execução de entrada.



     Escolha Visualizar resumo para ver os detalhes da execução, como o ID da execução, o gatilho de origem e o nome do próximo estágio.

   - Escolha o pipeline e, em seguida, escolha Visualizar histórico.

Visualizar revisões de origem de execução do pipeline (console)

Você pode visualizar os detalhes sobre os artefatos de origem (artefato de saída originado no primeiro estágio de um pipeline) que são usados na execução de um pipeline. Os detalhes incluem identificadores, como IDs de confirmação, comentários de dicas e, ao usar a CLI, números de versão das ações de criação de pipelines. No caso de alguns tipos de revisão, é possível visualizar e abrir o URL da confirmação. As revisões de origem são compostas pelo seguinte:

- Resumo: informações de resumo sobre a revisão mais recente do artefato. Para GitHub e CodeCommit repositórios, a mensagem de confirmação. Para buckets ou ações do Amazon S3, o conteúdo fornecido pelo usuário de uma chave codepipeline-artifact-revision-summary especificada nos metadados do objeto.
- revisionUrl: o URL da revisão do artefato (por exemplo, o URL do repositório externo).
- revisionId: o ID da revisão do artefato. Por exemplo, para uma alteração de origem em um GitHub repositório CodeCommit or, esse é o ID do commit. Para artefatos armazenados em nossos

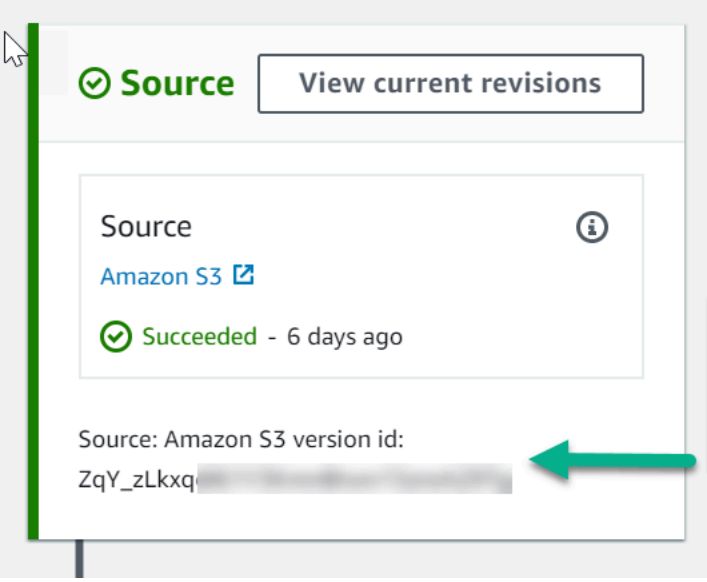
GitHub CodeCommit repositórios, o ID do commit está vinculado a uma página de detalhes do commit.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

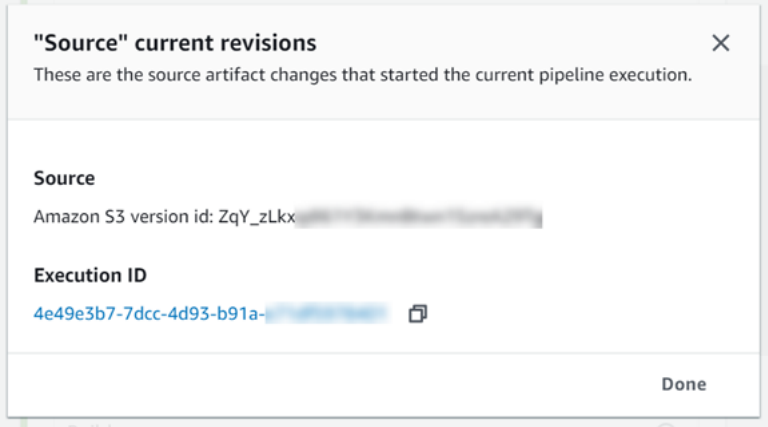
   Os nomes de todos os pipelines associados ao seu Conta da AWS serão exibidos.

2. Escolha o nome do pipeline para o qual você deseja visualizar os detalhes de revisão de origem. Execute um destes procedimentos:

   - Selecione Visualizar histórico. Em Source revisions (Revisões de origem), é listada a alteração de origem para cada execução.
   - Localize uma ação para a qual você deseja visualizar os detalhes de revisão de origem e depois encontre as informações de revisão na parte inferior de seu estágio:



   Escolha View current revisions (Visualizar revisões atuais) para visualizar informações da origem. Com exceção dos artefatos armazenados nos buckets do Amazon S3, identificadores como IDs de confirmação nesta visualização detalhada de informações são vinculados às páginas de informações de origem dos artefatos.



Visualizar execuções da ação (console)

Você pode visualizar os detalhes da ação para um pipeline, como o ID de execução da ação, os artefatos de entrada, os artefatos de saída e o status. Você pode visualizar os detalhes da ação escolhendo um pipeline no console e selecionando um ID de execução.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes de todos os canais associados à sua AWS conta são exibidos.
2. Escolha o nome do pipeline para o qual deseja visualizar os detalhes da ação e selecione View history (Exibir histórico).
3. Em Execution ID (ID de execução), escolha o ID de execução para a qual deseja visualizar os detalhes da execução da ação.
4. Você pode visualizar as seguintes informações na guia Timeline (Cronograma):

   a. Em Action name (Nome da ação), escolha o link para abrir uma página de detalhes da ação, onde é possível visualizar o status, o nome do estágio, o nome da ação, os dados de configuração e as informações do artefato.

   b. Em Provider (Provedor), escolha o link para visualizar os detalhes do provedor da ação. Por exemplo, no exemplo anterior de pipeline, se você escolher entre CodeDeploy os estágios de preparação ou produção, a página do CodeDeploy console do CodeDeploy aplicativo configurado para esse estágio será exibida.

Visualizar informações sobre artefatos e armazenamento de artefatos da ação (console)

Você pode visualizar detalhes de artefatos de entrada e saída para uma ação. Você também pode escolher um link que direciona você para as informações de artefato dessa ação. Como o armazenamento de artefatos usa versionamento, cada execução da ação tem um local de artefato de entrada e saída exclusivo.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes de todos os canais associados à sua AWS conta são exibidos.
2. Escolha o nome do pipeline para o qual deseja visualizar os detalhes da ação e selecione View history (Exibir histórico).
3. Em Execution ID (ID de execução), escolha o ID de execução para o qual deseja visualizar os detalhes da ação.
4. Na guia Timeline (Cronograma), em Action name (Nome da ação), escolha o link para abrir uma página de detalhes da ação.
5. Na página de detalhes, na guia Execução, visualize o status e a duração da ação.
6. Na guia Configuração, visualize a configuração do recurso para a ação (por exemplo, o nome do projeto de CodeBuild construção).
7. Em Artefatos, visualize os detalhes do artefato em Tipo de artefato e Provedor de artefatos. Escolha o link em Artifact name (Nome do artefato) para visualizar os artefatos no armazenamento de artefatos.
8. Na guia Variáveis de saída, visualize as variáveis resolvidas das ações no pipeline para a execução da ação.

Visualizar detalhes e histórico do pipeline (CLI)

Você pode executar os seguintes comandos para visualizar detalhes sobre seus pipelines e execuções de pipelines:

- list-pipelinescomando para ver um resumo de todos os pipelines associados ao seu Conta da AWS.
- Comando get-pipeline para revisar os detalhes de um único pipeline.

- list-pipeline-executions para visualizar resumos das execuções mais recentes de um pipeline.
- get-pipeline-execution para visualizar informações sobre a execução de um pipeline, incluindo os detalhes sobre os artefatos, o ID de execução do pipeline e o nome, a versão e o status do pipeline.
- Comando get-pipeline-state para visualizar o pipeline, o estágio e o status da ação.
- list-action-executions para visualizar detalhes de execução da ação para um pipeline.

Tópicos

- Visualize o histórico de execução com list-pipeline-executions (CLI)
- Visualize o estado do pipeline com get-pipeline-state (CLI)
- Visualize o status de execução de entrada com get-pipeline-state (CLI)
- Visualize o status e as revisões da fonte com get-pipeline-execution (CLI)
- Visualize execuções de ações com list-action-executions (CLI)

Visualize o histórico de execução com list-pipeline-executions (CLI)

Você pode visualizar o histórico de execução do pipeline.

- Para visualizar detalhes sobre execuções anteriores de um pipeline, execute o comando list-pipeline-executions, especificando o nome exclusivo do pipeline. Por exemplo, para visualizar detalhes do estado atual de um pipeline designado *MyFirstPipeline*, insira o seguinte:

  ```
  aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline
  ```

  Esse comando retorna informações de resumo sobre todas as execuções de pipeline para as quais um histórico foi registrado. O resumo inclui períodos de início e de término, duração e status.

  Execuções de pipeline que foram revertidas mostrarão o tipo de execução `Rollback`. Para a execução com falha que acionou a reversão automática, o ID da execução com falha é exibido.

  O exemplo a seguir mostra os dados retornados para um pipeline designado *MyFirstPipeline* que teve três execuções:

  ```
  {
      "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 visualizar detalhes adicionais sobre a execução de um pipeline, execute get-pipeline-execution, especificando o ID exclusivo da execução do pipeline. Por exemplo, para visualizar mais detalhes sobre a primeira execução do exemplo anterior, insira o seguinte:

```
aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
execution-id 7cf7f7cb-3137-539g-j458-d7eu3EXAMPLE
```

Este comando retorna informações resumidas sobre a execução de um pipeline, incluindo detalhes sobre artefatos, ID de execução do pipeline e o nome, versão e status do pipeline.

O exemplo a seguir mostra os dados retornados para um pipeline designado *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"
            }
        ]
    }
}
```

Visualize o estado do pipeline com get-pipeline-state (CLI)

Você pode usar a CLI para visualizar o pipeline, o estágio e o status da ação.

- Para visualizar detalhes sobre o estado atual de um pipeline, execute o comando get-pipeline-state, especificando o nome exclusivo do pipeline. Por exemplo, para visualizar detalhes do estado atual de um pipeline designado *MyFirstPipeline*, insira o seguinte:

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

  Este comando retorna o status atual de todos os estágios do pipeline e o status das ações dentro desses estágios.

O exemplo a seguir mostra os dados retornados para um pipeline de três estágios designado *MyFirstPipeline*, onde os dois primeiros estágios e ações tiveram êxito, o terceiro estágio falhou e a transição entre o segundo e o terceiro estágios está desabilitada:

```
{
    "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"
        }
    ]
}
```

Visualize o status de execução de entrada com get-pipeline-state (CLI)

Você pode usar a CLI para visualizar o status de execução de entrada. Quando a transição é habilitada ou o estágio fica disponível, uma execução de entrada `InProgress` continua e entra no estágio. Uma execução de entrada com o `Stopped` status não entra no estágio. O status de execução de entrada será alterado para `Failed` se o pipeline for editado. Quando você edita um pipeline, todas as execuções em andamento são interrompidas e o status da execução é alterado para `Failed`.

- Para visualizar detalhes sobre o estado atual de um pipeline, execute o comando get-pipeline-state, especificando o nome exclusivo do pipeline. Por exemplo, para visualizar detalhes do estado atual de um pipeline designado *MyFirstPipeline*, insira o seguinte:

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

Este comando retorna o status atual de todos os estágios do pipeline e o status das ações dentro desses estágios. A saída também mostra o ID de execução do pipeline em cada estágio e se há um ID de execução de entrada para um estágio com transição desabilitada.

O exemplo a seguir mostra os dados retornados de um pipeline de dois estágios chamado*MyFirstPipeline*, em que o primeiro estágio mostra uma transição habilitada e uma execução bem-sucedida do pipeline, e o segundo estágio, chamado`Beta`, mostra uma transição desativada e um ID de execução de entrada. A execução de entrada pode ter o status `InProgress`, `Stopped` ou `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
}
```

Visualize o status e as revisões da fonte com get-pipeline-execution (CLI)

Você pode visualizar os detalhes sobre os artefatos de origem (artefatos de saída originados no primeiro estágio de um pipeline) que são usados na execução de um pipeline. Os detalhes incluem identificadores, como IDs de confirmação, comentários de dicas, hora desde que o artefato foi criado ou atualizado e quando você usa a CLI, bem como os números de versão e as ações de criação. No caso de alguns tipos de revisão, é possível visualizar e abrir o URL da confirmação da versão do artefato. As revisões de origem são compostas pelo seguinte:

- Resumo: informações de resumo sobre a revisão mais recente do artefato. Para GitHub e AWS CodeCommit repositórios, a mensagem de confirmação. Para buckets ou ações do Amazon S3, o conteúdo fornecido pelo usuário de uma chave codepipeline-artifact-revision-summary especificada nos metadados do objeto.
- revisionUrl: o ID de confirmação da revisão do artefato. Para artefatos armazenados em nossos GitHub AWS CodeCommit repositórios, o ID do commit está vinculado a uma página de detalhes do commit.

Você pode executar o console get-pipeline-execution para visualizar informações sobre as revisões de origem mais recentes que foram incluídas na execução de um pipeline. Após executar o comando get-pipeline-state pela primeira vez a fim de obter detalhes sobre todos os estágios em um pipeline, identifique o ID de execução que se aplica a um estágio sobre o qual deseja obter detalhes da revisão de origem. Depois, use o ID de execução no comando get-pipeline-execution. (Como os estágios em um pipeline podem ter sido concluídos com êxito durante execuções de pipeline diferentes, eles podem ter IDs de execução diferentes.)

Ou seja, para vusialuzar detalhes sobre os artefatos atualmente no estágio Preparação, execute o comando get-pipeline-state, identifique o ID de execução atual do estágio Preparação e em seguida, execute o comando get-pipeline-execution usando esse ID de execução.

Como visualizar o status e as revisões de origem em um pipeline

1. Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows) e use a AWS CLI para executar o comando get-pipeline-state. Para um pipeline chamado *MyFirstPipeline*, digite:

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

Este comando retorna o estado mais recente de um pipeline, incluindo a ID de execução do pipeline mais recente para cada estágio.

2. Para visualizar detalhes sobre a execução de um pipeline, execute o comando get-pipeline-execution, especificando o nome exclusivo do pipeline e o ID da execução do pipeline para a qual deseja visualizar os detalhes do artefato. Por exemplo, para visualizar detalhes sobre a execução de um pipeline chamado *MyFirstPipeline*, com o ID de execução 3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE, digite o seguinte:

```
aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
execution-id 3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE
```

Este comando retorna informações sobre cada revisão de origem que faz parte da execução do pipeline e identifica informações sobre o pipeline. Somente informações sobre estágios de pipeline que foram incluídos na execução estão incluídos. Pode haver outros estágios no pipeline que não foram parte dessa execução do pipeline.

O exemplo a seguir mostra os dados retornados para uma parte do pipeline chamada*MyFirstPipeline*, na qual um artefato chamado "MyApp" é armazenado em um GitHub repositório:

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"
        }
    }
}
```

Visualize execuções de ações com list-action-executions (CLI)

Você pode visualizar os detalhes de execução da ação para um pipeline, como o ID de execução da ação, os artefatos de entrada, os artefatos de saída, o resultado da execução e o status. Forneça o filtro do ID de execução para retornar uma lista de ações em uma execução de pipeline:

> **Note**
>
> O histórico de execução detalhado está disponível para execuções ocorridas em 21 de fevereiro de 2019 ou posteriormente.

- Para visualizar as execuções da ação de um pipeline, execute uma das seguintes ações:
  - Para visualizar detalhes sobre todas as execuções de ação de um pipeline, execute o comando list-action-executions, especificando o nome exclusivo do pipeline. Por exemplo, para visualizar as execuções de ação em um pipeline chamado *MyFirstPipeline*, insira o seguinte:

    ```
    aws codepipeline list-action-executions --pipeline-name MyFirstPipeline
    ```

    Veja a seguir uma parte do exemplo de saída para esse 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 visualizar todas as execuções de ação na execução de um pipeline, execute o comando list-action-executions, especificando o nome exclusivo do pipeline e o ID de execução. Por exemplo, para visualizar as execuções de ações de um*Execution_ID*, digite o seguinte:

```
aws codepipeline list-action-executions --pipeline-name MyFirstPipeline --filter
 pipelineExecutionId=Execution_ID
```

- Veja a seguir uma parte do exemplo de saída para esse 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"
                    }
                ]
            },


. . .
```

Definir ou alterar o modo de execução do pipeline

Você pode definir o modo de execução do pipeline para especificar como múltiplas execuções são tratadas.

Para ter mais informações sobre os modos de execução de pipeline, consulte Como funcionam as execuções de pipeline.

> **Important**
>
> Em pipelines no modo PARALLEL, ao editar o modo de execução do pipeline para QUEUED ou SUPERSEDED, o estado do pipeline não exibirá o estado atualizado como PARALLEL. Para obter mais informações, consulte Os pipelines alterados do modo PARALLEL exibirão um modo de execução anterior.

> **Important**
>
> Em pipelines no modo PARALLEL, ao editar o modo de execução do pipeline para QUEUED ou SUPERSEDED, a definição do pipeline em cada modo não será atualizada. Para obter mais informações, consulte Ao alterar para os modos QUEUED ou SUPERSEDED, pipelines configurados no modo PARALLEL podem apresentar uma definição desatualizada..

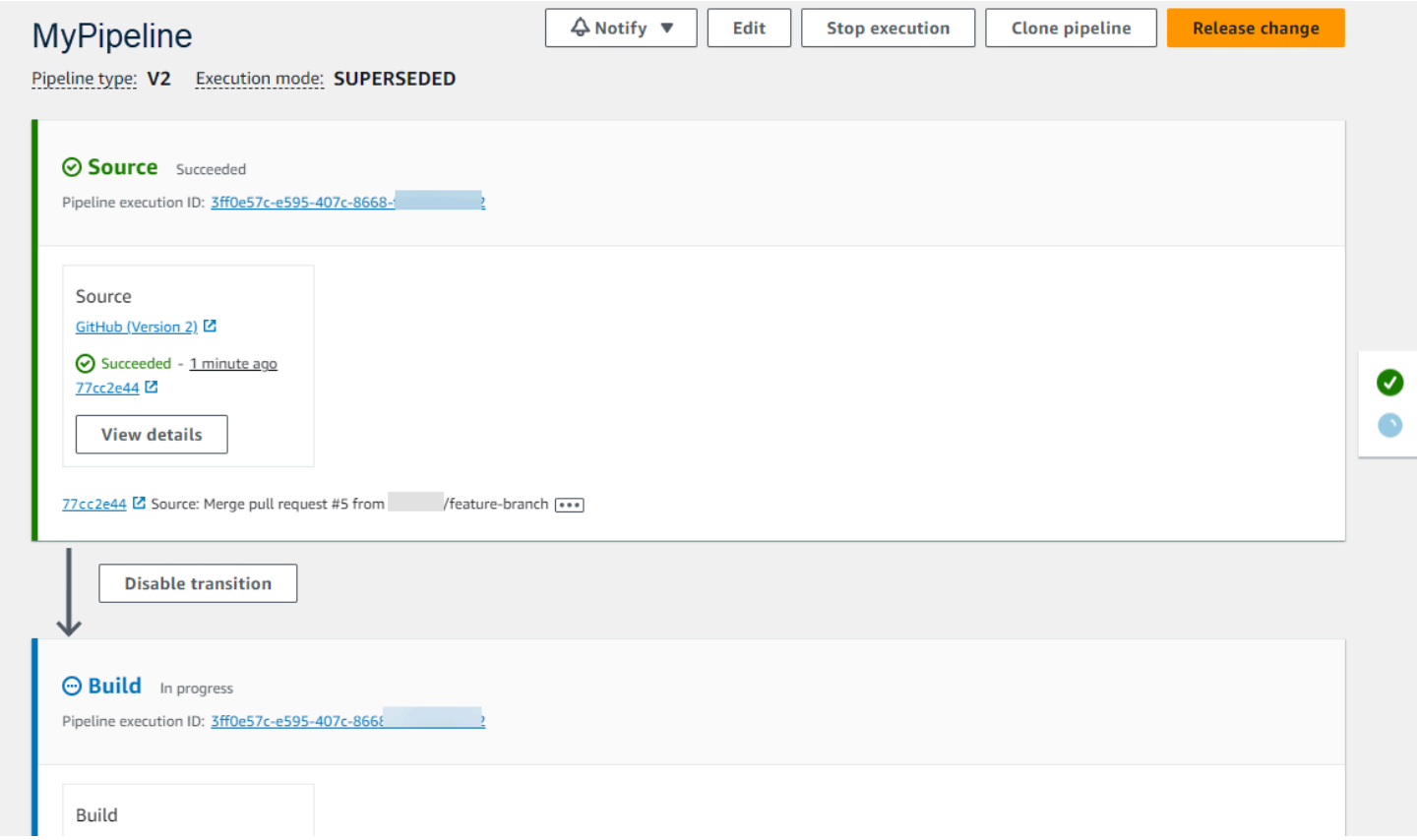
> **Important**
>
> Para pipelines em modo PARALLEL, a reversão do estágio não está disponível. Da mesma forma, condições de falha com um tipo de resultado de reversão não podem ser adicionadas a um pipeline de modo PARALLEL.

Considerações para visualizar modos de execução

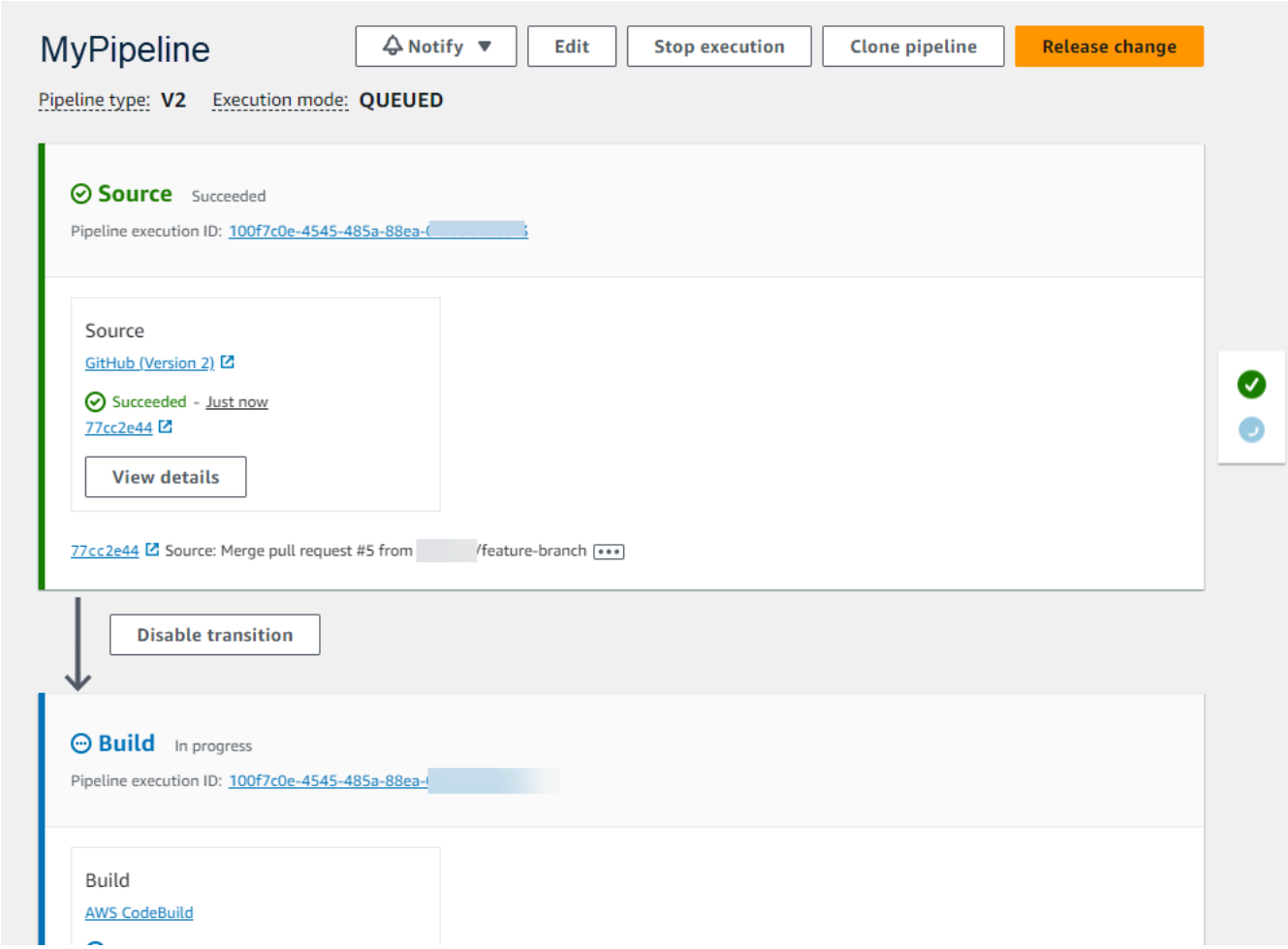
Há considerações para visualizar pipelines em modos de execução específicos.

Para os modos SUPERSEDED e QUEUED, use a visualização do pipeline para ver as execuções em andamento e clique no ID da execução para visualizar detalhes e histórico. No modo PARALLEL, clique no ID de execução para visualizar a execução em andamento na guia Visualização.

O seguinte mostra a exibição do modo SUPERSEDED em. CodePipeline

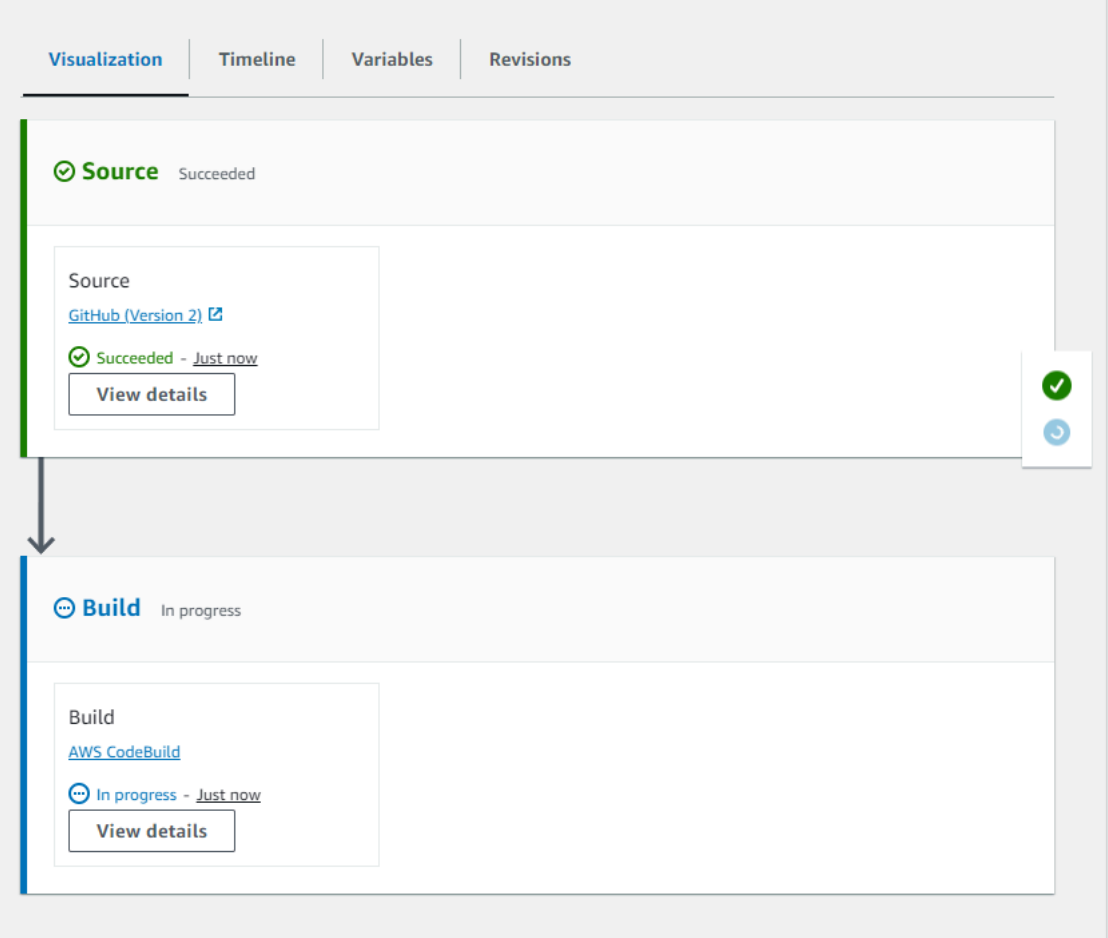


O seguinte mostra a exibição do modo FILA em. CodePipeline



O seguinte mostra a exibição do modo PARALELO em CodePipeline.

> **Important**
>
> Para pipelines em modo PARALLEL, a reversão do estágio não está disponível. Da mesma forma, as condições de falha com um tipo de resultado de reversão não podem ser adicionadas a um pipeline de modo PARALLEL.



Considerações para alternar entre os modos de execução

A seguir estão algumas considerações sobre pipelines ao alterar o modo do pipeline. Ao alternar de um modo de execução para outro no modo Editar e salvar a alteração, certas visualizações ou estados poderão se ajustar.

Por exemplo, ao alternar do modo PARALLEL para o modo QUEUED ou SUPERSEDED, a execução que foi iniciada no modo PARALLEL continuará em andamento. Essas informações podem ser consultadas na página de histórico de execução. A visualização do pipeline exibirá a execução que ocorreu no modo QUEUED ou SUPERSEDED anteriormente, ou um estado vazio, se não houver nenhuma execução.

Como outro exemplo, ao alternar do modo QUEUED ou SUPERSEDED para o PARALLEL, você não verá mais a página do pipeline. view/state Para ver uma execução no modo PARALLEL, utilize a guia de visualização na página de detalhes da execução. As execuções iniciadas no modo SUPERSEDED ou QUEUED serão canceladas.

A tabela a seguir apresenta mais detalhes.

Alteração de modo	Detalhes das execuções pendente e ativa	Detalhes do estado do pipeline
SUPERSEDED para SUPERSEDED ou SUPERSEDED para QUEUED	• As execuções ativas são canceladas após a conclusão das ações em andamento. • As execuções pendentes são canceladas.	O estado do pipeline, como cancelado, é preservado entre a versão do primeiro modo e o segundo modo.
QUEUED para QUEUED ou QUEUED para SUPERSEDED	• As execuções ativas são canceladas após a conclusão das ações em andamento. • As execuções pendentes são canceladas.	O estado do pipeline, como cancelado, é preservado entre a versão do primeiro modo e o segundo modo.
PARALLEL para PARALLEL	Todas as execuções podem ocorrer sem depender das atualizações na definição do pipeline.	Vazio. O modo paralelo não tem um estado de pipeline.
SUPERSEDED para PARALLEL ou QUEUED para PARALLEL	• As execuções ativas são canceladas após a conclusão das ações em andamento. • As execuções pendentes são canceladas.	Vazio. O modo paralelo não tem um estado de pipeline.

Definir ou alterar o modo de execução do pipeline (console)

É possível usar o console para configurar o modo de execução do pipeline.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

Os nomes e o status de todos os canais associados à sua AWS conta são exibidos.

2. Em Nome, selecione o nome do pipeline que você deseja editar.
3. Na página de detalhes do pipeline, selecione Editar.
4. Na página Editar, escolha Editar: propriedades do pipeline.
5. Escolha o modo para o pipeline.

   - Substituído
   - Em fila (tipo de pipeline V2 necessário)
   - Paralelo (tipo de pipeline V2 necessário)
6. Na página Editar, escolha Concluído.

Definir o modo de execução do pipeline (CLI)

Para usar o AWS CLI para definir o modo de execução do pipeline, use o `update-pipeline` comando `create-pipeline` ou.

1. Abra uma sessão de terminal (Linux, macOS, or Unix) ou prompt de comando (Windows) e execute o comando get-pipeline para copiar a estrutura do pipeline em um arquivo JSON. Por exemplo, para um pipeline nomeado **`MyFirstPipeline`**, insira o seguinte comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando retorna nada, mas o arquivo que você criou deve aparecer no diretório onde você executou o comando.
2. Abra o arquivo JSON em qualquer editor de texto simples e modifique a estrutura do arquivo para refletir o modo de execução do pipeline que você deseja definir, como QUEUED.

   ```
   "executionMode": "QUEUED"
   ```

   O exemplo a seguir mostra como você definiria o modo de execução como QUEUED em um pipeline de exemplo com dois estágios.

   ```
   {
       "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. Se você estiver trabalhando com a estrutura do pipeline recuperada usando o comando get-pipeline, você deve modificar a estrutura no arquivo JSON. Você deve remover as linhas `metadata` do arquivo para que o comando update-pipeline possa usá-lo. Remova a seção da estrutura do pipeline no arquivo JSON (as linhas de `"metadata": { }` e os campos `"created"`, `"pipelineARN"` e `"updated"`).

   Por exemplo, remova as seguintes linhas da estrutura:

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

Salve o arquivo.

4. Para aplicar suas alterações, execute o comando update-pipeline especificando o arquivo JSON do pipeline:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

Este comando retorna toda a estrutura do pipeline editado.

> **Note**
>
> O comando update-pipeline interrompe o pipeline. Se uma revisão estiver sendo executada pelo pipeline quando você executar o comando update-pipeline, essa execução será interrompida. Você deve iniciar manualmente o pipeline para executar a revisão através do pipeline atualizado.

Reverter ou repetir os estágios

Você pode usar o console do AWS CodePipeline ou a AWS CLI para reverter manualmente ou repetir um estágio ou ações em um estágio. Para configurar condições para os estágios em que você deseja usar a função Reverter ou Repetir como resultado configurado, consulte Configurar condições para um estágio.

Tópicos

- Configurar a repetição do estágio para um estágio com falha ou ações com falha
- Configuração da reversão de estágio

Configurar a repetição do estágio para um estágio com falha ou ações com falha

Você pode tentar novamente um estágio com falha sem precisar executar novamente um pipeline desde o início. Você faz isso repetindo as ações que falharam em um estágio ou repetindo todas as ações do estágio desde a primeira ação. Quando você repete as ações com falha em um estágio, todas as ações que ainda estão em andamento continuam funcionando, e as ações com falha são acionadas novamente. Quando você repete um estágio com falha desde a primeira ação do estágio, o estágio não pode ter nenhuma ação em andamento. Para que um estágio possa ser repetido, é necessário que todas as ações tenham falhado ou que algumas ações tenham falhado e algumas tenham sido bem-sucedidas.

> **Important**
>
> A repetição de um estágio com falha repete todas as ações do estágio desde a primeira ação, e a repetição das ações com falha repete todas as ações com falha do estágio. Isso substitui os artefatos de saída de ações anteriormente bem-sucedidas na mesma execução. Embora os artefatos possam ser substituídos, o histórico de execução de ações anteriormente bem-sucedidas ainda é mantido.

Se você estiver usando o console para visualizar um pipeline, o botão Tentar a etapa novamente ou Repetir ações com falha será exibido no estágio que pode ser repetido.

Se você estiver usando a AWS CLI, poderá usar o get-pipeline-state comando para determinar se alguma ação falhou.

> **Note**
>
> Nos casos a seguir, talvez não seja possível repetir um estágio:
>
> - Todas as ações no estágio foram bem-sucedidas. Portanto, o estágio não está em um status de falha.
> - A estrutura geral do pipeline foi alterada após a falha do estágio.
> - Outra tentativa no estágio já está em andamento.

Tópicos

- Considerações para repetição de estágio
- Repetir manualmente um estágio com falha
- Configurar a repetição automática em caso de falha para um estágio

Considerações para repetição de estágio

As considerações para a repetição de estágio são as seguintes:

- Você só pode configurar a repetição automática em caso de falha de estágio para uma tentativa de repetição.
- Você pode configurar a repetição automática em caso de falha de estágio para todas as ações, incluindo ações `Source`.

Repetir manualmente um estágio com falha

Você pode repetir manualmente um estágio com falha usando o console ou a CLI.

Você também pode configurar um estágio para repetir automaticamente em caso de falha, conforme detalhado em Configurar a repetição automática em caso de falha para um estágio.

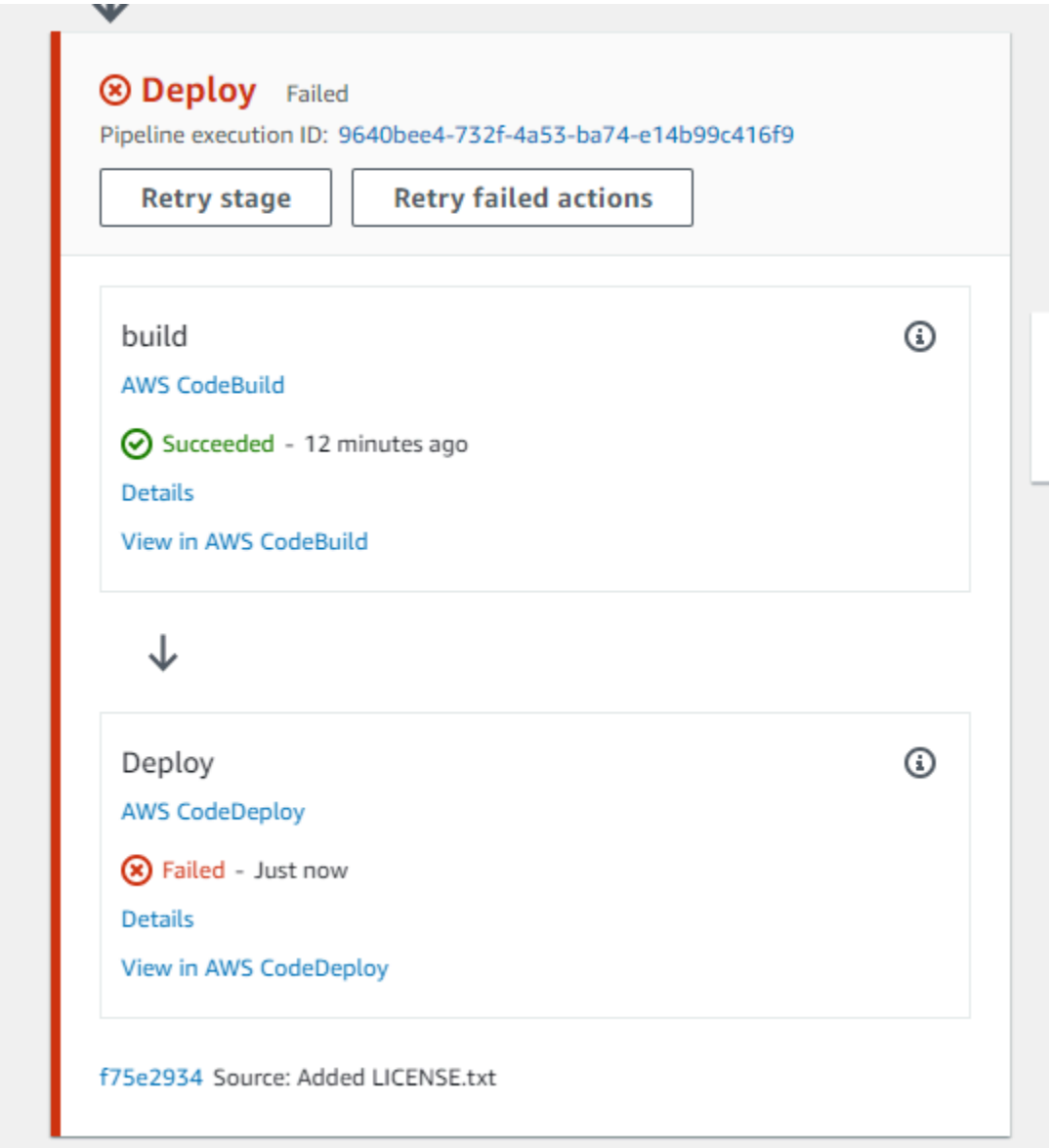
Repetir manualmente um estágio com falha (console)

Para repetir um estágio com falha ou ações com falha em um estágio - console

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes de todos os canais associados à sua AWS conta são exibidos.
2. Em Nome, selecione o nome do pipeline.
3. Localize o estágio com a ação com falha e escolha uma das seguintes alternativas:

   - Para repetir todas as ações no estágio, escolha Tentar a etapa novamente.
   - Para repetir somente as ações com falha no estágio, escolha Repetir ações com falha.



Se todas as ações repetidas no estágio são concluídas com êxito, o pipeline continuará a ser executado.

Repetir manualmente um estágio com falha (CLI)

Para repetir um estágio com falha ou ações com falha em um estágio (CLI)

Para usar o AWS CLI para repetir todas as ações ou todas as ações que falharam, execute o retry-stage-execution comando com os seguintes parâmetros:

```
--pipeline-name <value>
--stage-name <value>
--pipeline-execution-id <value>
--retry-mode ALL_ACTIONS/FAILED_ACTIONS
```

> **Note**
>
> Os valores que você pode usar para `retry-mode` são `FAILED_ACTIONS` e `ALL_ACTIONS`.

1. Em um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows), execute o comando retry-stage-execution, conforme mostrado no exemplo a seguir para um pipeline chamado `MyPipeline`.

   ```
   aws codepipeline retry-stage-execution --pipeline-name MyPipeline --stage-name
    Deploy --pipeline-execution-id b59babff-5f34-EXAMPLE --retry-mode FAILED_ACTIONS
   ```

   A saída retorna o ID de execução:

   ```
   {
       "pipelineExecutionId": "b59babff-5f34-EXAMPLE"
   }
   ```

2. Também é possível executar o comando com um arquivo de entrada JSON. Primeiro você deve criar um arquivo JSON que identifique o pipeline, o estágio que contém as ações com falha e a última execução do pipeline naquele estágio. Execute o comando retry-stage-execution com o parâmetro `--cli-input-json`. Para recuperar os detalhes necessários para o arquivo JSON, é mais fácil usar o comando get-pipeline-state.

   a. Em um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows), execute o comando get-pipeline-state em um pipeline. Por exemplo, para um pipeline chamado MyFirstPipeline, você digitaria algo semelhante ao seguinte:

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

      A resposta ao comando inclui informações do estado do pipeline para cada etapa. No exemplo a seguir, a resposta indica que uma ou mais ações falharam no estágio de 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. Em um editor de texto plano, crie um arquivo para registrar o seguinte, no formato JSON:

- O nome do pipeline que contém as ações com falha
- O nome do estágio que contém as ações com falha
- A ID da execução mais recente do pipeline no estágio
- O modo de repetição.

No MyFirstPipeline exemplo anterior, seu arquivo teria a seguinte aparência:

```
{
    "pipelineName": "MyFirstPipeline",
    "stageName": "Staging",
    "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
    "retryMode": "FAILED_ACTIONS"
}
```

c. Salve o arquivo com um nome como **`retry-failed-actions.json`**.

d. Invoque o arquivo que você criou quando executou o comando retry-stage-execution. Por exemplo:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

```
aws codepipeline retry-stage-execution --cli-input-json file://retry-failed-actions.json
```

e. Para ver os resultados da nova tentativa, abra o CodePipeline console e escolha o pipeline que contém as ações que falharam ou use o get-pipeline-state comando novamente. Para obter mais informações, consulte Veja os pipelines e os detalhes em CodePipeline.

Configurar a repetição automática em caso de falha para um estágio

Você pode configurar a repetição automática em caso de falha para um estágio. O estágio repetirá e exibirá um status de repetição no estágio com falha na página Exibir do pipeline.

Você pode configurar o modo de repetição especificando que o estágio deve repetir automaticamente todas as ações no estágio com falha ou apenas as ações com falha no estágio.

Configurar a repetição automática em caso de falha para um estágio (console)

Você pode usar o console para configurar a repetição automática para um estágio.

Configurar a repetição automática para um estágio (console)

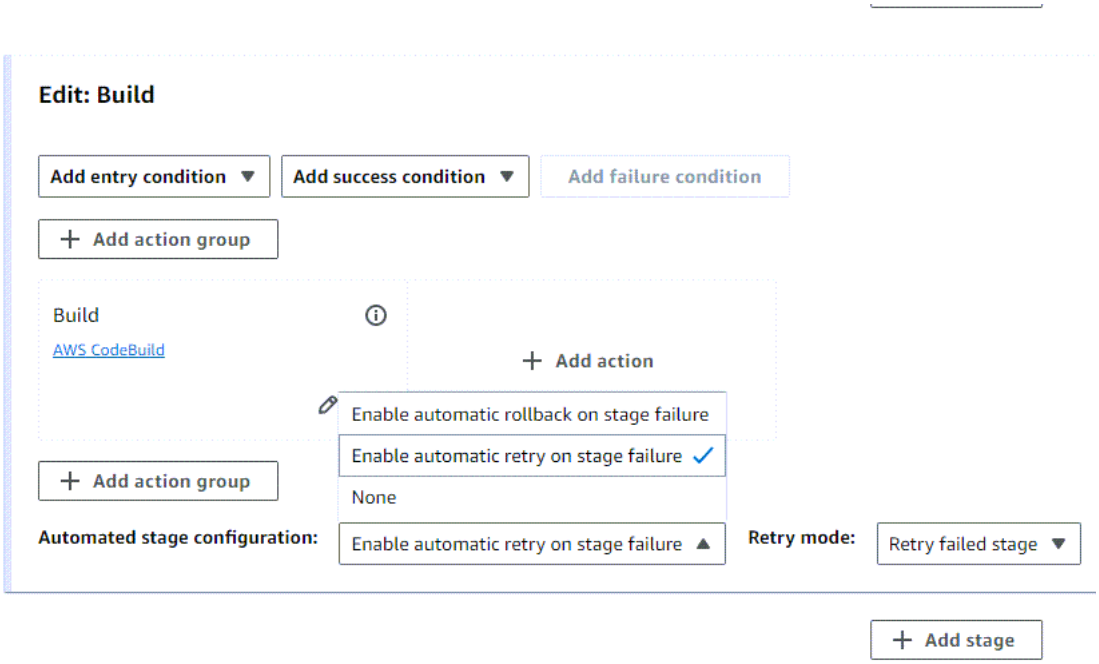
1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes e o status de todos os canais associados à sua AWS conta são exibidos.
2. Em Nome, selecione o nome do pipeline que você deseja editar.
3. Na página de detalhes do pipeline, selecione Editar.
4. Na página Editar, para a ação que você deseja editar, escolha Editar estágio.

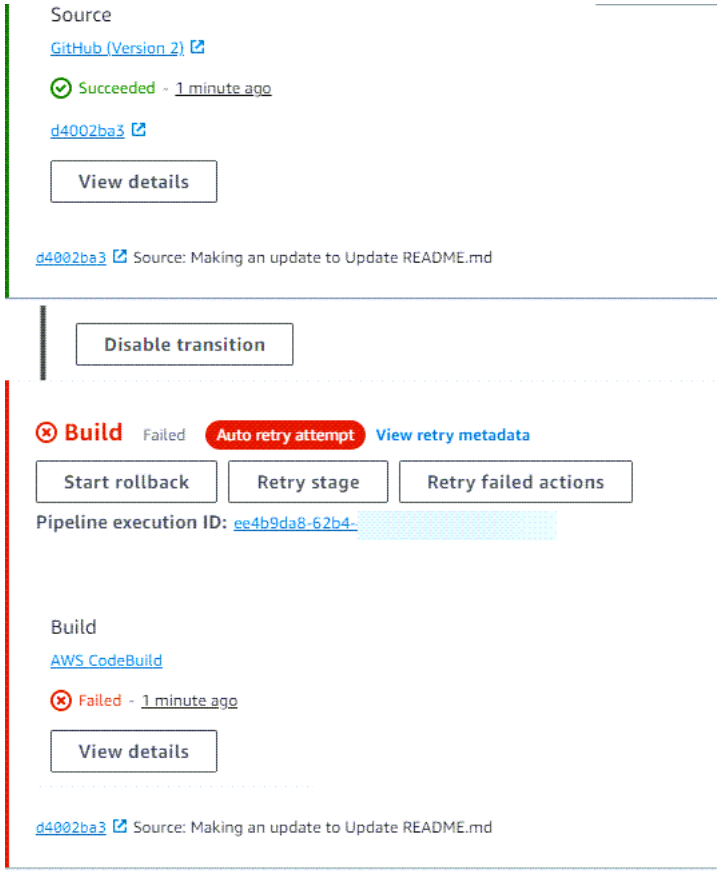
5. Selecione Configuração de etapa automatizada:, depois escolha Habilitar nova tentativa automática em caso de falha na etapa. Salve as alterações no pipeline.
6. Em Configuração de etapa automatizada:, selecione um dos seguintes modos de repetição:

   - Para especificar que o modo repetirá todas as ações no estágio, selecione Repetir estágio com falha.
   - Para especificar o modo de repetição voltado apenas às ações com falha no estágio, selecione Repetir ações com falha.

   Salve as alterações no pipeline.



7. Após a execução do pipeline, se ocorrer uma falha de estágio, a repetição automática será feita. Os exemplos a seguir mostram um estágio de compilação repetido automaticamente.



8. Para ver detalhes sobre a nova repetição, selecione Visualizar detalhes. A janela é exibida.



Configurar a repetição automática para um estágio (CLI)

Para usar o AWS CLI para configurar um estágio para tentar novamente automaticamente em caso de falha, use os comandos para criar ou atualizar um pipeline conforme detalhado em Criar um pipeline, estágios e ações e. Editar um pipeline no CodePipeline

- Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use o AWS CLI para executar o comando `update-pipeline`, especificando a condição de falha na estrutura

do pipeline. O exemplo a seguir configura a repetição automática para um estágio chamado `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",
                    },
            }
```

Configurar um estágio para nova tentativa automática (CloudFormation)

Para configurar um estágio CloudFormation para uma nova tentativa automática em caso de falha, use o parâmetro de ciclo de vida do `OnFailure` estágio. Use o parâmetro `RetryConfiguration` para configurar o modo de repetição.

```
OnFailure:
     Result: RETRY
     RetryConfiguration:
         RetryMode: ALL_ACTIONS
```

- Atualize o modelo conforme mostrado no snippet a seguir. O exemplo a seguir configura a repetição automática para um estágio chamado `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 obter mais informações sobre como configurar a nova tentativa em caso de falha, consulte OnFailure abaixo `StageDeclaration` no Guia do CloudFormation usuário.

Configuração da reversão de estágio

Você pode reverter um estágio para uma execução que foi bem-sucedida naquele estágio. Você pode pré-configurar um estágio para reversão em caso de falha ou pode reverter um estágio manualmente. A operação revertida resultará em uma nova execução. A execução do pipeline de destino escolhida para reversão é usada para recuperar revisões e variáveis de origem.

O tipo de execução, padrão ou reversão, é exibido no histórico do pipeline, no estado do pipeline e nos detalhes da execução do pipeline.

Tópicos

- Considerações para reversões
- Reverter um estágio manualmente
- Configurar um estágio para reversão automática
- Exibir o status de reversão na lista de execução
- Exibir detalhes do status da reversão

Considerações para reversões

As considerações para a reversão de estágio são as seguintes:

- Você não pode reverter um estágio de origem.
- O pipeline só pode realizar a reversão para uma execução anterior caso ela tenha sido iniciada na versão vigente da estrutura do pipeline.
- Não é possível reverter para um ID de execução de destino classificado como do tipo de execução de reversão.
- CodePipeline usará as variáveis e os artefatos da execução para a qual está sendo revertida.

Reverter um estágio manualmente

É possível realizar a reversão manual de um estágio por meio do console ou da CLI. O pipeline só pode realizar a reversão para uma execução anterior caso ela tenha sido iniciada na versão vigente da estrutura do pipeline.

Também é possível configurar um estágio para realizar a reversão automática em situações de falha, conforme especificado em Configurar um estágio para reversão automática.

Reverter um estágio manualmente (console)

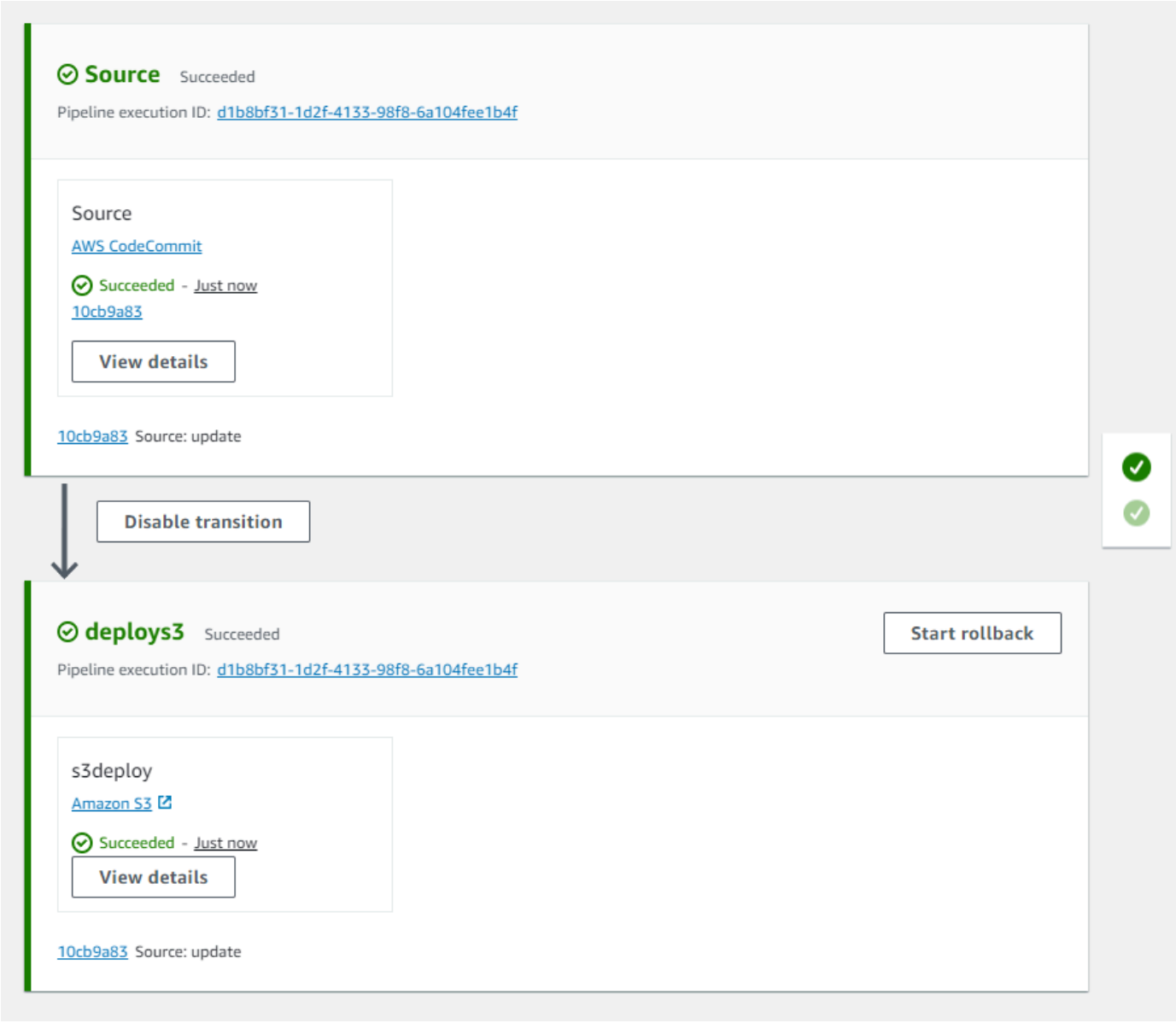
É possível utilizar o console para realizar manualmente a reversão de um estágio para uma execução de pipeline específica. Quando um estágio é revertido, um rótulo Reversão é exibido na visualização do pipeline no console.

Reverter um estágio manualmente (console)

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

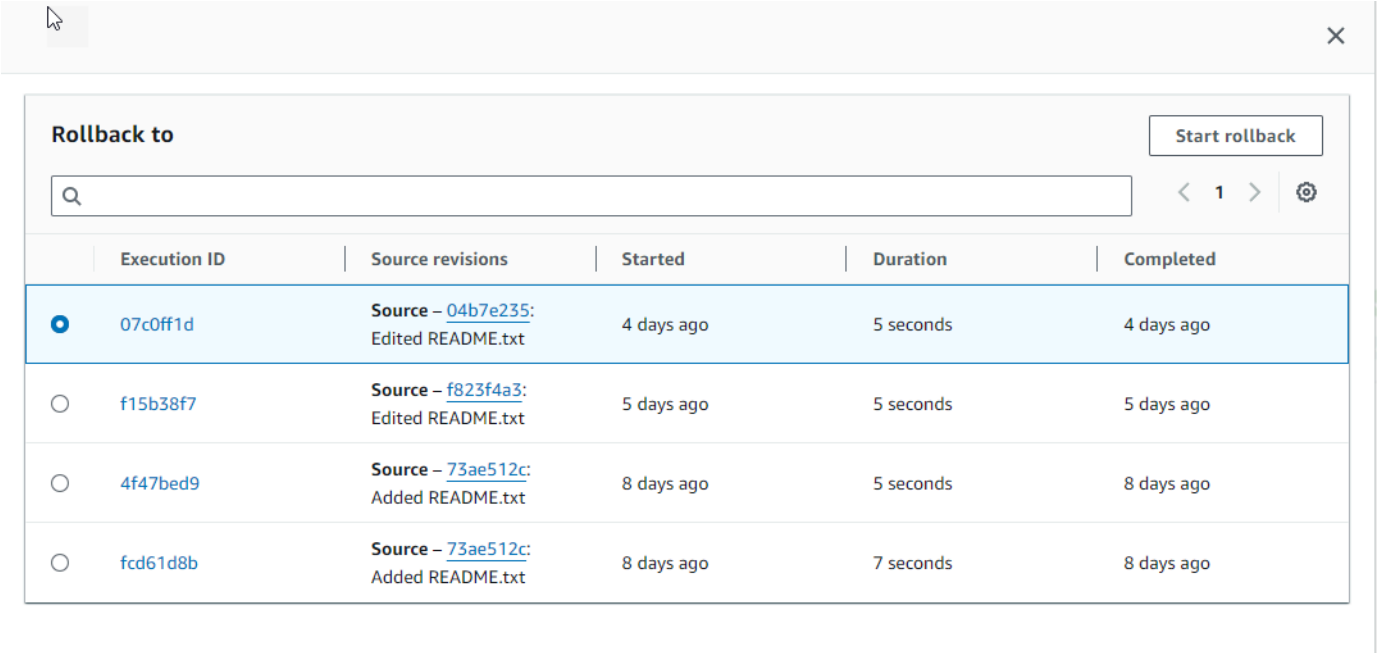
   Os nomes e o status de todos os canais associados à sua AWS conta são exibidos.

2. Em Nome, selecione o pipeline que contém o estágio a ser revertido.

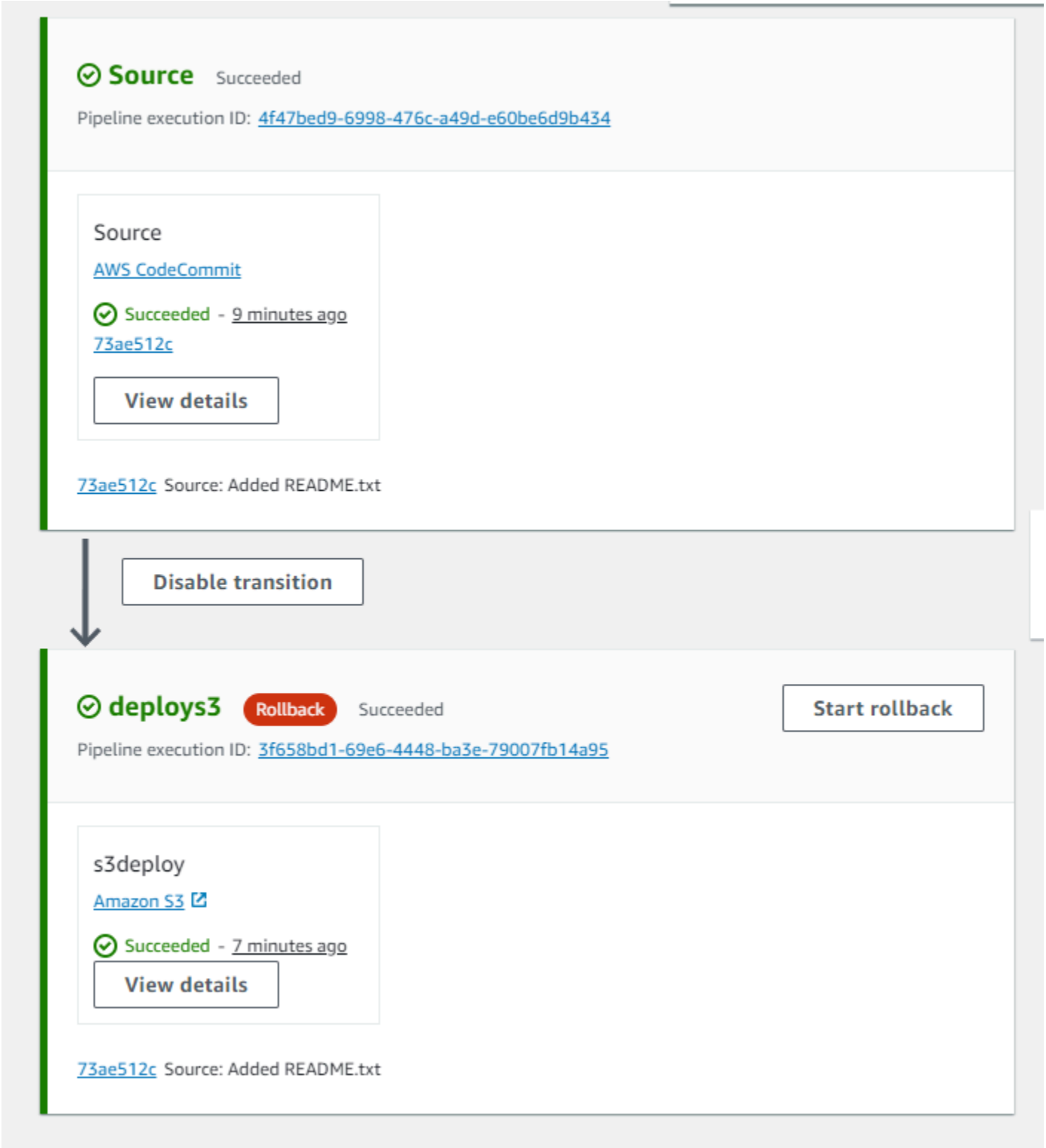


3. No estágio, selecione Iniciar reversão. A página Reverter para é exibida.
4. Selecione a execução de destino para onde deseja reverter o estágio.

> **Note**
>
> A lista de execuções de pipeline disponíveis como destino incluirá todas as execuções iniciadas na versão atual do pipeline desde 1º de fevereiro de 2024.



O diagrama a seguir ilustra um exemplo de estágio que foi revertido, exibindo o novo ID de execução.



Reverter um estágio manualmente (CLI)

Para usar o AWS CLI para reverter manualmente um estágio, use o `rollback-stage` comando.

Também é possível reverter um estágio manualmente, conforme detalhado em Reverter um estágio manualmente.

> **Note**
>
> A lista de execuções de pipeline disponíveis como destino incluirá todas as execuções iniciadas na versão atual do pipeline desde 1º de fevereiro de 2024.

Como reverter um estágio manualmente (CLI)

1. Para realizar a reversão manual via CLI, será necessário fornecer o ID de execução de uma execução de pipeline bem-sucedida previamente concluída no estágio. Para obter o ID de execução do pipeline de destino que você especificará, use o comando list-pipeline-executions com um filtro que retornará as execuções bem-sucedidas no estágio. Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use o AWS CLI para executar o `list-pipeline-executions` comando, especificando o nome do pipeline e o filtro para execuções bem-sucedidas no estágio. Neste exemplo, a saída listará as execuções de pipeline para o pipeline nomeado MyFirstPipeline e para execuções bem-sucedidas no estágio nomeado. `deploys3`

   ```
   aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline --filter
    succeededInStage={stageName=deploys3}
   ```

   Na saída, copie o ID da execução bem-sucedida anterior que você deseja especificar para reversão. Você o usará na próxima etapa como ID de execução de destino.

2. Acesse um terminal (Linux, macOS ou Unix) ou o prompt de comando (Windows) e use o AWS CLI para executar o comando `rollback-stage`, especificando o nome do pipeline, o nome do estágio e a execução de destino que será utilizada para a reversão. Por exemplo, para reverter um estágio chamado Deploy para um pipeline chamado*`MyFirstPipeline`*:

   ```
   aws codepipeline rollback-stage --pipeline-name MyFirstPipeline --stage-name Deploy
    --target-pipeline-execution-id bc022580-4193-491b-8923-9728dEXAMPLE
   ```

   A saída apresenta o ID da nova execução após a reversão. Trata-se de um ID distinto que aproveita as revisões de origem e os parâmetros da execução de destino definida.

Configurar um estágio para reversão automática

Você pode configurar estágios em um pipeline para reverter automaticamente em caso de falha. Em caso de falha, o estágio retorna à execução bem-sucedida mais recente. O pipeline só pode realizar a reversão para uma execução anterior caso ela tenha sido iniciada na versão vigente da estrutura do pipeline. Como a configuração de reversão automática faz parte da definição do pipeline, seu estágio de pipeline será revertido automaticamente somente após uma execução bem-sucedida do pipeline no estágio do pipeline.

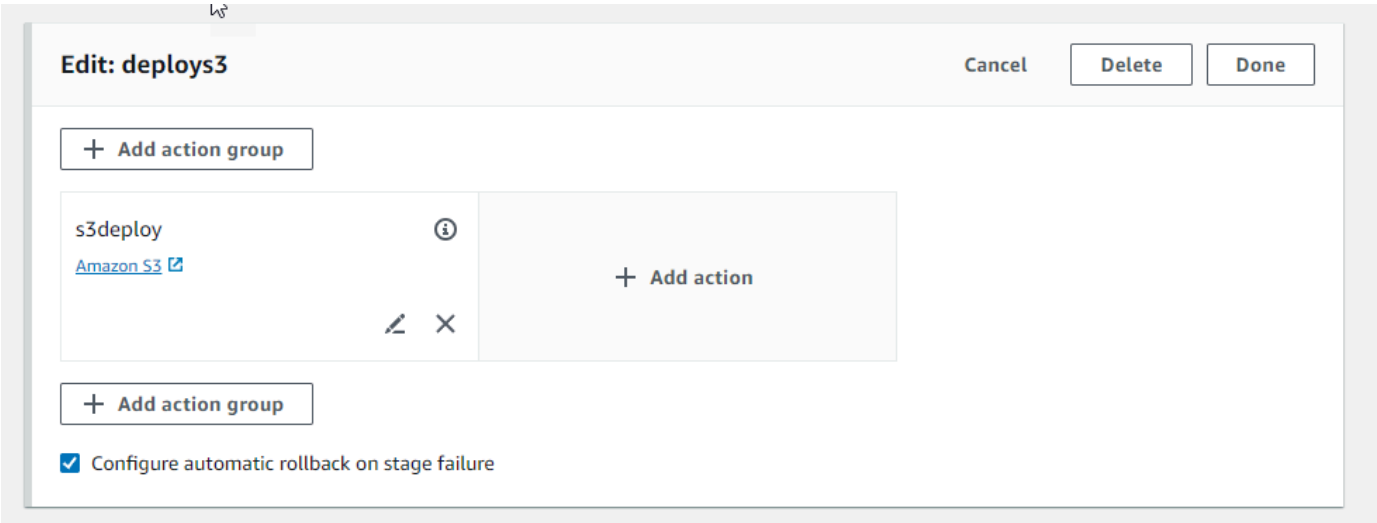
Configurar um estágio para reversão automática (console)

É possível reverter um estágio para uma execução anterior bem-sucedida definida. Para obter mais informações, consulte RollbackStage o Guia de CodePipeline API.

Configurar um estágio para reversão automática (console)

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes e o status de todos os canais associados à sua AWS conta são exibidos.
2. Em Nome, selecione o nome do pipeline que você deseja editar.
3. Na página de detalhes do pipeline, selecione Editar.
4. Na página Editar, para a ação que você deseja editar, escolha Editar estágio.
5. Escolha Configuração de etapa automatizada: e Configurar reversão automática em caso de falha de estágio. Salve as alterações no pipeline.



Configurar um estágio para reversão automática (CLI)

Para usar o AWS CLI para configurar um estágio com falha para reverter automaticamente para a execução bem-sucedida mais recente, use os comandos para criar ou atualizar um pipeline conforme detalhado em Criar um pipeline, estágios e ações Editar um pipeline no CodePipeline e.

- Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use o AWS CLI para executar o comando `update-pipeline`, especificando a condição de falha na estrutura do pipeline. O exemplo a seguir configura a reversão automática para um estágio chamado: `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 obter mais informações sobre como configurar as condições de falha para a reversão de estágios, consulte a Referência FailureConditions da CodePipeline API.

Configurar um estágio para reversão automática (CloudFormation)

CloudFormation Para configurar um estágio para reverter automaticamente em caso de falha, use o `OnFailure` parâmetro. Em caso de falha, o estágio retornará automaticamente para à última execução bem-sucedida.

```
OnFailure:
     Result: ROLLBACK
```

- Atualize o modelo conforme mostrado no snippet a seguir. O exemplo a seguir configura a reversão automática para um estágio chamado: `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 obter mais informações sobre como configurar as condições de falha para a reversão de estágios, consulte OnFailure abaixo `StageDeclaration` no Guia do CloudFormation usuário.

Exibir o status de reversão na lista de execução

É possível verificar o status e o ID de execução de destino de uma reversão em andamento.

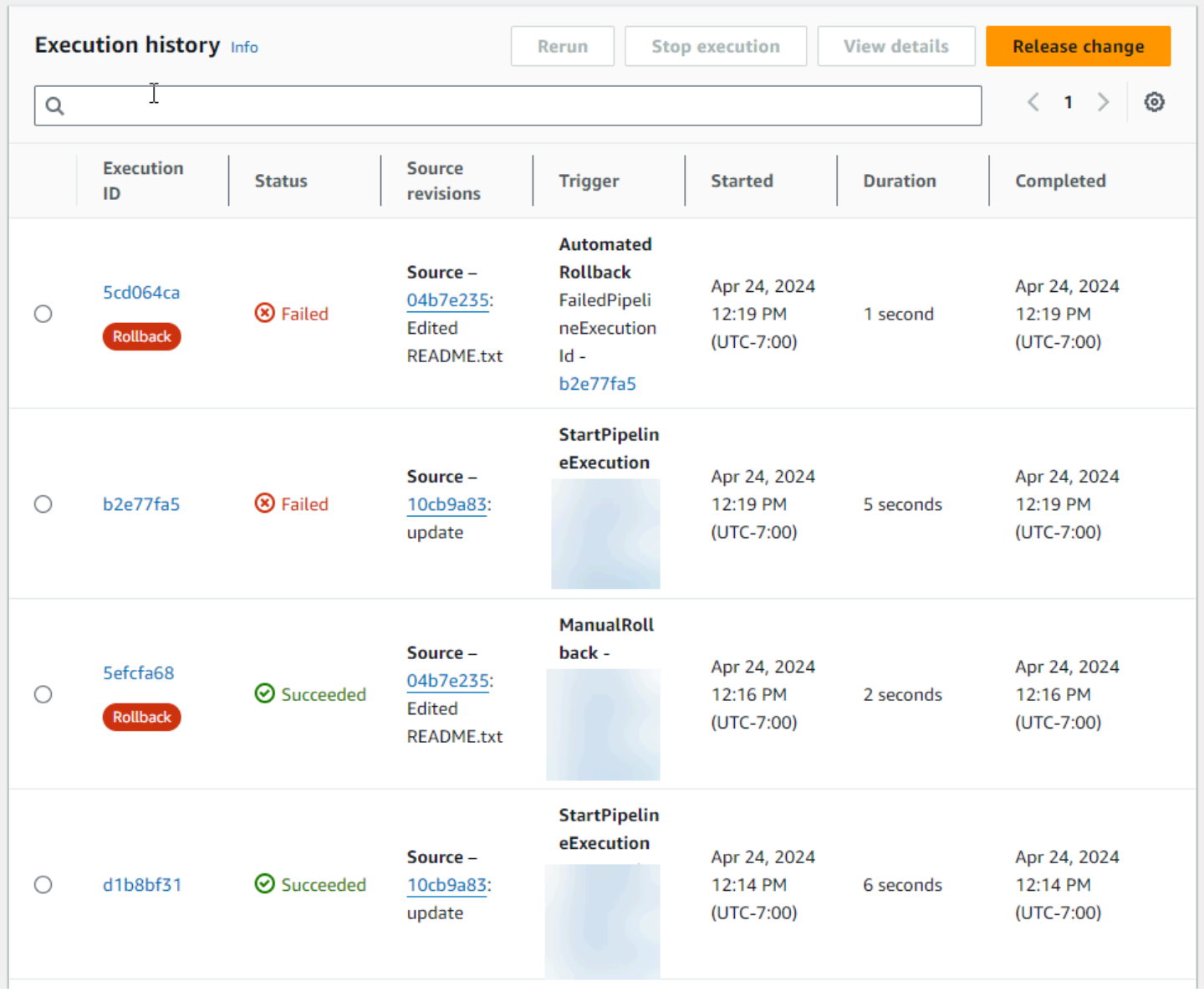
Exibir status de reversão na lista de execuções (console)

No console, é possível visualizar o status e o ID de execução de destino de uma reversão na listagem de execuções.

Exibir status da execução da reversão na lista de execuções (console)

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes e status de todos os pipelines vinculados à sua Conta da AWS são exibidos.
2. Em Nome, selecione o nome do pipeline que você deseja ver.
3. Escolha History (Histórico). A lista de execuções mostra o rótulo Reversão.



Selecione o ID de execução cujos detalhes você deseja visualizar.

Visualize o status de reversão com list-pipeline-executions (CLI)

Você pode usar a CLI para visualizar o status e o ID de execução de destino para uma execução de reversão.

- Abra um terminal (Linux, macOS ou Unix) ou um prompt de comando (Windows) e use a AWS CLI para executar o comando list-pipeline-executions:

```
aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline
```

Esse comando gera uma lista de todas as execuções finalizadas vinculadas ao pipeline.

O exemplo a seguir mostra os dados retornados de um pipeline chamado *MyFirstPipeline* onde uma execução de reversão iniciou o pipeline.

```
{
    "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"
```

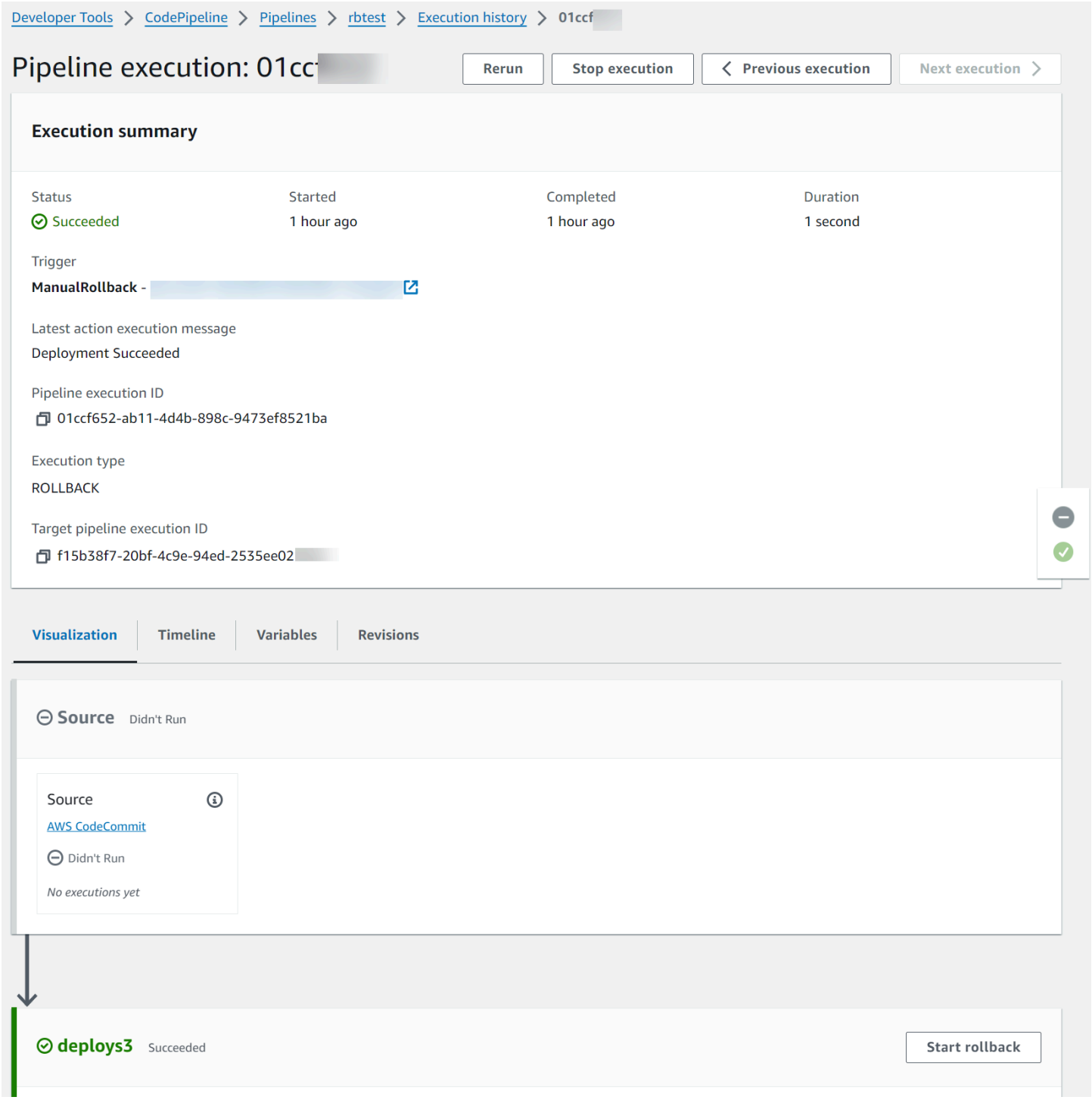
```
                }
            ],
            "trigger": {
                "triggerType": "StartPipelineExecution",
                "triggerDetail": "arn:aws:sts::<account_ID>:assumed-role/<role>"
            },
            "executionMode": "SUPERSEDED"
        },
        {
            "pipelineExecutionId": "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"
            }
         },
```

Exibir detalhes do status da reversão

É possível verificar o status e o ID de execução de destino de uma reversão em andamento.

Exibir o status da reversão na página de detalhes (console)

É possível utilizar o console para verificar o status e o ID de execução do pipeline de destino em uma reversão.



Veja os detalhes da reversão com get-pipeline-execution (CLI)

As execuções de pipeline revertidas serão exibidas no resultado ao consultar a execução do pipeline.

- Para visualizar detalhes sobre um pipeline, execute o comando get-pipeline-execution, especificando o nome exclusivo do pipeline. Por exemplo, para visualizar detalhes sobre um pipeline designado *MyFirstPipeline*, insira o seguinte:

  ```
  aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-execution-id 3f658bd1-69e6-4448-ba3e-79007EXAMPLE
  ```

  Este comando retorna a estrutura do pipeline.

  O exemplo a seguir mostra os dados retornados de uma parte de um pipeline chamada*MyFirstPipeline*, em que o ID de execução da reversão e os metadados são mostrados.

  ```
  {
      "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"
          }
  ```

```
    }
}
```

Visualize o estado de reversão com get-pipeline-state (CLI)

As execuções de pipeline revertidas serão exibidas no resultado ao consultar o estado do pipeline.

- Para visualizar detalhes sobre um pipeline, execute o comando get-pipeline-state, especificando o nome exclusivo do pipeline. Por exemplo, para ver os detalhes do estado de um pipeline chamado*MyFirstPipeline*, insira o seguinte:

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

  O exemplo a seguir mostra os dados gerados com o tipo de execução de reversão.

  ```
  {
      "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"
}
```

Configurar condições para um estágio

Você pode especificar uma condição para um estágio, como verificar uma variável específica na execução do pipeline e, em seguida, incluir um resultado para a condição, como pular ou reprovar o estágio. Um pipeline pode ser configurado para verificar as condições do estágio durante a execução ao especificar as verificações de um estágio e, em seguida, especifica como o estágio deve continuar quando determinadas condições forem atendidas. As condições contêm uma ou mais regras que estão disponíveis em uma lista de regras em CodePipeline. Se todas as regras em uma condição forem bem-sucedidas, a condição será atendida. É possível definir condições para que, caso os critérios não sejam cumpridos, o resultado especificado seja ativado.

Toda condição contém um conjunto estruturado de regras, organizadas em uma ordem específica e avaliadas simultaneamente. Portanto, se uma regra falhar na condição, a condição falhará. Você pode substituir as condições da regra no runtime do pipeline.

As condições são usadas para tipos específicos de expressões e cada uma tem opções específicas para resultados disponíveis da seguinte forma:

- Entrada: as condições para fazer verificações que, se atendidas, permitem a entrada em um estágio. As regras são usadas com as seguintes opções de resultado: Falhar ou Ignorar
- Em caso de falha: as condições para fazer verificações para o estágio em que ela falha. As regras estão envolvidas com a seguinte opção de resultado: Reversão
- Bem-sucedido: as condições para fazer verificações para o estágio em que for bem-sucedida. As regras são envolvidas com as seguintes opções de resultado: Reversão ou Falha

As condições são compatível com um conjunto de regras para cada tipo de condição.

Para cada tipo de condição, há ações específicas configuradas pela condição. A ação é o resultado da verificação de condição bem-sucedida ou com falha. Por exemplo, a condição de entrada (condição de entrada) encontra um alarme (regra), então a verificação é bem-sucedida e o resultado (ação) é que a entrada do estágio está bloqueada.

Você também pode usar o AWS CodePipeline console ou o AWS CLI para reverter manualmente ou tentar novamente um estágio ou ações em um estágio. Consulte Configurar condições para um estágio.

Tópicos

- Casos de uso de condições de estágio
- Considerações sobre resultados configurados para condições de estágio
- Considerações sobre regras configuradas para condições de estágio
- Criar condições de entrada
- Criar condições “em caso de falha”
- Criar condições “em caso de sucesso”
- Excluir condições de estágio
- Condições de estágio de substituição

Casos de uso de condições de estágio

As condições de estágio têm vários casos de uso para configurar a segurança de liberação e alteração em pipelines. A seguir estão alguns exemplos de casos de uso para condições de estágio.

- Use uma condição de entrada para definir uma condição que verificará o estado do CloudWatch alarme, o que bloqueará uma alteração se o ambiente de produção não estiver em um estado saudável.
- Use uma condição de entrada com um tempo de espera de 60 para definir uma condição a ser avaliada quando todas as ações em um estágio tiverem sido concluídas com êxito e, em seguida, reverta as alterações se um CloudWatch alarme entrar no estado ALARM em 60 minutos.
- Configure uma condição “em caso de sucesso” para garantir que, ao concluir um estágio com êxito, seja verificado se o horário atual está dentro da janela de implantação, procedendo com a implantação somente se a condição for atendida.

Considerações sobre resultados configurados para condições de estágio

As considerações sobre as condições do palco são as seguintes:

- Não é possível usar a repetição automática de estágio com condições onFailure.
- Ao configurar uma condição com um resultado Reversão, o estágio só pode reverter para uma execução anterior se disponível na versão atual da estrutura do pipeline.
- Ao configurar uma condição com um resultado Reversão, você não pode reverter para um ID de execução de destino que seja do tipo de execução de reversão.

- Para condições de entrada que usam o resultado Ignorar para ignorar o estágio se a condição falhar, somente as regras `LambdaInvoke` e `VariableCheck` são compatíveis.
- Não é possível executar uma nova tentativa manual de estágio em um estágio com status Ignorado.
- Não é possível executar uma reversão manual para um estágio no status Ignorado.
- Não é possível substituir uma condição se ela estiver configurada com um resultado Ignorar.
- Com exceção dos resultados Ignorar, você pode substituir uma condição de estágio ao iniciar uma execução de pipeline. Em uma condição de estágio em que uma substituição é ativada, a execução será realizada conforme detalhado na tabela a seguir.
-

Tipo	Resultado configurado em caso de falha de condição	Status de estágio	Comportamento de substituição
Entrada	Falha	In-progress	A etapa continua.
Entrada	Ignorar	Ignorado	Não aplicável.
OnFailure	Reversão	Failed	O estágio falhou.
OnSuccess	Reversão	Bem-sucedido	A etapa continua.
OnSuccess	Falha	Failed	A etapa continua.

Considerações sobre regras configuradas para condições de estágio

As considerações sobre as regras disponíveis para condições de estágio são as seguintes:

- Na regra `LambdaInvoke`, você deve primeiro configurar a função do Lambda a ser usada na regra. Tenha o ARN da função do Lambda pronto para ser fornecido ao configurar a regra.
- Para a `CloudWatchAlarm` regra, você deve primeiro configurar o evento CloudWatch Events para ser usado na regra. Tenha o ARN do evento pronto para ser fornecido ao configurar a regra.

Criar condições de entrada

Você pode configurar condições de entrada para um estágio usando o console ou a CLI. Você configurará as regras e os resultados correspondentes para cada condição. Em um resultado de reversão, o pipeline só poderá retornar a uma execução anterior se ela tiver sido iniciada na versão atual da estrutura do pipeline.

As etapas fornecem um exemplo de condição de entrada que usa uma regra de monitor.

Para obter mais informações, consulte Condição RuleTypeId, e RuleExecution no Guia CodePipeline da API.

Criação de condições de entrada - exemplo de CloudWatchAlarm regra (console)

Você pode configurar condições de entrada para um estágio com as regras e os resultados que você deseja que o estágio execute quando as condições forem atendidas.

Configurar uma condição Entrada (console)

1. Preencha todos os pré-requisitos, como criar o recurso e o ARN para uma regra em que um recurso é fornecido, como o. AWS CloudWatchAlarm
2. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes e status de todos os pipelines vinculados à sua Conta da AWS são exibidos.
3. Em Nome, selecione o nome do pipeline que você deseja editar.
4. Na página de detalhes do pipeline, selecione Editar.
5. Na página Editar, para a ação que você deseja editar, escolha Editar estágio.
6. Selecione Adicionar condição de entrada. O cartão de Condição de entrada antes do estágio é exibido com a opção Falha disponível para essa condição.
7. Selecione Adicionar regra e faça o seguinte.

   a. Em Nome da regra, insira um nome para a regra. Neste exemplo, insira `MyAlarmRule`.

   b. Em Provedor de regras, escolha o provedor de regras pré-configurado para adicionar à condição. Para este exemplo AWS CloudWatchAlarm, escolha e conclua as etapas a seguir.

c. Em Região, escolha a Região para a condição ou deixe o padrão.

d. Em Nome do alarme, escolha o CloudWatch recurso a ser usado para a regra. Você já deve ter criado o recurso na conta.

e. (Opcional) Em Tempo de espera, insira o tempo de CodePipeline espera se o alarme estiver no estado ALARME quando for avaliado pela primeira vez. Se o alarme estiver no estado OK quando a regra for verificada pela primeira vez, a regra será imediatamente bem-sucedida.

f. (Opcional) Insira o estado de alarme específico para monitorar e insira o ARN do perfil, se apropriado.

g. Ao concluir a edição do cenário, selecione Concluído. Na página de edição do pipeline, selecione Salvar.

8. Após a execução, veja o resultado.

Criar condições de entrada com resultado Ignorar e regra **`VariableCheck`** (console)

Você pode configurar as condições de entrada para um estágio de forma que, se a condição de entrada não for atendida, o estágio seja ignorado. Se a condição falhar, o resultado será ativado e o estágio será ignorado. Quando um estágio é ignorado, o status do estágio é Ignorado e o status da ação é Não executado. Em considerações sobre condições de estágio com resultados Ignorar, consulte Considerações sobre resultados configurados para condições de estágio.

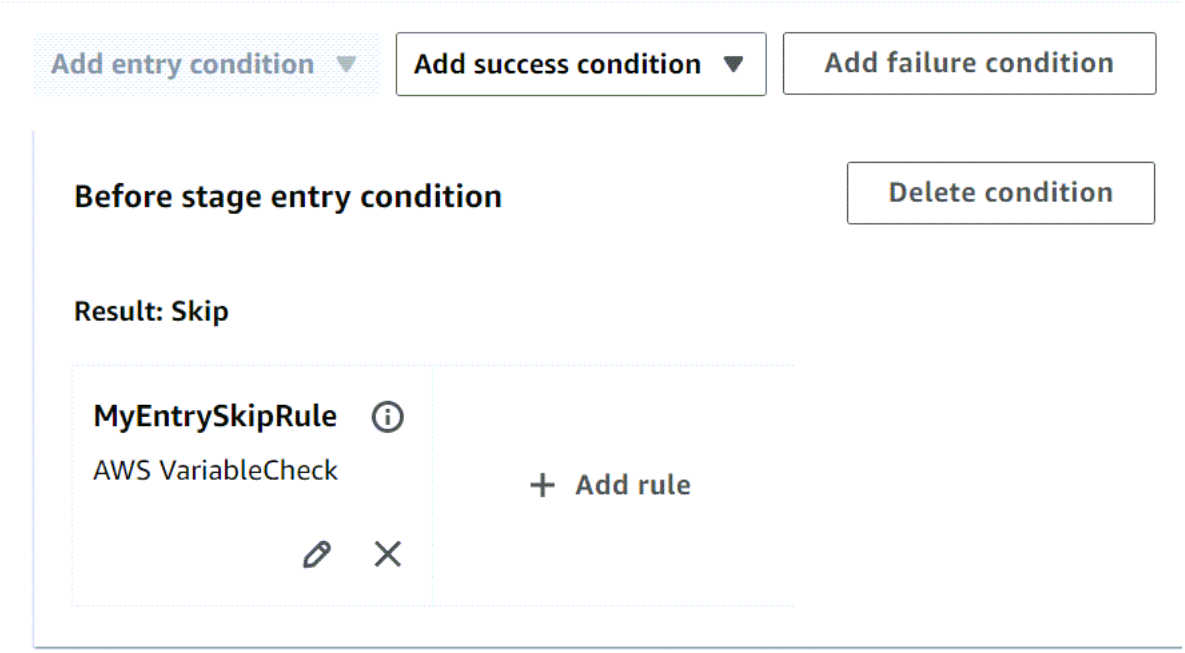
No exemplo a seguir, a regra de verificação de variável descobre que o valor não é uma correspondência e o estágio de compilação é ignorado.

Configurar uma condição de entrada com um Ignorar resultado (console)

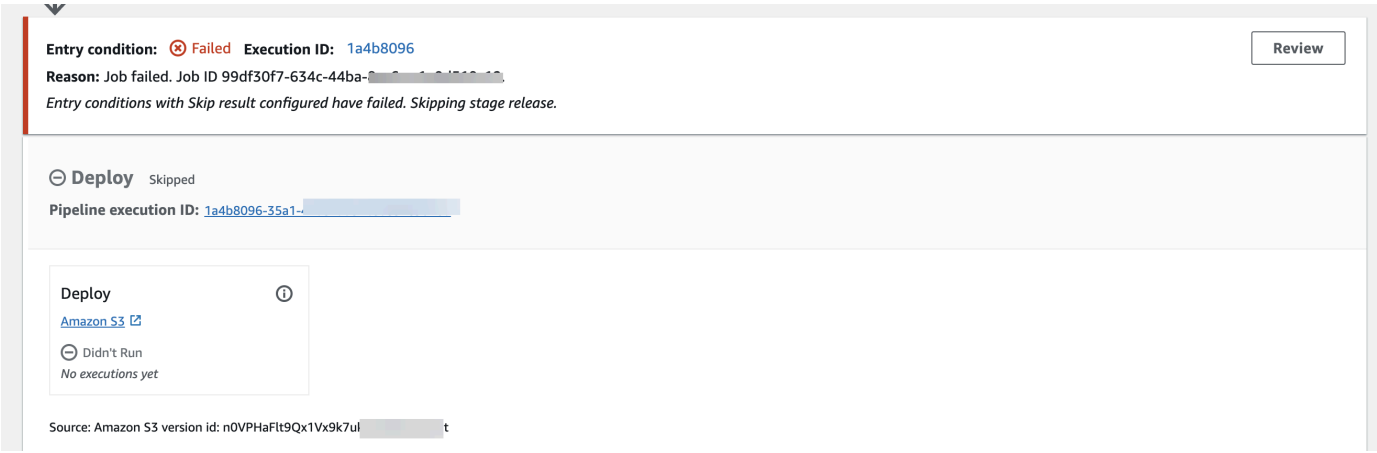
1. Preencha todos os pré-requisitos, como criar o recurso e o ARN para uma regra em que um recurso é fornecido, como o. AWS CloudWatchAlarm
2. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes e status de todos os pipelines vinculados à sua Conta da AWS são exibidos.
3. Em Nome, selecione o nome do pipeline que você deseja editar.
4. Na página de detalhes do pipeline, selecione Editar.

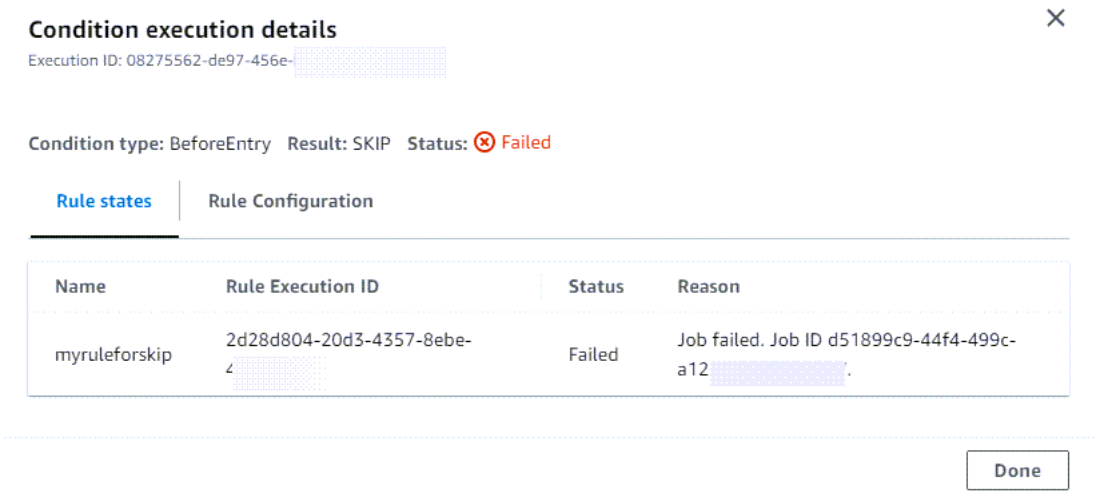
5. Na página Editar, para a ação que você deseja editar, escolha Editar estágio.
6. Selecione Adicionar condição de entrada e, em seguida, escolha Ignorar como resultado.
7. Selecione Adicionar regra e faça o seguinte.
   a. Em Nome da regra, insira um nome para a regra. Neste exemplo, insira `MyAlarmRule`.
   b. Em Provedor de regras, escolha o provedor de regras pré-configurado para adicionar à condição. Para este exemplo VariableCheck, escolha e conclua as etapas a seguir.



   c. Em Região, escolha a Região para a condição ou deixe o padrão.
   d. Em Variável, escolha a variável com a qual comparar, como `#{SourceVariables.FullRepositoryName}` para um pipeline que tem uma ação de origem GitHub (via GitHub aplicativo). Insira o nome do repositório e escolha o operador, como Equals.
   e. Ao concluir a edição do cenário, selecione Concluído. Na página de edição do pipeline, selecione Salvar.
8. Após a execução, veja o resultado.



9. Para analisar os detalhes, selecione Revisar. O detalhe no exemplo a seguir mostra que o resultado configurado para a condição é Ignorar, que não pode ser substituído. O status da regra é Falha porque a condição não foi atendida.



Criar condições de entrada (CLI)

Para usar o AWS CLI para configurar uma condição de entrada, use os comandos para criar ou atualizar um pipeline conforme detalhado em Criar um pipeline, estágios e ações Editar um pipeline no CodePipeline e.

Configurar a condição e as regras (CLI)

- Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use o AWS CLI para executar o comando `update-pipeline`, especificando a condição de falha na estrutura do pipeline. O exemplo a seguir configura uma condição de entrada para um estágio chamado`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 obter mais informações sobre como configurar as condições de sucesso para a reversão de estágios, consulte a Referência SuccessConditions da CodePipeline API.

Criar condições de entrada (CFN)

Para usar CloudFormation para configurar uma condição de entrada, use o `beforeEntry` parâmetro. Na entrada, o palco executará a regra e executará o resultado.

```
beforeEntry:
     Result: FAIL
```

- Atualize o modelo conforme mostrado no snippet a seguir. O exemplo a seguir configura uma condição de entrada com uma regra chamada `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 obter mais informações sobre como configurar as condições BeforeEntry, consulte AWS::CodePipeline: :Pipeline BeforeEntryConditions abaixo `StageDeclaration` no Guia do CloudFormation usuário.

Criar condições “em caso de falha”

Você pode configurar condições “em caso de falha” para um estágio usando o console ou a CLI. Você configurará as regras e os resultados correspondentes para cada condição. Em um resultado de reversão, o pipeline só poderá retornar a uma execução anterior se ela tiver sido iniciada na versão atual da estrutura do pipeline.

Criar condições “em caso de falha” (console)

Você pode configurar condições “em caso de falha” para um estágio com as regras e os resultados que você deseja que o estágio execute quando as condições forem atendidas.

Configurar uma condição “em caso de falha” (console)

1. Preencha todos os pré-requisitos, como criar o recurso e o ARN para uma regra em que um recurso é fornecido, como a regra. LambdaInvoke
2. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

Os nomes e status de todos os pipelines vinculados à sua Conta da AWS são exibidos.

3. Em Nome, selecione o nome do pipeline que você deseja editar.
4. Na página de detalhes do pipeline, selecione Editar.
5. Na página Editar, para a ação que você deseja editar, escolha Editar estágio.
6. Selecione Adicionar condição de falha. O cartão de Condição de falha é exibido com a opção Reverter disponível para essa condição.
7. Selecione Adicionar regra e faça o seguinte.

    a. Em Nome da regra, insira um nome para a regra. Neste exemplo, insira `MyLambdaRule`.
    b. Em Provedor de regras, escolha o provedor de regras pré-configurado para adicionar à condição. Para este exemplo AWS LambdaInvoke, escolha e conclua as etapas a seguir.
    c. Em Região, escolha a Região para a condição ou deixe o padrão.
    d. Em Artefatos de entrada, escolha o artefato de criação.
    e. Em Nome da função, escolha o recurso Lambda a ser usado para a regra. Você já deve ter criado o recurso na conta.
    f. (Opcional) Em Parâmetros do usuário, insira os pares que representem parâmetros para configuração adicional.
    g. (Opcional) Em ARN do perfil, insira o ARN do perfil, se configurado.
    h. (Opcional) Em Tempo limite em minutos, insira o tempo em minutos que a regra deve esperar antes do tempo limite.
    i. Ao concluir a edição do cenário, selecione Concluído. Na página de edição do pipeline, selecione Salvar.

Criar condições onFailure com um exemplo de resultado de repetição (console)

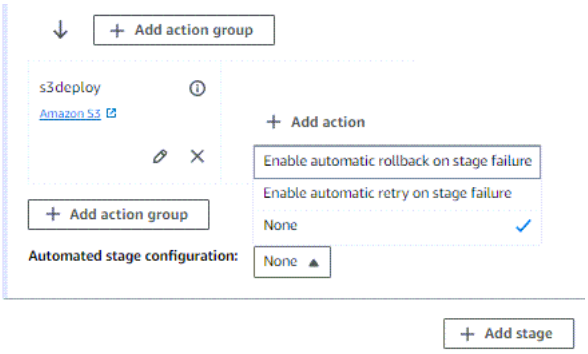
É possível configurar condições onFailure para um estágio, permitindo que ele seja reexecutado caso a condição de entrada não seja satisfeita. Como parte desse resultado, você configura o modo de repetição, especificando se deseja repetir as ações com falha ou repetir o estágio com falha.

Configurar uma condição onFailure com um resultado de repetição (console)

1. Preencha todos os pré-requisitos, como criar o recurso e o ARN para uma regra em que um recurso é fornecido, como o. AWS CloudWatchAlarm

2. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes e status de todos os pipelines vinculados à sua Conta da AWS são exibidos.
3. Em Nome, selecione o nome do pipeline que você deseja editar.
4. Na página de detalhes do pipeline, selecione Editar.
5. Na página Editar, para a ação que você deseja editar, escolha Editar estágio.
6. Na parte inferior do estágio, em Configuração de etapa automatizada:, escolha Habilitar nova tentativa automática em caso de falha na etapa. Em Modo Repetir, escolha Repetir estágio com falha ou Repetir ações com falha.



7. Escolha adicionar uma condição onFailure, depois selecione Adicionar regra e insira uma regra para a condição.

   a. Em Nome da regra, insira um nome para a regra. Neste exemplo, insira `MyAlarmRule`.
   b. Em Provedor de regras, escolha o provedor de regras pré-configurado para adicionar à condição. Para este exemplo CloudWatchAlarm, escolha e conclua as etapas a seguir.
   c. Em Região, escolha a Região para a condição ou deixe o padrão.
   d. Em Nome do alarme, escolha o recurso configurado para o alerta.
   e. Ao concluir a edição do cenário, selecione Concluído. Na página de edição do pipeline, selecione Salvar.
8. Após a execução, veja o resultado.

Criar condições “em caso de falha” (CLI)

Para usar o AWS CLI para configurar uma condição On Failure, use os comandos para criar ou atualizar um pipeline conforme detalhado em Criar um pipeline, estágios e ações Editar um pipeline no CodePipeline e.

Configurar a condição e as regras (CLI)

- Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use o AWS CLI para executar o comando `update-pipeline`, especificando a condição de falha na estrutura do pipeline. O exemplo a seguir configura uma condição On Failure para um estágio chamado`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 obter mais informações sobre como configurar condições de falha, consulte FailureConditions a Referência da CodePipeline API.

Criar condições “em caso de falha” (CFN)

CloudFormation Para configurar uma condição On Failure, use o `OnFailure` parâmetro. Em caso de falha, o estágio executará a regra e executará o resultado.

```
OnFailure:
     Result: ROLLBACK
```

- Atualize o modelo conforme mostrado no snippet a seguir. O exemplo a seguir configura uma OnFailure condição com uma regra chamada`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 obter mais informações sobre como configurar condições de falha, consulte OnFailure abaixo `StageDeclaration` no Guia do CloudFormation usuário.

Criar condições “em caso de sucesso”

Você pode configurar condições “em caso de sucesso” para um estágio usando o console ou a CLI. Você configurará as regras e os resultados correspondentes para cada condição. Em um resultado de reversão, o pipeline só poderá retornar a uma execução anterior se ela tiver sido iniciada na versão atual da estrutura do pipeline.

Os estágios fornecem um exemplo de condição “em caso de sucesso” que usa uma regra de janela de implantação.

Para obter mais informações, consulte Condição RuleTypeId, e RuleExecution no Guia CodePipeline da API.

Criar condições “em caso de sucesso” (console)

Você pode configurar condições “em caso de sucesso” para um estágio com as regras e os resultados que você deseja que o estágio execute quando as condições forem atendidas.

Configurar uma condição “em caso de sucesso” (console)

1. Preencha todos os pré-requisitos, como criar o recurso e o ARN para uma regra em que um recurso é fornecido, como o. AWS LambdaRule
2. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

   Os nomes e status de todos os pipelines vinculados à sua Conta da AWS são exibidos.
3. Em Nome, selecione o nome do pipeline que você deseja editar.
4. Na página de detalhes do pipeline, selecione Editar.
5. Na página Editar, para a ação que você deseja editar, escolha Editar estágio.
6. Selecione Adicionar condição de êxito. O cartão de Condição de sucesso no estágio é exibido. Selecione a opção Reversão ou Falha exibida como os resultados disponíveis para esse tipo de condição.
7. Selecione Adicionar regra e faça o seguinte.

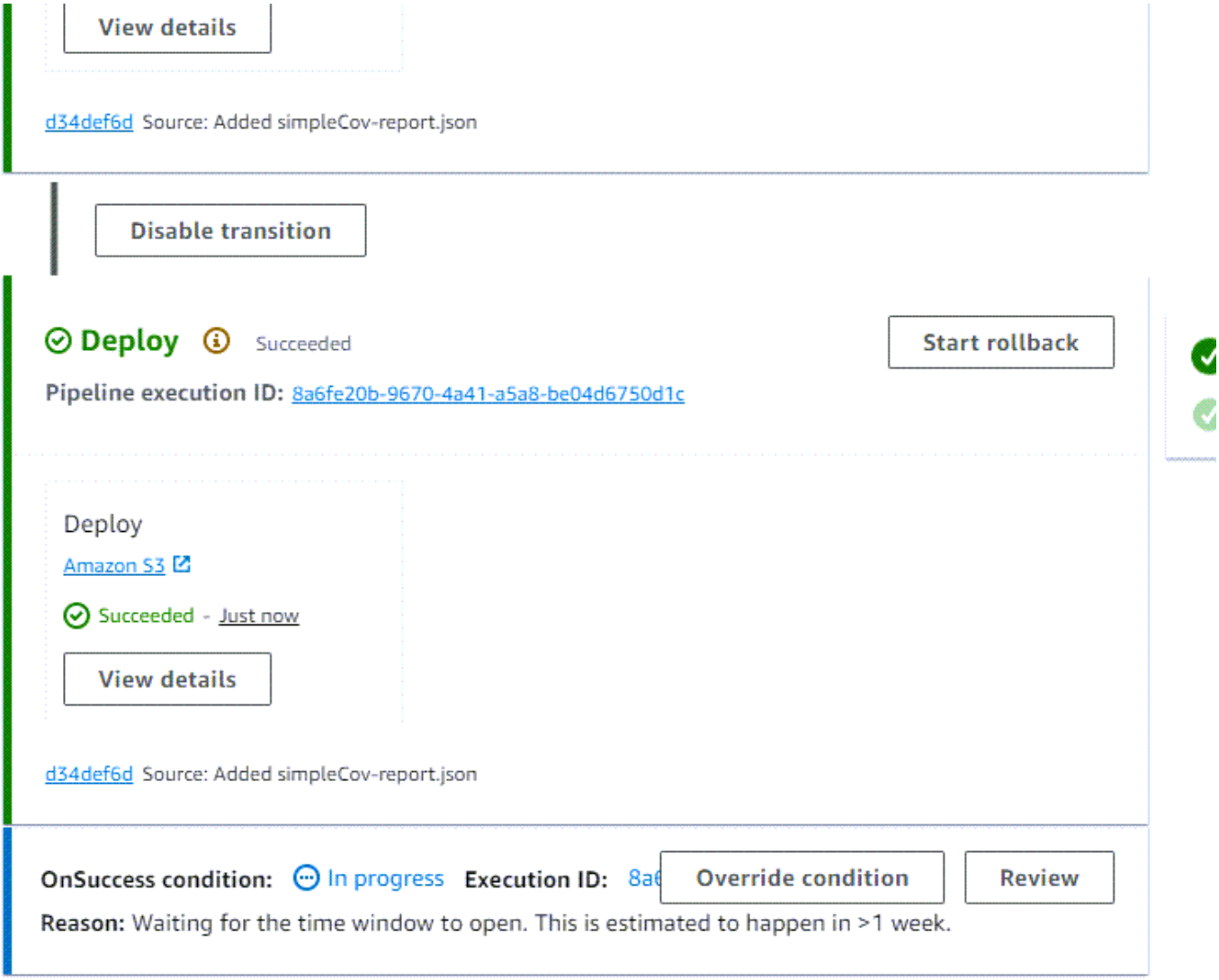
   a. Em Nome da regra, insira um nome para a condição. Neste exemplo, insira `MyDeploymentRule`.

   b. Em Provedor de regras, selecione a regra pré-configurada para adicionar à condição. Para este exemplo AWS DeploymentWindow, escolha e conclua as etapas a seguir.

   c. Em Região, escolha a Região para a condição ou deixe o padrão.

   d. Em Cron, insira a expressão cron para a janela de implantação. A expressão cron define os dias e horários em que a implantação deve ser permitida. Para ter informações de referência sobre expressões cron, consulte Using cron and rate expressions to schedule rules.

   e. (Opcional) Em TimeZone, insira o fuso horário da janela de implantação.
8. Após a execução, veja o resultado.



Criar condições “em caso de sucesso” (CLI)

Para usar o AWS CLI para configurar uma condição On Success, use os comandos para criar ou atualizar um pipeline conforme detalhado em Criar um pipeline, estágios e ações Editar um pipeline no CodePipeline e.

Configurar a condição e as regras (CLI)

- Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use o AWS CLI para executar o comando `update-pipeline`, especificando a condição de falha na

estrutura do pipeline. O exemplo a seguir configura uma condição On Success para um estágio chamado`Deploy`, em que a regra é nomeada`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 obter mais informações sobre como configurar as condições de sucesso, consulte SuccessConditions na Referência da CodePipeline API.

Criar uma condição “em caso de sucesso” (CFN)

CloudFormation Para configurar uma condição On Success, use o `OnSuccess` parâmetro. Em caso de sucesso, o estágio executará a regra e executará o resultado.

```
OnSuccess:
     Result: ROLLBACK
```

- Atualize o modelo conforme mostrado no snippet a seguir. O exemplo a seguir configura uma OnSuccess condição com uma regra chamada`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 obter mais informações sobre como configurar as condições de falha para a reversão de estágios, consulte OnFailure abaixo `StageDeclaration` no Guia do CloudFormation usuário.

Excluir condições de estágio

Você pode excluir condições de estágio que foram configuradas para o pipeline.

Como excluir uma condição de estágio

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

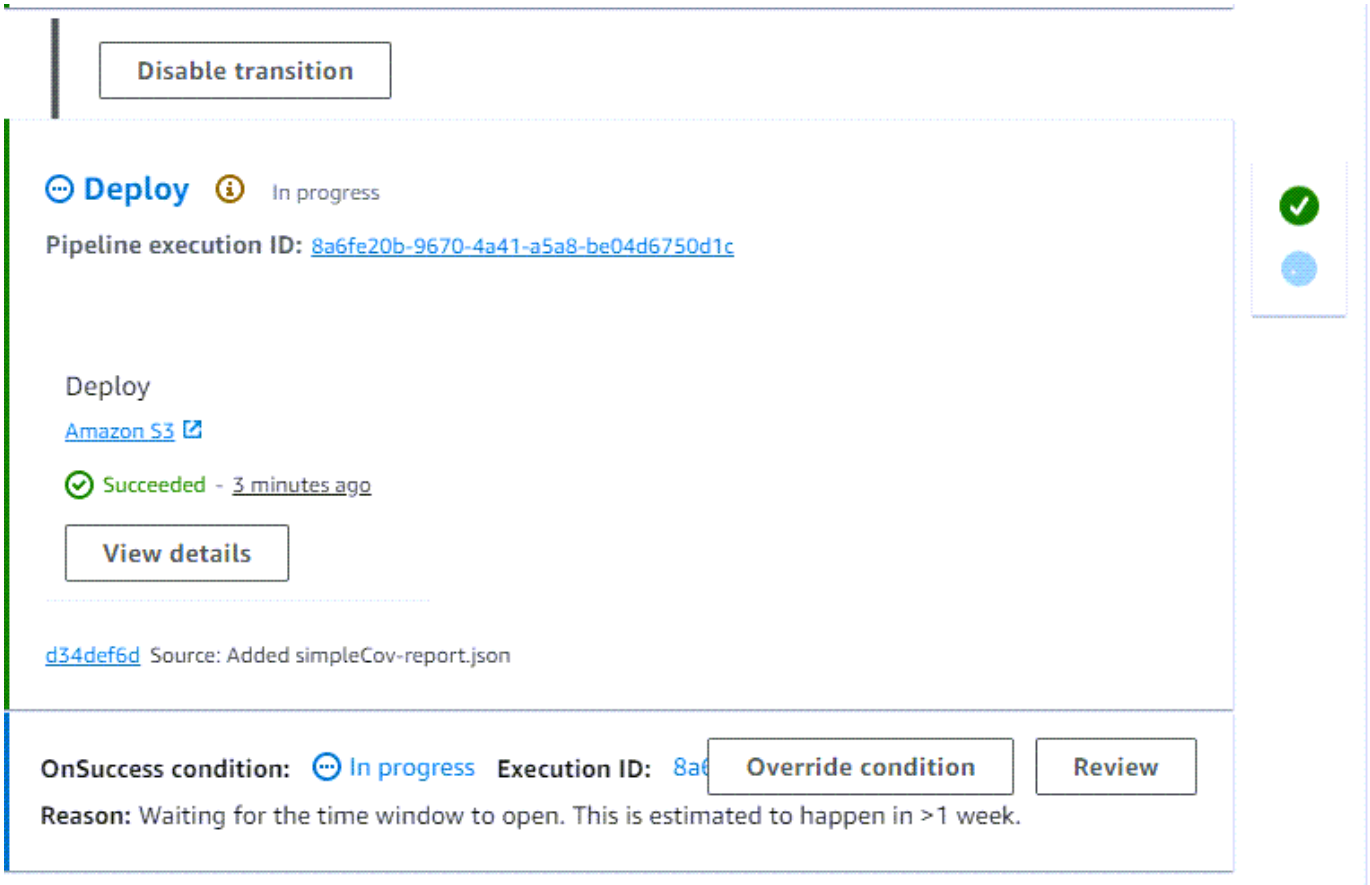
   Os nomes e o status de todos os pipelines associados ao seu Conta da AWS são exibidos.
2. Em Nome, selecione o nome do pipeline que você deseja editar.
3. Na página de detalhes do pipeline, selecione Editar.
4. Na página Editar, para a condição que você deseja editar, escolha Editar estágio.
5. Ao lado da condição que você deseja excluir, selecione Excluir condição.

Condições de estágio de substituição

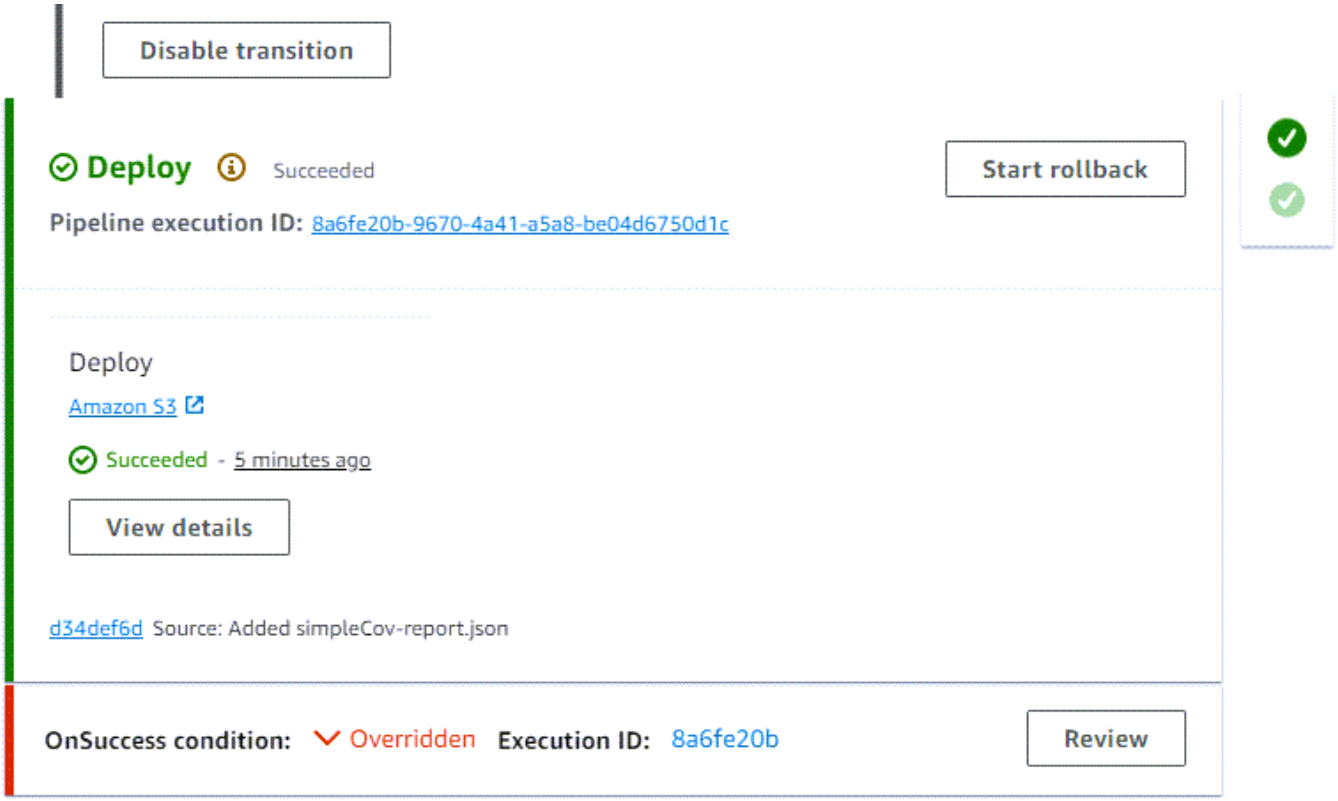
Você pode substituir as condições do estágio que foram configuradas para o pipeline. No console, quando o estágio e a regra estão em execução, você pode substituir a condição do estágio. Isso resulta na corrida por etapas.

Como substituir uma condição de estágio

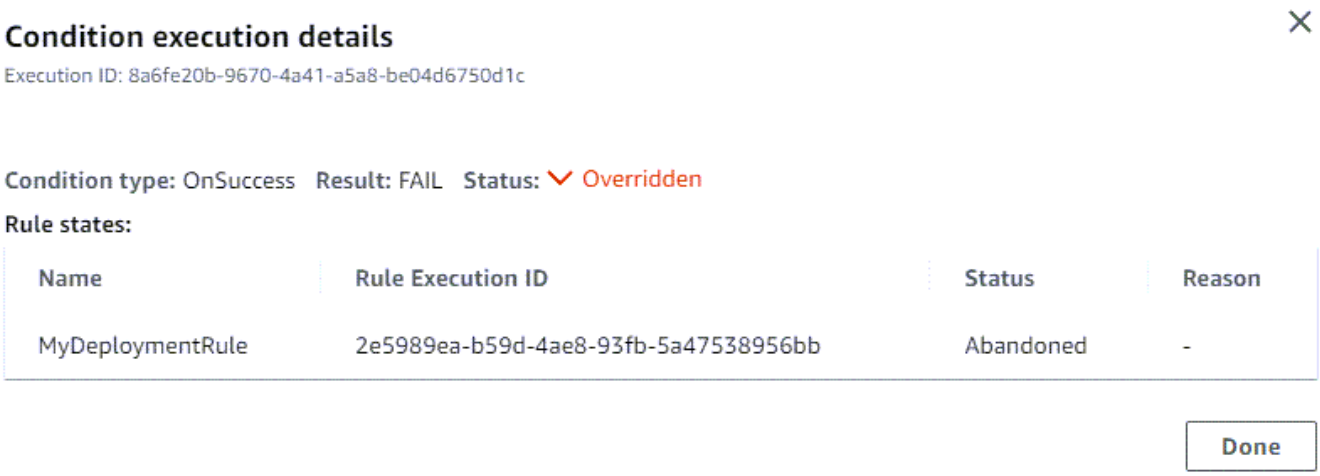
1. Neste exemplo, o estágio do pipeline está sendo executado com uma condição. O botão Substituir está ativado.



2. Ao lado da condição que você deseja substituir, escolha Substituir.



3. Para analisar os detalhes, selecione Revisar. O detalhe no exemplo a seguir mostra que o resultado configurado para a condição é Falha, que foi substituído. O status da regra é Abandonado devido à substituição.



Usar tipos de ação, ações personalizadas e ações de aprovação

Em AWS CodePipeline, uma ação faz parte da sequência em um estágio de um pipeline. É uma tarefa executada no artefato no estágio em questão. As ações do pipeline ocorrem em uma ordem específica, em sequência ou em paralelo, conforme determinado na configuração do estágio.

CodePipeline fornece suporte para seis tipos de ações:

- Fonte
- Criar
- Testar
- Implantar
- Aprovação
- Invocar

Para obter informações sobre os produtos AWS service (Serviço da AWS) e serviços de parceiros que você pode integrar ao seu pipeline com base no tipo de ação, consulteIntegrações com tipos de CodePipeline ação.

Tópicos

- Como trabalhar com tipos de ação
- Crie e adicione uma ação personalizada no CodePipeline
- Marque uma ação personalizada em CodePipeline
- Invocar um AWS Lambda função em um pipeline em CodePipeline
- Adicionar uma ação de aprovação manual a uma fase
- Adicione uma ação entre regiões em CodePipeline
- Trabalhar com variáveis

Como trabalhar com tipos de ação

Os tipos de ação são ações pré-configuradas que você, como provedor, cria para os clientes usando um dos modelos de integração compatíveis com o AWS CodePipeline.

Você pode solicitar, visualizar e atualizar os tipos de ação. Se o tipo de ação for criado para sua conta como proprietário, você poderá usar o AWS CLI para visualizar ou atualizar as propriedades e a estrutura do tipo de ação. Se você for o provedor ou proprietário do tipo de ação, seus clientes poderão escolher a ação e adicioná-la aos funis depois que ela estiver disponível. CodePipeline

> **Note**
>
> Você cria ações `custom` no campo `owner` para serem executadas com um operador de trabalho. Você não as cria com um modelo de integração. Para obter informações sobre as ações personalizadas, consulte Crie e adicione uma ação personalizada no CodePipeline.

Componentes do tipo de ação

Os componentes a seguir formam um tipo de ação.

- ID do tipo de ação: o ID consiste na categoria, proprietário, provedor e versão. O exemplo a seguir mostra um ID de tipo de ação com um proprietário `ThirdParty`, uma categoria `Test`, um provedor chamado `TestProvider` e a versão `1`.

```
{
    "Category": "Test",
    "Owner": "ThirdParty",
    "Provider": "TestProvider",
    "Version": "1"
},
```

- Configuração do executor: o modelo de integração, ou mecanismo de ação, especificado quando a ação é criada. Ao especificar o executor para um tipo de ação, você escolhe um dos dois tipos:
  - Lambda: o proprietário do tipo de ação grava a integração como uma função Lambda, que é invocada CodePipeline sempre que há um trabalho disponível para a ação.
  - JobWorker: o proprietário do tipo de ação escreve a integração como um funcionário que pesquisa as vagas disponíveis nos canais de clientes. Em seguida, o funcionário executa o trabalho e envia o resultado do trabalho de volta CodePipeline usando CodePipeline APIs.

> **Note**
>
> O modelo de integração do operador de trabalho não é o modelo de integração preferido.

- Artefatos de entrada e saída: limites para os artefatos que o proprietário do tipo de ação designa para os clientes da ação.
- Permissões: a estratégia de permissões que designa os clientes que podem acessar o tipo de ação de terceiros. As estratégias de permissões disponíveis dependem do modelo de integração escolhido para o tipo de ação.
- URLs: links profundos para recursos com os quais o cliente pode interagir, como a página de configuração do proprietário do tipo de ação.

Tópicos

- Solicitar um tipo de ação
- Adicionar um tipo de ação disponível a um pipeline (console)
- Visualizar um tipo de ação
- Atualizar um tipo de ação

Solicitar um tipo de ação

Quando um novo tipo de CodePipeline ação é solicitado por um provedor terceirizado, o tipo de ação é criado para o proprietário do tipo de ação em CodePipeline, e o proprietário pode gerenciar e visualizar o tipo de ação.

Um tipo de ação pode ser uma ação privada ou pública. Quando seu tipo de ação é criado, ele é privado. Para solicitar que um tipo de ação seja alterado para uma ação pública, entre em contato com a equipe CodePipeline de atendimento.

Antes de criar seu arquivo de definição de ação, recursos do executor e solicitação de tipo de ação para a CodePipeline equipe, você deve escolher um modelo de integração.

Etapa 1: Escolher seu modelo de integração

Escolha seu modelo de integração e, em seguida, crie a configuração para esse modelo. Após escolher o modelo de integração, você deve configurar seus recursos de integração.

- Para o modelo de integração do Lambda, crie uma função do Lambda e adicione permissões. Adicione permissões à função Lambda do seu integrador para fornecer ao CodePipeline serviço permissões para invocá-lo usando o principal do CodePipeline serviço:. `codepipeline.amazonaws.com` As permissões podem ser adicionadas usando CloudFormation ou a linha de comando.
  - Exemplo de adição de permissões usando o 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
```

  - Documentação para linha de comando
- Para o modelo de integração do funcionário, você cria uma integração com uma lista de contas permitidas em que o funcionário pesquisa vagas com as CodePipeline APIs.

Etapa 2: Criar um arquivo de definição de tipo de ação

Você define um tipo de ação em um arquivo de definição de tipo de ação usando a linguagem JSON. No arquivo, você inclui a categoria da ação, o modelo de integração usado para gerenciar o tipo de ação e as propriedades de configuração.

> **Note**
>
> Após criar uma ação pública, você não pode alterar a propriedade do tipo de ação em `properties` de `optional` para`required`. Você não pode alterar o `owner`.

Para obter mais informações sobre os parâmetros do arquivo de definição do tipo de ação, consulte ActionTypeDeclaration e UpdateActionType na Referência da CodePipeline API.

Há oito seções no arquivo de definição de tipo de ação:

- `description`: a descrição do tipo de ação a ser atualizado.
- `executor`: informações sobre o executor de um tipo de ação criado com um modelo de integração compatível, `Lambda` ou `job worker`. Você só pode fornecer um `jobWorkerExecutorConfiguration` ou `lambdaExecutorConfiguration`, com base no seu tipo de executor.
  - `configuration`: recursos para a configuração do tipo de ação, com base no modelo de integração escolhido. No modelo de integração do Lambda, use o ARN da função do Lambda. No modelo de integração do operador de trabalho, use a conta ou a lista de contas na qual o operador de trabalho atua.
  - `jobTimeout`: o tempo limite, em segundos, do trabalho. A execução de uma ação pode consistir em vários trabalhos. Esse é o tempo limite de um único trabalho, e não de toda a execução da ação.

    > **Note**
    >
    > No modelo de integração Lambda, o tempo limite máximo é 15 minutos.

  - `policyStatementsTemplate`: a declaração de política que especifica as permissões na conta do CodePipeline cliente que são necessárias para executar com êxito uma ação.
  - `type`: o modelo de integração usado para criar e atualizar o tipo de ação, `Lambda` ou `JobWorker`.
- `id`: o ID da categoria, do proprietário, do provedor e da versão para o tipo de ação:
  - `category`: O tipo de ação pode ser executado no estágio: Origem, Compilar, Implantar, Testar, Invocar ou Aprovar.
  - `provider`: o provedor do tipo de ação que está sendo chamado, por exemplo, a empresa fornecedora ou o nome do produto. O nome do provedor é fornecido quando o tipo de ação é criado.
  - `owner`: O criador do tipo de ação que está sendo chamado: `AWS` ou `ThirdParty`.
  - `version`: uma string usada para criar a versão do tipo de ação. Para a primeira versão, defina o número da versão como 1.
- `inputArtifactDetails`: o número de artefatos esperados do estágio anterior do pipeline.
- `outputArtifactDetails`: o número de artefatos esperados do resultado do estágio do tipo de ação.

- `permissions`: detalhes que identificam as contas com permissões para usar o tipo de ação.
- `properties`: os parâmetros necessários para que as tarefas do seu projeto sejam concluídas.
  - `description`: a descrição da propriedade exibida para os usuários.
  - `optional`: se a propriedade da configuração é opcional.
  - `noEcho`: se o valor do campo inserido pelo cliente foi omitido do log. Se `true` sim, o valor será editado quando retornado com uma solicitação de GetPipeline API.
  - `key`: se a propriedade da configuração é uma chave.
  - `queryable`: Se a propriedade é usada com pesquisas. Um tipo de ação pode ter até uma propriedade consultável. Se tiver uma, essa propriedade deve ser obrigatória e não secreta.
  - `name`: o nome de propriedade que é exibido para os usuários.
- `urls`: Uma lista dos URLs é CodePipeline exibida para seus usuários.
  - `entityUrlTemplate`: URL para os recursos externos do tipo de ação, como uma página de configuração.
  - `executionUrlTemplate`: URL para os detalhes da última execução da ação.
  - `revisionUrlTemplate`: URL exibida no CodePipeline console para a página em que os clientes podem atualizar ou alterar a configuração da ação externa.
  - `thirdPartyConfigurationUrl`: URL de uma página na qual os usuários podem se cadastrar em um serviço externo e executar a configuração inicial da ação fornecida por esse serviço.

O código a seguir mostra um exemplo de arquivo de definição de tipo de ação.

```
{
   "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"
      }
   }
}
```

Etapa 3: registre sua integração com CodePipeline

Para registrar seu tipo de ação com CodePipeline, entre em contato com a equipe CodePipeline de atendimento com sua solicitação.

A equipe CodePipeline de serviço registra a nova integração do tipo de ação fazendo alterações na base de código do serviço. CodePipeline registra duas novas ações: uma ação pública e uma ação privada. Você usa a ação privada para testar e, quando está pronto, ativa a ação pública para atender ao tráfego de clientes.

Para registrar uma solicitação de integração do Lambda

- Envie uma solicitação para a equipe CodePipeline de atendimento usando o formulário a seguir.

```
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 Regiões da AWS where your action will be available

5. Will this be available as a public action?
```

Para registrar uma solicitação de integração do operador de trabalho

- Envie uma solicitação para a equipe CodePipeline de atendimento usando o formulário a seguir.

```
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 Regiões da AWS where your action will be available

5. Will this be available as a public action?
```

Etapa 4: Ativar a nova integração

Entre em contato com a equipe de CodePipeline serviço quando estiver pronto para usar a nova integração publicamente.

Adicionar um tipo de ação disponível a um pipeline (console)

Você adiciona seu tipo de ação a um pipeline para testá-lo. Você pode fazer isso criando um novo pipeline ou editando um existente.

> **Note**
>
> Se seu tipo de ação for uma ação de categoria de origem, criação ou implantação, você poderá adicioná-la criando um pipeline. Se seu tipo de ação estiver na categoria de teste, você deve adicioná-la ao editar um pipeline existente.

Para adicionar seu tipo de ação a um pipeline existente a partir do CodePipeline console

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Na lista de pipelines, escolha o pipeline em que deseja adicionar o tipo de ação.
3. Na página de visualização do resumo do pipeline, escolha Editar.
4. Opte por editar o estágio. Na etapa em que você deseja adicionar seu tipo de ação, escolha Adicionar grupo de ações. A página Editar ação é exibida.
5. Na página Editar ação, em Nome da ação, insira um nome para a ação. Esse será o nome exibido para o estágio no seu pipeline.
6. Em Provedor de ação, selecione seu tipo de ação na lista.

   Observe que o valor na lista baseia-se no `provider` especificado no arquivo de definição de tipo de ação.
7. Em Artefatos de entrada, insira o nome do artefato neste formato:

   `Artifactname``::`*`FileName`*

   Observe que as quantidades mínima e máxima permitidas são definidas com base no `inputArtifactDetails` especificado no arquivo de definição de tipo de ação.
8. Escolha Conectar a < Action_Name >.

   Uma janela de navegador é aberta e se conecta ao site que você criou para seu tipo de ação.
9. Faça login no site como cliente e conclua as etapas que um cliente deve executar para usar seu tipo de ação. Suas etapas variam dependendo da categoria da ação, do site e da configuração, mas geralmente incluem uma ação de conclusão que retorna o cliente à página Editar ação.
10. Na página CodePipeline Editar ação, os campos de configuração adicionais da ação são exibidos. Os campos exibidos são as propriedades de configuração que você especificou no arquivo de definição de ação. Insira as informações nos campos que são personalizados para o seu tipo de ação.

    Por exemplo, se o arquivo de definição de ação especificou uma propriedade chamada `Host`, um campo com o rótulo Host será mostrado na página Editar ação da sua ação.
11. Em Artefatos de saída, insira o nome do artefato neste formato:

    `Artifactname``::`*`FileName`*

Observe que as quantidades mínima e máxima permitidas são definidas com base no `outputArtifactDetails` especificado no arquivo de definição de tipo de ação.

12. Escolha Concluído para retornar à página de detalhes do pipeline.

> **Note**
>
> Se desejarem, seus clientes poderão usar a CLI para adicionar o tipo de ação ao pipeline.

13. Para testar sua ação, confirme uma alteração na origem especificada no estágio de origem do pipeline ou siga as etapas em Iniciar manualmente um pipeline.

Para criar um pipeline com seu tipo de ação, siga as etapas em Criar um pipeline, estágios e ações e escolha o tipo de ação em quantas etapas desejar testar.

Visualizar um tipo de ação

É possível usar a CLI para visualizar seu tipo de ação. Use o comando get-action-type para visualizar os tipos de ação que foram criados por meio de um modelo de integração.

Visualizar um tipo de ação

1. Crie um arquivo JSON de entrada e nomeie o arquivo como `file.json`. Adicione o ID do seu tipo de ação no formato JSON como mostrado no exemplo a seguir.

```
{
    "category": "Test",
    "owner": "ThirdParty",
    "provider": "TestProvider",
    "version": "1"
}
```

2. Em uma janela de terminal ou na linha de comando, execute o comando get-action-type.

```
aws codepipeline get-action-type --cli-input-json file://file.json
```

Esse comando retorna a saída da definição de ação para um tipo de ação. Este exemplo mostra um tipo de ação que foi criado com o modelo de integração 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": []
    }
}
```

Atualizar um tipo de ação

Você pode usar a CLI para editar tipos de ação criados com um modelo de integração.

Para um tipo de ação pública, você não pode atualizar o proprietário, não pode alterar propriedades opcionais para obrigatórias e só pode adicionar novas propriedades opcionais.

1. Use o comando `get-action-type` para obter a estrutura do seu tipo de ação. Copie a estrutura.
2. Crie um arquivo JSON e o nomeie como `action.json`. Cole nele a estrutura de tipo de ação que você copiou na etapa anterior. Atualize os parâmetros que deseja atualizar. Você também pode adicionar parâmetros opcionais.

   Para obter mais informações sobre os parâmetros do arquivo de entrada, consulte a descrição do arquivo de definição de ação em Etapa 2: Criar um arquivo de definição de tipo de ação.

   O exemplo a seguir mostra como atualizar um exemplo de tipo de ação criado com o modelo de integração Lambda. Este exemplo efetua as seguintes alterações:

   - Mude o nome do `provider` para `TestProvider1`.
   - Adicione um tempo limite de trabalho de 900 segundos.
   - Adiciona uma propriedade de configuração de ação chamada `Host` que é exibida para o cliente que utiliza a ação.

```
{
    "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. No terminal ou na linha de comando, execute o comando update-action-type.

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

Esse comando retorna a saída do tipo de ação com base nos parâmetros atualizados.

Crie e adicione uma ação personalizada no CodePipeline

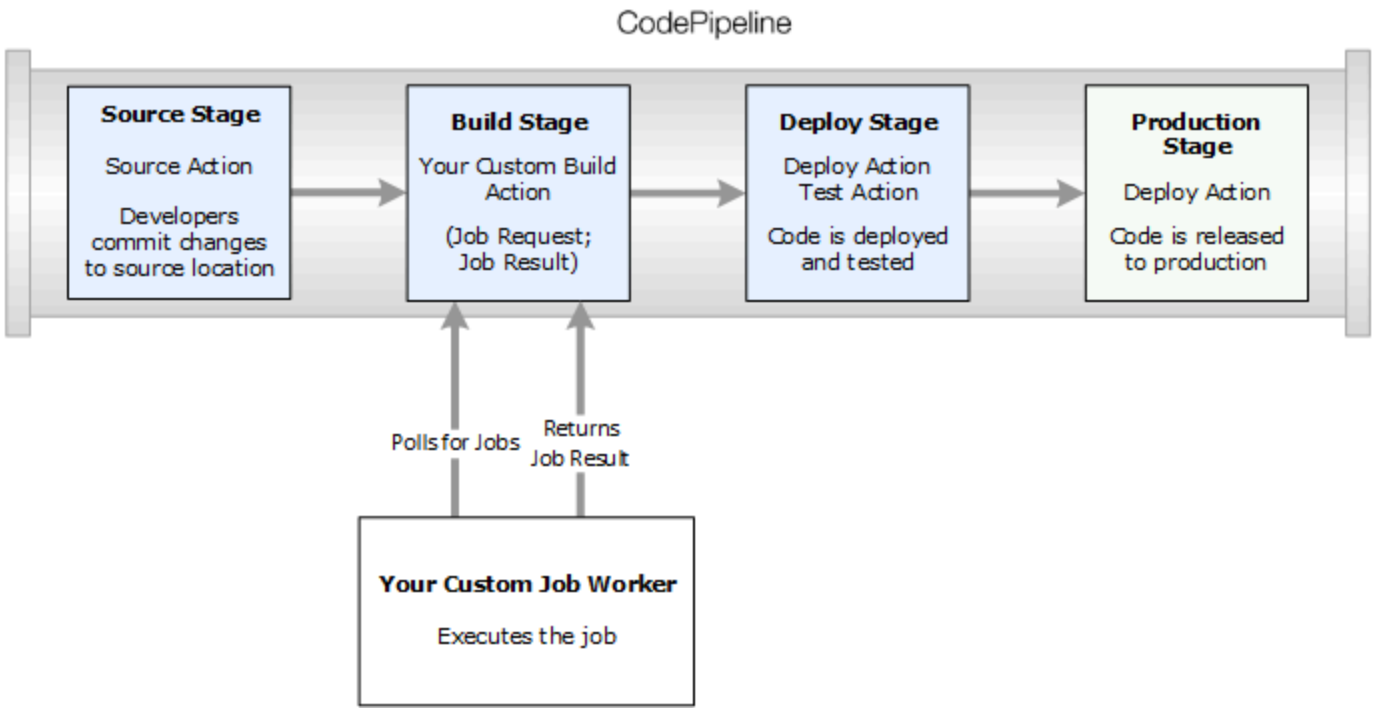
AWS CodePipeline inclui várias ações que ajudam você a configurar a criação, o teste e a implantação de recursos para seu processo de lançamento automatizado. Se o processo de liberação tiver atividades que não estão incluídas nas ações padrão, como um processo de compilação desenvolvido internamente ou um conjunto de testes, você pode criar uma ação personalizada para essa finalidade e incluí-la no pipeline. Você pode usar o AWS CLI para criar ações personalizadas em funis associados à sua AWS conta.

Você pode criar ações personalizadas para as seguintes categorias de AWS CodePipeline ação:

- Uma ação de compilação personalizada que cria ou transforma itens
- Uma ação de implantação personalizada que implanta itens em um ou mais servidores, sites ou repositórios
- Uma ação de teste personalizada que configura e executa testes automatizados
- Uma ação de invocação personalizada que executa funções

Ao criar uma ação personalizada, você também deve criar um funcionário que pesquisará as solicitações de trabalho CodePipeline para essa ação personalizada, executará o trabalho e retornará o resultado do status para CodePipeline. Esse operador de trabalho pode estar localizado em qualquer computador ou recurso, desde que tenha acesso ao endpoint público do CodePipeline. Para gerenciar facilmente o acesso e a segurança, é recomendável hospedar o operador de trabalho em uma instância do Amazon EC2.

O diagrama a seguir mostra uma exibição de alto nível de um pipeline que inclui uma ação personalizada de compilação:



Quando um pipeline inclui uma ação personalizada como parte de um estágio, ele criará uma solicitação de trabalho. Um operador de trabalho personalizado detecta essa solicitação e executa o trabalho (neste exemplo, em um processo personalizado usando o software de compilação de terceiros). Quando a ação for concluída, o operador de trabalho retornará um resultado de êxito ou de falha. Se um resultado de êxito for recebido, o pipeline fornecerá a revisão e seus artefatos na próxima ação. Se uma falha for retornada, o pipeline não fornecerá a revisão na próxima ação no pipeline.

> **Note**
>
> Estas instruções supõem que você já tenha concluído as etapas em Começando com CodePipeline.

Tópicos

- Criar uma ação personalizada
- Criar um operador de trabalho para a ação personalizada
- Adicionar uma ação personalizada a um pipeline

Criar uma ação personalizada

Para criar uma ação personalizada com o AWS CLI

1. Abra um editor de texto e crie um arquivo JSON para sua ação personalizada que inclua a categoria da ação, o provedor da ação e as configurações necessárias para sua ação personalizada. Por exemplo, para criar uma ação de construção personalizada que requeira apenas uma propriedade, seu arquivo JSON pode ter a seguinte aparência:

```
{
    "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"
    }],
```

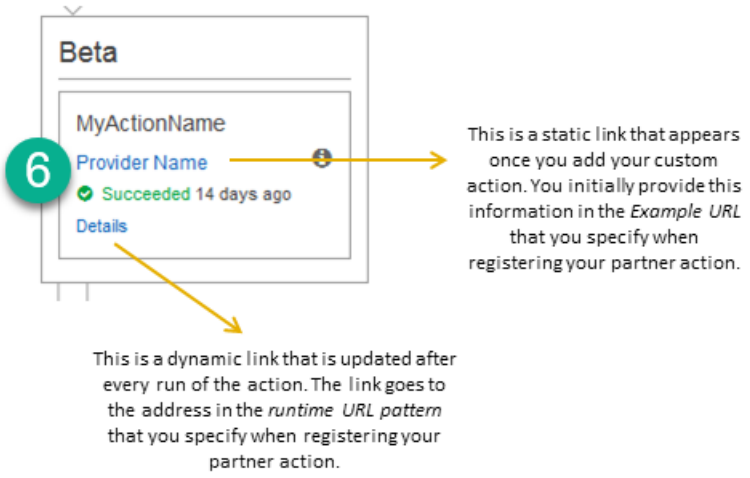
```
    "inputArtifactDetails": {
        "maximumCount": integer,
        "minimumCount": integer
    },
    "outputArtifactDetails": {
        "maximumCount": integer,
        "minimumCount": integer
    },
    "tags": [{
      "key": "Project",
      "value": "ProjectA"
    }]
}
```

Este exemplo adiciona tags à ação personalizada por meio da inclusão da chave de tag `Project` e o valor `ProjectA` na ação personalizada. Para obter mais informações sobre como marcar recursos em CodePipeline, consulteMarcando atributos.

Há duas propriedades incluídas no arquivo JSON, `entityUrlTemplate` e `executionUrlTemplate`. Você pode consultar um nome nas propriedades de configuração da ação personalizada dentro dos modelos de URL, seguindo o formato `{Config:`*`name`*`}`, desde que a propriedade de configuração seja necessária e não secreta. Por exemplo, no exemplo acima, o `entityUrlTemplate` valor se refere à propriedade de configuração*`ProjectName`*.

- `entityUrlTemplate`: o link estático que fornece informações sobre o provedor de serviços para a ação. No exemplo, o sistema de construção inclui um link estático para cada projeto de construção. O formato do link poderá variar, dependendo do seu provedor de construção (ou, se estiver criando um outro tipo de ação, como teste, outro provedor de serviços). Você deve fornecer este formato de link de forma que quando a ação personalizada for adicionada, o usuário possa escolher este link para abrir um navegador em uma página de seu site que forneça informações específicas para o projeto de criação (ou ambiente de teste).
- `executionUrlTemplate`: o link dinâmico que será atualizado com informações sobre a execução atual ou mais recente da ação. Quando seu operador de trabalho personalizado atualiza o status de uma tarefa (por exemplo, sucesso, falha ou em andamento), ele também fornece uma `externalExecutionId` que será usada para completar o link. Este link pode ser usado para fornecer detalhes sobre a execução de uma ação.

Por exemplo, quando você visualizar a ação no pipeline, você verá os dois links a seguir:



1

Este link estático é exibido depois que você adicionar sua ação personalizada e aponta para o endereço no `entityUrlTemplate`, que você especifica ao criar sua ação personalizada.

2

Este link dinâmico é atualizado após cada execução da ação e aponta para o endereço no `executionUrlTemplate`, que você especifica ao criar sua ação personalizada.

Para obter mais informações sobre esses tipos de link, bem como sobre `RevisionURLTemplate` e `ThirdPartyURL`, consulte ActionTypeSettings e CreateCustomActionType na Referência da CodePipeline API. Para mais informações sobre os requisitos de estrutura de ação e como criar uma ação, consulte CodePipeline referência de estrutura de tubulação.

2. Salve o arquivo JSON e dê a ele um nome que você possa lembrar facilmente (por exemplo, *`MyCustomAction`* .json).
3. Abra uma sessão de terminal (Linux, OS X, Unix) ou prompt de comando (Windows) em um computador onde você tenha instalado o AWS CLI.
4. Use o AWS CLI para executar o aws codepipeline create-custom-action-type comando, especificando o nome do arquivo JSON que você acabou de criar.

   Por exemplo, para criar uma ação personalizada de compilação:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

```
aws codepipeline create-custom-action-type --cli-input-json
 file://MyCustomAction.json
```

5. Este comando retorna a estrutura inteira da ação personalizada que você criou, bem como a propriedade de configuração da ação `JobList`, que é adicionada para você. Quando você adiciona a ação personalizada a um pipeline, você pode usar `JobList` para especificar quais projetos do provedor você pode consultar para tarefas. Se você não configurar isso, todos as tarefas disponíveis serão retornadas quando o operador do trabalho personalizado efetuar uma consulta por tarefas.

   Por exemplo, o comando anterior pode retornar uma estrutura similar ao seguinte:

```
{
    "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 da saída do create-custom-action-type comando, a `id` seção inclui`"owner": "Custom"`. CodePipeline atribui automaticamente `Custom` como proprietário dos tipos de ação personalizados. Esse valor não pode ser atribuído ou alterado quando você usar o comando create-custom-action-type ou o comando update-pipeline.

Criar um operador de trabalho para a ação personalizada

As ações personalizadas exigem que um funcionário faça uma pesquisa de solicitações de trabalho CodePipeline para a ação personalizada, execute o trabalho e retorne o resultado do status para CodePipeline. O operador de trabalho pode estar localizado em qualquer computador ou recurso desde que tenha acesso ao endpoint público do CodePipeline.

Há várias maneiras de criar o operador de trabalho. As seções a seguir fornecem orientações práticas para o desenvolvimento de seu operador de trabalho personalizado para o CodePipeline.

Tópicos

- Escolher e configurar uma estratégia de gerenciamento de permissões para o operador de trabalho
- Desenvolver um operador de trabalho para a ação personalizada
- Arquitetura e exemplos de operadores de trabalho personalizados

Escolher e configurar uma estratégia de gerenciamento de permissões para o operador de trabalho

Para desenvolver um funcionário de trabalho personalizado para sua ação personalizada em CodePipeline, você precisará de uma estratégia para a integração do gerenciamento de usuários e permissões.

A estratégia mais simples é adicionar a infraestrutura necessária para seu operador de trabalho personalizado criando instâncias do Amazon EC2 com um perfil de instância do IAM, que permite aumentar verticalmente a escala dos recursos necessários à integração com facilidade. Você pode usar a integração integrada com AWS para simplificar a interação entre seu funcionário personalizado CodePipeline e.

Para configurar as instâncias do Amazon EC2

1. Saiba mais sobre o Amazon EC2 e determine se esta é a escolha certa para sua integração. Para obter mais informações, consulte Amazon EC2 - Hospedagem de servidor virtual.
2. Comece a criar suas instâncias do Amazon EC2. Para obter informações, consulte Conceitos básicos das instâncias do Amazon EC2 Linux.

Outra estratégia a ser considerada é usar a federação de identidades com o IAM para integrar seu sistema e recursos existentes do provedor de identidades. Essa estratégia é especialmente útil se você já tiver um provedor de identidades corporativas ou já estiver configurado para oferecer suporte aos usuários usando provedores de identidades da web. A federação de identidades permite que você conceda acesso seguro aos AWS recursos CodePipeline, inclusive, sem precisar criar ou gerenciar usuários do IAM. É possível usar recursos e políticas para requisitos de segurança de senhas e rotação de credenciais. Você pode usar aplicações de exemplo como modelos para seu próprio design.

Para configurar a federação de identidades

1. Saiba mais sobre a federação de identidades do IAM. Para mais informações, consulte Gerenciar federação.
2. Analise os exemplos em Situações de concessão de acesso temporário para identificar o cenário para acesso temporário que melhor atenda às necessidades de sua ação personalizada.
3. Revise os exemplos de código de federação de identidades relevantes para sua infraestrutura, como:

- Aplicativo de amostra de federação de identidades para um caso de uso do diretório ativo

4. Comece a configuração da federação de identidades. Para obter informações, consulte Provedores e federação de identidades no Guia do usuário do IAM.

Crie uma das opções a seguir para usar abaixo de você Conta da AWS ao executar sua ação personalizada e seu funcionário de trabalho.

Os usuários precisam de acesso programático se quiserem interagir com pessoas AWS fora do Console de gerenciamento da AWS. A forma de conceder acesso programático depende do tipo de usuário que está acessando AWS.

Para conceder acesso programático aos usuários, selecione uma das seguintes opções:

Qual usuário precisa de acesso programático?	Para	Por
IAM	(Recomendado) Use as credenciais do console como credenciais temporárias para assinar solicitações programáticas para os AWS SDKs ou AWS CLI APIs. AWS	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Login para desenvolvimento AWS local no Guia AWS Command Line Interface do usuário. • Para AWS SDKs, consulte Login para desenvolvimento AWS local no Guia de referência de AWS SDKs e ferramentas.
Identidade da força de trabalho (Usuários gerenciados no Centro de Identidade do IAM)	Use credenciais temporárias para assinar solicitações programáticas para os AWS SDKs AWS CLI ou APIs. AWS	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Configurando o AWS CLI para uso Centro de Identidade do AWS IAM no

Qual usuário precisa de acesso programático?	Para	Por
		Guia do AWS Command Line Interface usuário. • Para AWS SDKs, ferramentas e AWS APIs, consulte a autenticação do IAM Identity Center no Guia de referência de AWS SDKs e ferramentas.
IAM	Use credenciais temporárias para assinar solicitações programáticas para os AWS SDKs AWS CLI ou APIs. AWS	Siga as instruções em Como usar credenciais temporárias com AWS recursos no Guia do usuário do IAM.
IAM	(Não recomendado) Use credenciais de longo prazo para assinar solicitações programáticas para os AWS CLI AWS SDKs ou APIs. AWS	Siga as instruções da interface que deseja utilizar. • Para o AWS CLI, consulte Autenticação usando credenciais de usuário do IAM no Guia do AWS Command Line Interface usuário. • Para AWS SDKs e ferramentas, consulte Autenticar usando credenciais de longo prazo no Guia de referência de AWS SDKs e ferramentas. • Para AWS APIs, consulte Gerenciamento de chaves de acesso para usuários do IAM no Guia do usuário do IAM.

Veja a seguir uma política de exemplo que você pode criar para usar com o operador de trabalho personalizado. Essa política serve somente como exemplo e é fornecida no estado em que encontra.

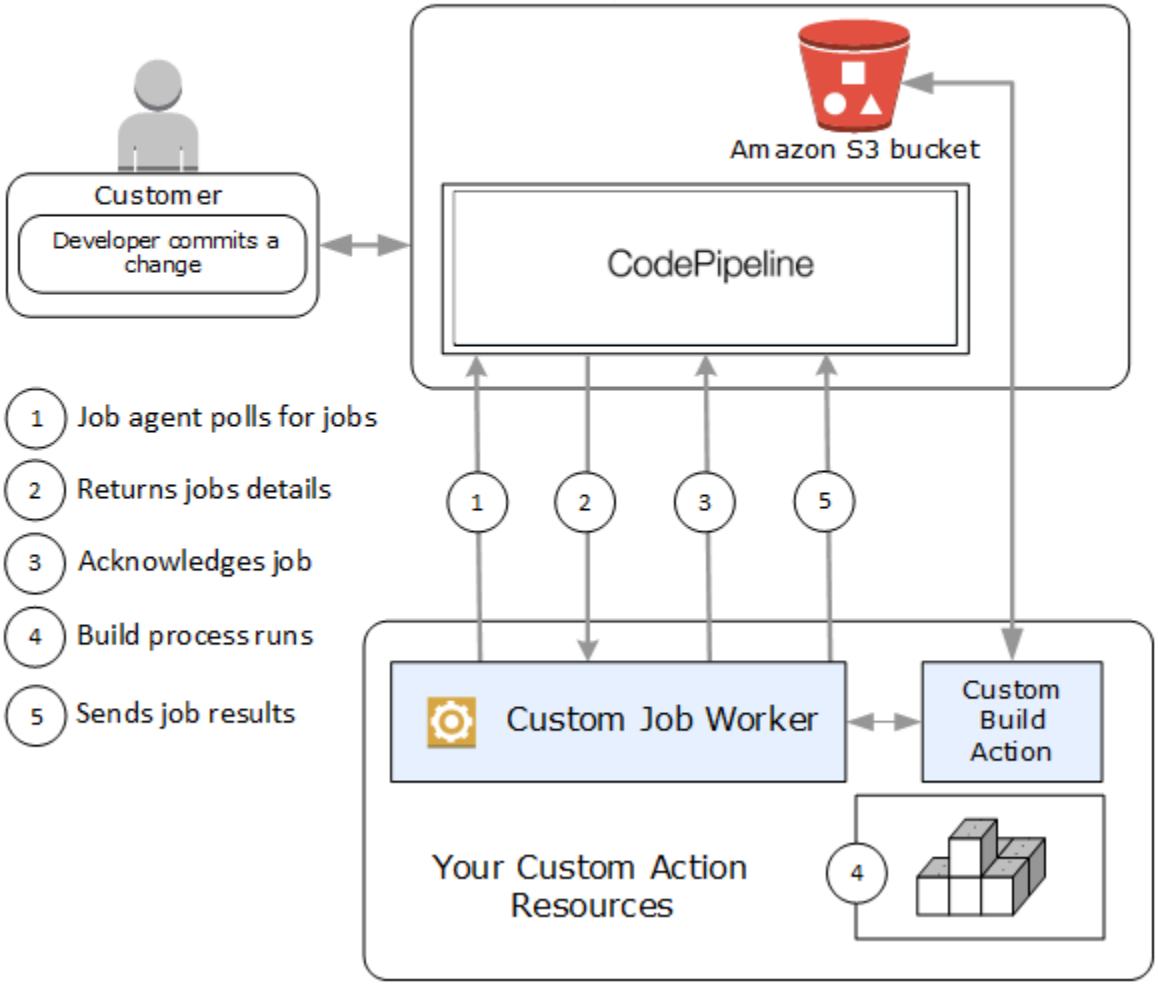
JSON

```
{
  "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**
>
> É recomendável usar a política gerenciada `AWSCodePipelineCustomActionAccess`.

Desenvolver um operador de trabalho para a ação personalizada

Depois de escolher sua estratégia de gerenciamento de permissões, considere como seu funcionário interagirá com ela CodePipeline. O diagrama de alto nível a seguir mostra o fluxo de trabalho de uma ação personalizada e do operador de trabalho em um processo de compilação.



1. Seu trabalhador faz pesquisas de CodePipeline emprego usando`PollForJobs`.
2. Quando um pipeline é acionado por uma alteração em seu estágio de origem (por exemplo, quando um desenvolvedor confirma uma alteração), o processo automatizado de liberação é iniciado. O processo continuará até que o estágio em sua ação personalizada seja configurado. Quando ele atinge sua ação nesse estágio, coloca um trabalho na CodePipeline fila. Esse trabalho aparecerá se o operador de trabalho chamar `PollForJobs` novamente para obter o status. Pegue os detalhes do trabalho e de `PollForJobs` e passe-os de volta ao operador de trabalho.
3. O funcionário liga `AcknowledgeJob` para enviar CodePipeline uma confirmação de emprego. CodePipeline retorna uma confirmação que indica que o funcionário deve continuar o trabalho (`InProgress`) ou, se você tiver mais de um funcionário pesquisando vagas e outro funcionário já tiver reivindicado a vaga, uma resposta de `InvalidNonceException` erro será retornada. Após a `InProgress` confirmação, CodePipeline espera que os resultados sejam retornados.

4. O operador de trabalho inicia sua ação personalizada na revisão e, em seguida, a ação é executada. Juntamente com outras ações, sua ação personalizada retornará um resultado ao operador de trabalho. No exemplo de uma ação de compilação personalizada, a ação efetua o pull de artefatos do bucket do Amazon S3, compila-os e envia por push os artefatos compilados de volta ao bucket do Amazon S3.
5. Quando a ação está em execução, o operador de trabalho pode chamar `PutJobSuccessResult` com um token de continuação (a serialização do estado do trabalho gerado pelo operador de trabalho, por exemplo, um identificador de compilação em formato JSON ou uma chave de objeto do Amazon S3), bem como as informações `ExternalExecutionId` que serão usadas para preencher o link em `executionUrlTemplate`. Isso atualizará a exibição do console do pipeline com um link funcional para os detalhes específicos da ação enquanto ela estiver em andamento. Embora não seja necessário, é uma melhor prática, porque permite que os usuários visualizem o status da ação personalizada durante sua execução.

   Assim que `PutJobSuccessResult` for chamada , o trabalho é considerado como concluído. Um novo trabalho é criado e inclui CodePipeline o token de continuação. Esse trabalho aparecerá se o operador de trabalho chamar `PollForJobs` novamente. Esse novo trabalho pode ser usado para verificar o estado da ação e retornará com um token de continuação ou retornará sem um token de continuação assim que a ação estiver concluída.

   > **Note**
   >
   > Se o operador de trabalho executa todo o trabalho para uma ação personalizada, você deve considerar quebrar o processamento do operador de trabalho em pelo menos em duas etapas. A primeira etapa estabelece a página de detalhes para a ação. Assim que tiver criado a página de detalhes, você pode serializar o estado do operador de trabalho e retorná-lo como um token de continuação, sujeito a limites de tamanho (consulte Cotas em AWS CodePipeline). Por exemplo, você pode gravar o estado da ação na string que usar como o token de continuação. A segunda etapa (e as etapas subsequentes) do processamento do operador de trabalho desempenha o trabalho real da ação. A etapa final retorna o sucesso ou a falha CodePipeline, sem nenhum token de continuação na etapa final.

   Para obter mais informações sobre como usar o token de continuação, consulte as especificações para `PutJobSuccessResult` na Referência de APIs do CodePipeline .

6. Depois que a ação personalizada for concluída, o funcionário retornará o resultado da ação personalizada para CodePipeline chamando uma das duas APIs:
   - `PutJobSuccessResult` sem um token de continuação, indicando que a ação personalizada foi executada com êxito
   - `PutJobFailureResult`, indicando que a ação personalizada não foi executada com êxito

   Dependendo do resultado, o pipeline prosseguirá para a ação seguinte (êxito) ou parará (falha).

Arquitetura e exemplos de operadores de trabalho personalizados

Depois que tiver mapeado seu fluxo de trabalho de alto nível, você pode criar o operador de trabalho. Embora as especificidades de sua ação personalizada basicamente determinem o que é necessário para seu operador de trabalho, a maioria dos operadores de trabalho para ações personalizadas incluem a seguinte funcionalidade:

- Pesquisa de empregos a partir do CodePipeline usoPollForJobs.
- Reconhecendo trabalhos e retornando resultados ao CodePipeline usar AcknowledgeJobPutJobSuccessResult, e. PutJobFailureResult
- Recuperação de artefatos ao and/or colocar artefatos no bucket do Amazon S3 para o pipeline. Para fazer download dos artefatos no bucket do Amazon S3, é necessário criar um cliente do Amazon S3 que use a assinatura do Signature versão 4 (Sig V4). O Sig V4 é necessário para. AWS KMS

  Para fazer upload dos artefatos no bucket do Amazon S3, você deve configurar a solicitação do Amazon S3 PutObject para usar criptografia. Atualmente, somente o AWS Key Management Service (AWS KMS) é suportado para criptografia. AWS KMS usa AWS KMS keys. Para saber se deve usar uma chave gerenciada pelo cliente Chave gerenciada pela AWS ou uma chave gerenciada pelo cliente para fazer upload de artefatos, seu funcionário personalizado deve examinar os dados do trabalho e verificar a propriedade da chave de criptografia. Se a propriedade estiver definida, você deverá usar esse ID de chave gerenciado pelo cliente ao configurar. AWS KMS Se a propriedade da chave for nula, você usa o. Chave gerenciada pela AWS CodePipeline usa o, a Chave gerenciada pela AWS menos que seja configurado de outra forma.

  Para ver um exemplo que mostra como criar os AWS KMS parâmetros em Java ou.NET, consulte Especificação do AWS Key Management Service no Amazon S3 usando os AWS SDKs. Para

obter mais informações sobre o bucket Amazon S3 para CodePipeline, consulteCodePipeline conceitos.

Um exemplo mais complexo de funcionário personalizado está disponível em GitHub. Esse exemplo é de código aberto e fornecido no estado que se encontra.

- Sample Job Worker para CodePipeline: Baixe a amostra do GitHub repositório.

Adicionar uma ação personalizada a um pipeline

Depois de contratar um funcionário, você pode adicionar sua ação personalizada a um pipeline criando uma nova e escolhendo-a ao usar o assistente Create Pipeline, editando um pipeline existente e adicionando a ação personalizada ou usando os AWS CLI SDKs ou as APIs.

> **Note**
>
> Você pode criar um pipeline no assistente de criação de pipeline que inclua uma ação personalizada se for uma ação de criação ou implantação. Se a ação personalizada estiver na categoria de teste , você deve adicioná-la ao editar um pipeline existente.

Tópicos

- Adicionar uma ação personalizada a um pipeline existente (CLI)

Adicionar uma ação personalizada a um pipeline existente (CLI)

Você pode usar o AWS CLI para adicionar uma ação personalizada a um funil existente.

1. Abra uma sessão de terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e execute o comando get-pipeline para copiar a estrutura do pipeline a ser editada para um arquivo JSON. Por exemplo, para um pipeline chamado **`MyFirstPipeline`**, você deve digitar o seguinte comando:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Este comando retorna nada, mas o arquivo que você criou deve aparecer no diretório onde você executou o comando.

2. Abra o arquivo JSON em qualquer editor de texto e modifique a estrutura do arquivo para adicionar sua ação personalizada a um estágio existente.

> **Note**
>
> Se você deseja que a sua ação seja executada em paralelo com outra ação naquele estágio, certifique-se de atribuir-lhe o mesmo valor `runOrder` da ação.

Por exemplo, para modificar a estrutura de um pipeline para adicionar um estágio chamado construção e adicionar uma ação de construção personalizada para esse estágio, você deve modificar o JSON para adicionar o estágio Construção antes de um estágio de implantação, conforme a seguir:

```
,
    {
      "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 suas alterações, execute o comando update-pipeline, especificando o pipeline do arquivo JSON, de modo semelhante ao seguinte:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

Este comando retorna toda a estrutura do pipeline editado.

4. Abra o CodePipeline console e escolha o nome do pipeline que você acabou de editar.

   O pipeline exibirá suas alterações. Na próxima vez que você alterar a localização de origem, o pipeline executará a revisão através da estrutura revisada do pipeline.

Marque uma ação personalizada em CodePipeline

As tags são pares de valores-chave associados AWS aos recursos. Você pode usar o console ou a CLI para aplicar tags às suas ações personalizadas no CodePipeline. Para obter informações sobre marcação de CodePipeline recursos, casos de uso, restrições de chave e valor da tag e tipos de recursos compatíveis, consulte. Marcando atributos

Você pode adicionar, remover e atualizar os valores de tags em uma ação personalizada. Você pode adicionar até 50 tags para cada ação personalizada.

Tópicos

- Adicionar tags a uma ação personalizada
- Visualizar tags para uma ação personalizada
- Editar tags para uma ação personalizada
- Remover tags de uma ação personalizada

Adicionar tags a uma ação personalizada

Siga estas etapas para usar o AWS CLI para adicionar uma tag a uma ação personalizada. Para adicionar uma tag a uma ação personalizada ao criá-la, consulte Crie e adicione uma ação personalizada no CodePipeline.

Nestas etapas, partimos do princípio de que você já instalou uma versão recente da AWS CLI ou atualizou para a versão atual. Para obter mais informações, consulte Instalar a AWS Command Line Interface.

No terminal ou na linha de comando, execute o comando tag-resource, especificando o nome de recurso da Amazon (ARN) da ação personalizada na qual você deseja adicionar tags e a chave e o valor da tag que você deseja adicionar. Você pode adicionar mais de uma tag a uma ação personalizada. Por exemplo, para marcar uma ação personalizada com duas tags, uma chave de tag nomeada *`TestActionType`* com o valor da tag de *`UnitTest`* e uma chave de tag nomeada *`ApplicationName`* com o valor da tag 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
```

Se houver êxito, o comando não retornará nada.

Visualizar tags para uma ação personalizada

Siga estas etapas para usar o AWS CLI para visualizar as AWS tags de uma ação personalizada. Se não foram adicionadas tags, a lista retornará vazia.

No terminal ou na linha de comando, execute o comando list-tags-for-resource. Por exemplo, para visualizar uma lista de chaves de tag e valores de tag para uma ação personalizada com o 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
```

Se houver êxito, o comando retornará informações semelhantes às seguintes:

```
{
    "tags": {
        "TestActionType": "UnitTest",
        "ApplicationName": "MyApplication"
    }
}
```

Editar tags para uma ação personalizada

Siga estas etapas para usar o AWS CLI para editar uma tag para uma ação personalizada. Você pode alterar o valor para uma chave existente ou adicionar outra chave. Você também pode remover tags de uma ação personalizada, como mostrado na próxima seção.

No terminal ou na linha de comando, execute o comando tag-resource, especificando o nome de recurso da Amazon (ARN) da ação personalizada onde você deseja atualizar uma tag e especifique a chave e o valor da tag:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:actiontype:Owner/Category/Provider/Version --tags
 key=TestActionType,value=IntegrationTest
```

Remover tags de uma ação personalizada

Siga estas etapas para usar o AWS CLI para remover uma tag de uma ação personalizada. Ao remover tags do recurso associado, as tags são excluídas.

> **Note**
>
> Se você excluir uma ação personalizada, todas as associações de tag são removidas da ação personalizada excluída. Você não precisa remover tags antes de excluir uma ação personalizada.

No terminal ou na linha de comando, execute o comando untag-resource, especificando o ARN da ação personalizada da qual deseja remover tags e a chave da tag que deseja remover. Por exemplo, para remover uma tag em uma ação personalizada com a chave de tag*TestActionType*:

```
aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-
id:actiontype:Owner/Category/Provider/Version --tag-keys TestActionType
```

Se houver êxito, o comando não retornará nada. Para verificar as tags associadas com a ação personalizada, execute o comando list-tags-for-resource.

Invocar um AWS Lambda função em um pipeline em CodePipeline

O AWS Lambda é um serviço de computação que permite executar código sem o provisionamento ou gerenciamento de servidores. É possível criar funções do Lambda e adicioná-las como ações em seus pipelines. Como o Lambda permite que você grave funções para executar praticamente qualquer tarefa, é possível personalizar a maneira como seu pipeline funciona.

> **Important**
>
> Não registre o evento JSON que é CodePipeline enviado ao Lambda, pois isso pode resultar no registro das credenciais do usuário nos registros. CloudWatch A CodePipeline função

usa um evento JSON para passar credenciais temporárias para o Lambda no campo. `artifactCredentials` Para obter um evento de exemplo, consulte Exemplo de evento JSON.

Veja a seguir como as funções do Lambda podem ser usadas nos pipelines:

- Para criar recursos sob demanda em um estágio de um funil usando CloudFormation e excluí-los em outro estágio.
- Implantar versões de aplicativos sem tempo de inatividade AWS Elastic Beanstalk com uma função do Lambda que troca valores de CNAME.
- Para implantar instâncias do Docker no Amazon ECS.
- Para fazer o backup de recursos antes da compilação ou implantação criando um snapshot do AMI.
- Para adicionar integração de produtos de terceiros a seu pipeline, como postar mensagens para um cliente IRC.

> **Note**
>
> Criar e executar funções do Lambda pode resultar em cobranças em sua AWS conta. Para obter mais informações, consulte Preço.

Este tópico pressupõe que você esteja familiarizado AWS CodePipeline AWS Lambda e saiba como criar pipelines, funções e as políticas e funções do IAM das quais elas dependem. Este tópico mostra como:

- Criar uma função do Lambda que teste se um página da web foi implantada com êxito.
- Configure as funções de execução CodePipeline e do Lambda e as permissões necessárias para executar a função como parte do pipeline.
- Editar um pipeline para adicionar a função do Lambda como uma ação.
- Testa a ação liberando uma alteração manualmente.

> **Note**
>
> Ao usar a ação de invocação do Lambda entre regiões em CodePipeline, o status da execução do Lambda usando o PutJobSuccessResult e PutJobFailureResult deve ser enviado para a AWS região em que a função Lambda está presente e não para a região em que existe. CodePipeline

Este tópico inclui exemplos de funções para demonstrar a flexibilidade de trabalhar com funções do Lambda em: CodePipeline

- Basic Lambda function
  - Criação de uma função básica do Lambda para usar. CodePipeline
  - Retornar resultados de sucesso ou falha para o link Detalhes da ação. CodePipeline
- Exemplo de função Python que usa um AWS CloudFormation modelo
  - Usando parâmetros JSON-encoded do usuário para passar vários valores de configuração para a função (`get_user_params`).
  - Interação com artefatos .zip em um bucket de artefatos (`get_template`).
  - Uso de um token de continuação para monitorar um longo processo assíncrono (`continue_job_later`). Isso permite que a ação continue e a função tenha êxito mesmo se exceder um runtime de quinze minutos (um limite no Lambda).

Cada função de exemplo inclui informações sobre as permissões que você deve adicionar à função. Para obter informações sobre limites em AWS Lambda, consulte Limites no Guia do AWS Lambda desenvolvedor.

> **Important**
>
> O código de exemplo, as funções, e as políticas incluídos neste tópico servem somente como exemplos e são fornecidos no estado em que encontram.

Tópicos

- Etapa 1: Criar um pipeline
- Etapa 2: Criar a função do Lambda

- Etapa 3: adicionar a função Lambda a um pipeline no console CodePipeline
- Etapa 4: Testar o pipeline com a função do Lambda
- Etapa 5: Próximas etapas
- Exemplo de evento JSON
- Funções de exemplo adicionais

Etapa 1: Criar um pipeline

Nesta etapa, você cria um pipeline ao qual adicionará a função do Lambda posteriormente. Este é o mesmo pipeline que você criou em CodePipeline tutoriais. Se esse pipeline ainda estiver configurado para sua conta e estiver na mesma região em que você planeja criar a função do Lambda, será possível ignorar esta etapa.

Para criar o pipeline

1. Siga as três primeiras etapas Tutorial: Criar um pipeline simples (bucket do S3) para criar um bucket do Amazon S3, CodeDeploy recursos e um pipeline de dois estágios. Escolha a opção Amazon Linux como tipos de instância. Você pode usar o nome que quiser para o pipeline, mas as etapas neste tópico utilizam MyLambdaTestPipeline.
2. Na página de status do seu funil, na CodeDeploy ação, escolha Detalhes. Na página de detalhes de implantação para o grupo de implantação, escolha uma ID de instância a partir da lista.
3. No console do Amazon EC2, na guia Detalhes da instância, copie o endereço IP em Endereço IPv4 público (por exemplo, **`192.0.2.4`**). Você usa esse endereço como o destino da função no AWS Lambda.

> **Note**
>
> A política padrão da função de serviço CodePipeline inclui as permissões do Lambda necessárias para invocar a função. No entanto, se você alterou a função de serviço padrão ou selecionou uma diferente, certifique-se de que a política para a função comporte as permissões `lambda:InvokeFunction` e `lambda:ListFunctions`. Caso contrário, haverá falha nos pipelines que incluem ações do Lambda.

Etapa 2: Criar a função do Lambda

Nesta etapa, você cria uma função do Lambda que faz uma solicitação HTTP e procura uma linha de texto em uma página da web. Como parte desta etapa, você também deve criar uma política do IAM e um perfil de execução do Lambda. Para obter mais informações, consulte Modelo de permissões no Guia do desenvolvedor do AWS Lambda .

Para criar a função de execução

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Escolha Policies (Políticas) e depois Create Policy (Criar política). Escolha a guia JSON e depois cole a política a seguir no campo.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Action": [
        "codepipeline:PutJobSuccessResult",
        "codepipeline:PutJobFailureResult"
        ],
        "Effect": "Allow",
        "Resource": "*"
     }
  ]
}
```

3. Selecione Revisar política.

4. Na página Review policy (Revisar política), em Name (Nome), digite um nome para a política (por exemplo, **`CodePipelineLambdaExecPolicy`**). Em Description (Descrição), insira **`Enables Lambda to execute code`**.

   Escolha Create Policy.

   > **Note**
   >
   > Essas são as permissões mínimas necessárias para que uma função do Lambda CodePipeline interopere com a Amazon. CloudWatch Se você quiser expandir essa política para permitir funções que interajam com outros AWS recursos, modifique essa política para permitir as ações exigidas por essas funções do Lambda.

5. Na página do painel da política, selecione Roles (Funções) e, depois, Create role (Criar função).
6. Na página Criar perfil, escolha AWS service (Serviço da AWS). Selecione Lambda, e, então, selecione Next: Permissions (Próximo: permissões).
7. Na página Attach permissions policies (Anexar políticas de permissões), marque a caixa de seleção ao lado de CodePipelineLambdaExecPolicy e, então, selecione Next: Tags (Próximo: tags). Escolha Próximo: revisar.
8. Na página Review (Revisar), em Role name (Nome da função), insira o nome e depois escolha Create role (Criar função).

Para criar a função Lambda de amostra para usar com CodePipeline

1. Faça login no Console de gerenciamento da AWS e abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Na página Functions (Funções), escolha Create function (Criar função).

   > **Note**
   >
   > Caso veja uma página de Boas-vindas, em vez da página do Lambda, escolha Comece a usar agora.

3. Na página Create function, selecione Author from scratch. Em Nome da função, insira um nome para a função do Lambda (por exemplo, **`MyLambdaFunctionForAWSCodePipeline`**). Em Tempo de execução, escolha Node.js 20.x.

4. Em Role (Função), selecione Choose an existing role (Selecionar uma função existente). Em Existing role (Função existente), escolha sua função, e depois escolha Create function (Criar função).

   A página de detalhes da função criada é aberta.

5. Copie o código a seguir na caixa Function code (Código da função):

> **Note**
>
> O objeto do evento, abaixo da CodePipeline.job chave, contém os detalhes do trabalho. Para ver um exemplo completo dos CodePipeline retornos do evento JSON para o Lambda, consulte. Exemplo 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. Deixe Handler (Manipulador) como o valor padrão e deixe Role (Função) como padrão **`CodePipelineLambdaExecRole`**.
7. Em Basic settings (Configurações básicas), para Timeout (Tempo limite), insira **`20`** segundos.
8. Escolha Salvar.

Etapa 3: adicionar a função Lambda a um pipeline no console CodePipeline

Nesta etapa, você adiciona um novo estágio a seu pipeline e, em seguida, adiciona uma ação do Lambda que chama sua função nesse estágio.

Para adicionar um estágio

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Na página Welcome (Bem-vindo), escolha o pipeline que criou.
3. Na página de visualização do pipeline, selecione Editar.
4. Na página Editar, escolha + Adicionar estágio para adicionar um estágio após o estágio de implantação com a CodeDeploy ação. Insira um nome (por exemplo, **`LambdaStage`**) e escolha o Add stage (Adicionar estágio).

   > **Note**
   >
   > Você também pode optar por adicionar a ação do Lambda a um estágio existente. Para fins de demonstração, estamos adicionando a função do Lambda como a única ação em um estágio que permite a você visualizar facilmente seu progresso à medida que os artefatos progridem em um pipeline.

5. Escolha + Add action group (Adicionar grupo de ação). Em Editar ação, em Nome da ação, insira um nome para sua ação do Lambda (por exemplo, **`MyLambdaAction`**). Em Provider (Provedor), selecione AWS Lambda. Em Nome da função, selecione ou insira o nome da sua função do Lambda (por exemplo, **`MyLambdaFunctionForAWSCodePipeline`**). Em Parâmetros do usuário, especifique o endereço IP da instância do Amazon EC2 que você copiou anteriormente (por exemplo, **`http://192.0.2.4`**) e escolha Concluído.

   > **Note**
   >
   > Este tópico usa um endereço IP, mas em um cenário do mundo real, você pode fornecer o nome do site registrado (por exemplo, **`http://www.example.com`**). Para obter mais informações sobre dados e manipuladores de eventos em AWS Lambda, consulte Modelo de programação no Guia do AWS Lambda desenvolvedor.

6. Na página Edit action (Editar ação), escolha Save (Salvar).

Etapa 4: Testar o pipeline com a função do Lambda

Para testar a função, libere a alteração mais recente pelo pipeline.

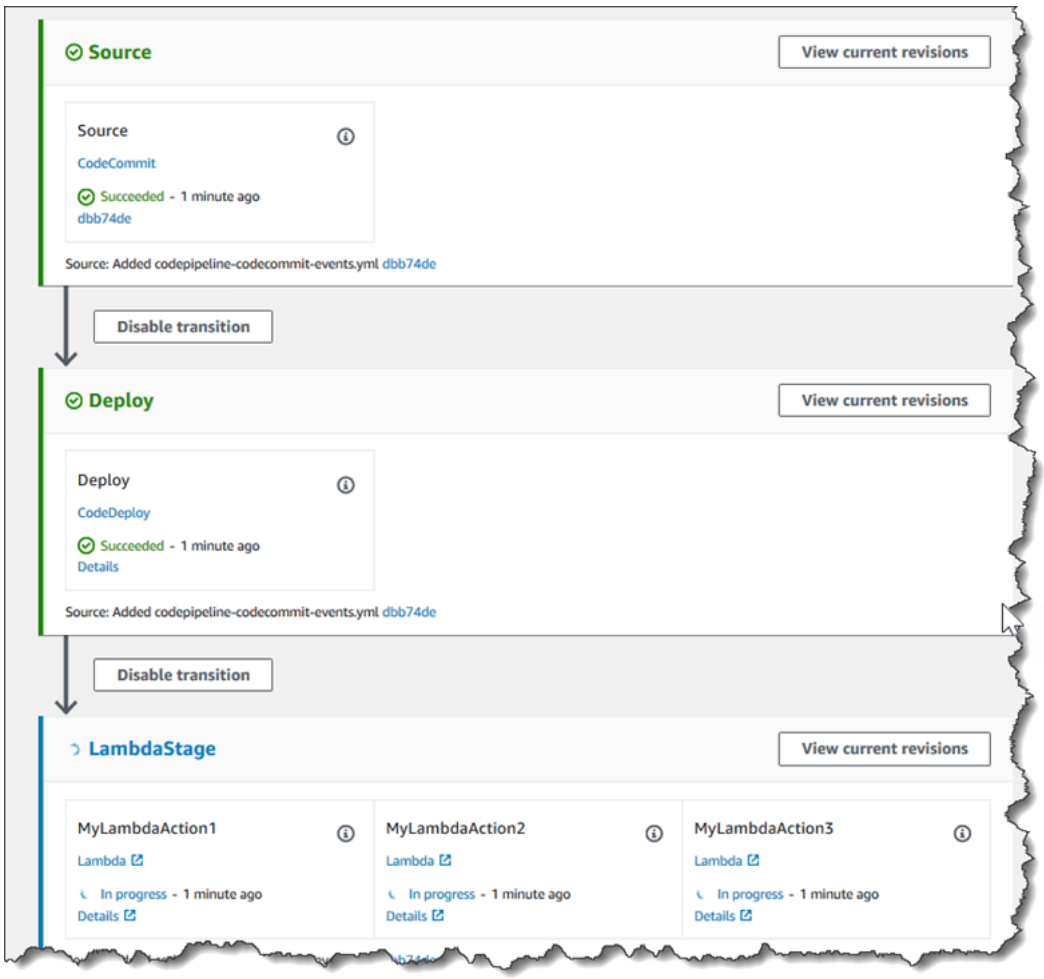
Para usar o console para executar a versão mais recente de um artefato através de um pipeline

1. Na página de detalhes do pipeline, escolha Lançar alteração. Essa ação executa a revisão mais recente disponível em cada local de origem especificado em uma ação de origem do pipeline.
2. Quando a ação do Lambda estiver concluída, escolha o link Detalhes para visualizar o fluxo de log da função na Amazon CloudWatch, incluindo a duração faturada do evento. Se a função falhar, o CloudWatch registro fornecerá informações sobre a causa.

Etapa 5: Próximas etapas

Agora que você criou uma função do Lambda com êxito e a adicionou como uma ação em um pipeline, tente o seguinte:

- Adicione mais ações do Lambda ao seu estágio para verificar outras páginas da web.
- Modifique a função do Lambda para procurar outra string de texto.
- Explore as funções do Lambda e crie e adicione suas próprias funções do Lambda aos pipelines.



Depois de terminar de experimentar a função Lambda, considere removê-la do seu pipeline, excluí-la e excluir a função do AWS Lambda IAM para evitar possíveis cobranças. Para obter mais informações, consulte Editar um pipeline no CodePipeline, Excluir um pipeline no CodePipeline. e Exclusão de funções ou de perfis de instância.

Exemplo de evento JSON

O exemplo a seguir mostra um exemplo de evento JSON enviado ao Lambda por. CodePipeline A estrutura deste evento é semelhante à resposta à `GetJobDetails API`, mas sem os tipos de dados `actionTypeId` e `pipelineContext`. Dois detalhes de configuração da ação, `FunctionName` e `UserParameters`, estão incluídos no evento JSON e na resposta à API `GetJobDetails`. Os valores em *`red italic text`* são exemplos ou explicações, não valores reais.

```
{
    "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"
            }
        }
    }
}
```

Funções de exemplo adicionais

Os exemplos de funções do Lambda a seguir demonstram funcionalidades adicionais que você pode usar em seus pipelines. CodePipeline Para usar essas funções, talvez você tenha que modificar a política para a função de execução do Lambda, como observado na introdução de cada exemplo.

Tópicos

- Exemplo de função Python que usa um AWS CloudFormation modelo

Exemplo de função Python que usa um AWS CloudFormation modelo

O exemplo a seguir mostra uma função que cria ou atualiza uma pilha com base em um CloudFormation modelo fornecido. O modelo cria um bucket do Amazon S3. É somente para fins de demonstração, para minimizar custos. O ideal é que você exclua a stack antes de carregar algo no bucket. Se carregar arquivos no bucket, você não poderá excluir o bucket quando excluir a stack. Você deve excluir manualmente tudo no bucket antes que possa excluir o próprio bucket.

Esse exemplo de Python supõe que você tenha um pipeline que use um bucket do Amazon S3 como ação de origem ou que você tenha acesso a um bucket do Amazon S3 com controle de versão que possa ser usado com o pipeline. Você cria o CloudFormation modelo, o compacta e o carrega nesse bucket como um arquivo.zip. Você deve então adicionar uma ação de origem a seu pipeline que recupere esse arquivo .zip do bucket.

> **Note**
>
> Quando o Amazon S3 é o provedor de origem do pipeline, é possível compactar o(s) arquivo(s) de origem em um único .zip e fazer upload do .zip para o bucket de origem.

Também é possível fazer upload de um único arquivo descompactado; no entanto, ocorrerão falha nas ações downstream que aguardam um arquivo .zip.

Esse exemplo demonstra:

- O uso de parâmetros JSON-encoded do usuário para passar vários valores de configuração para a função (`get_user_params`).
- A interação com artefatos .zip em um bucket de artefatos (`get_template`).
- O uso de um token de continuação para monitorar um longo processo assíncrono (`continue_job_later`). Isso permite que a ação continue e a função tenha êxito mesmo se exceder um runtime de quinze minutos (um limite no Lambda).

Para usar esse exemplo de função do Lambda, a política para a função de execução do Lambda deve ter `Allow` permissões no CloudFormation Amazon S3 e CodePipeline, conforme mostrado neste exemplo 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 criar o CloudFormation modelo, abra qualquer editor de texto simples e copie e cole o seguinte 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"
    }
  }
}
```

Salve isso como um arquivo JSON com o nome **`template.json`** em um diretório chamado **`template-package`**. Crie um arquivo compactado (.zip) desse diretório e do arquivo chamado **`template-package.zip`** e faça upload do arquivo compactado em um bucket do Amazon S3 com

controle de versão. Se já tiver um bucket configurado para seu pipeline, você poderá usá-lo. Em seguida, edite o pipeline para adicionar uma ação de origem que recupere o arquivo .zip. Nomeie a saída para essa ação*MyTemplate*. Para obter mais informações, consulte Editar um pipeline no CodePipeline.

> **Note**
>
> O exemplo de função do Lambda espera esses nomes de arquivos e estrutura compactada. No entanto, você pode substituir esse exemplo por seu próprio CloudFormation modelo. Se você usar seu próprio modelo, certifique-se de modificar a política da função de execução do Lambda para permitir qualquer funcionalidade adicional exigida pelo seu CloudFormation modelo.

Para adicionar o código a seguir como uma função no Lambda

1. Abra o console do Lambda e escolha Criar função.
2. Na página Create function, selecione Author from scratch. Em Nome da função, insira um nome para a função do Lambda.
3. Em Runtime (Tempo de execução), escolha Python 2.7.
4. Em Escolher ou criar uma função de execução, selecione Usar um perfil existente. Em Existing role (Função existente), escolha sua função, e depois escolha Create function (Criar função).

   A página de detalhes da função criada é aberta.
5. Copie o código a seguir na caixa Function code (Código da função):

```
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. Deixe Manipulador com o valor padrão e Perfil com o nome que você selecionou ou criou anteriormente, **`CodePipelineLambdaExecRole`**.
7. Em Basic settings (Configurações básicas), para Timeout (Tempo limite), substitua o padrão de 3 segundos por **`20`**.
8. Escolha Salvar.
9. No CodePipeline console, edite o pipeline para adicionar a função como uma ação em um estágio do seu funil. Escolha Editar para o estágio do pipeline que você deseja alterar e escolha Adicionar grupo de ações. Na página Editar ação, em Nome da ação, insira um nome para a ação. Em Provedor de ação, selecione Lambda.

   Em Artefatos de entrada, escolha `MyTemplate`. Em UserParameters, você deve fornecer uma string JSON com três parâmetros:

   - Nome da pilha
   - CloudFormation nome do modelo e caminho para o arquivo
   - Artefato de entrada

Use chaves ({ }) e separe os parâmetros com vírgulas. Por exemplo, para criar uma pilha chamada*MyTestStack*, para um pipeline com o artefato de entrada *MyTemplate* UserParameters, insira: {"stack”:” *MyTestStack* “, "file” :"template- package/template .json”, "artifact”:” “}. *MyTemplate*

> **Note**
>
> Mesmo que você tenha especificado o artefato de entrada em UserParameters, você também deve especificar esse artefato de entrada para a ação em Artefatos de entrada.

10. Salve as alterações efetuadas no pipeline e libere manualmente uma alteração para testar a ação e função do Lambda.

Adicionar uma ação de aprovação manual a uma fase

Em AWS CodePipeline, você pode adicionar uma ação de aprovação a um estágio em um funil no ponto em que você deseja que a execução do funil seja interrompida para que alguém com as AWS Identity and Access Management permissões necessárias possa aprovar ou rejeitar a ação.

Se a ação for aprovada, a execução do pipeline reiniciará. Se a ação for rejeitada, ou se ninguém aprovar ou rejeitar a ação em um prazo de sete dias depois que o pipeline acessar a ação e parar, o resultado será o mesmo de uma falha de ação e a execução do pipeline não prosseguirá.

Você pode usar aprovações manuais por estes motivos:

- Você deseja que alguém faça uma análise de código ou uma análise de gerenciamento de alterações antes de uma revisão ser permitida no próximo estágio de um pipeline.
- Você deseja que alguém faça testes manuais de garantia de qualidade na versão mais recente de um aplicativo ou confirme a integridade de um artefato de criação antes de ele ser lançado.
- Você deseja que alguém analise o texto novo ou atualizado antes de ele ser publicado no site de uma empresa.

Opções de configuração para ações de aprovação manual no CodePipeline

CodePipeline fornece três opções de configuração que você pode usar para informar os aprovadores sobre a ação de aprovação.

Publique notificações de aprovação Você pode configurar uma ação de aprovação para publicar uma mensagem em um tópico do Amazon Simple Notification Service quando o pipeline parar na ação. O Amazon SNS entrega a mensagem para cada endpoint inscrito no tópico. Você deve usar um tópico criado na mesma AWS região do funil que incluirá a ação de aprovação. Ao criar um tópico, é recomendável dar a ele um nome que identifique sua finalidade nos formatos como `MyFirstPipeline-us-east-2-approval`.

Ao publicar notificações de aprovação em tópicos do Amazon SNS, você pode escolher entre formatos como destinatários de e-mail ou SMS, filas SQS, HTTP/HTTPS endpoints ou AWS Lambda funções que você invoca usando o Amazon SNS. Para obter informações sobre as notificações dos tópicos do Amazon SNS, consulte estes tópicos:

- O que é o Amazon Simple Notification Service?
- Criar um tópico no Amazon SNS
- Como enviar mensagens do Amazon SNS para filas do Amazon SQS
- Como assinar uma fila de um tópico do Amazon SNS
- Envio de mensagens do Amazon SNS para endpoints HTTP/HTTPS
- Como chamar funções Lambda usando notificações do Amazon SNS

Para a estrutura dos dados JSON gerados para uma notificação de ação de aprovação, consulte Formato de dados JSON para notificações de aprovação manual em CodePipeline.

Especificar um URL para revisão Como parte da configuração da ação de aprovação, você pode especificar um URL a ser revisado. O URL pode ser um link para um aplicativo web que você deseja que os responsáveis pela aprovação testem ou uma página com mais informações sobre a solicitação de aprovação. O URL está incluído na notificação publicada no tópico do Amazon SNS. Os responsáveis pela aprovação podem usar o console ou o ILC para vê-lo.

Inserir comentários para os responsáveis pela aprovação Ao criar uma ação de aprovação, você também pode adicionar comentários que são exibidos às pessoas que receberem as notificações ou às pessoas que visualizarem a ação na resposta da ILC ou do console.

Sem opções de configuração Você também pode optar por não configurar nenhuma das três opções. As notificações não serão necessárias se, por exemplo, você notificar uma pessoa diretamente de que a ação está pronta para a revisão ou desejar simplesmente que o pipeline seja encerrado até que você decida aprovar a ação por conta própria.

Visão geral da configuração e do fluxo de trabalho para ações de aprovação no CodePipeline

Consulte uma visão geral de como configurar e usar aprovações manuais.

1. Você concede a um ou mais perfis do IAM na sua organização as permissões do IAM necessárias para aprovar ou rejeitar ações de aprovação.
2. (Opcional) Se você estiver usando notificações do Amazon SNS, você garante que a função de serviço que você usa em suas CodePipeline operações tenha permissão para acessar os recursos do Amazon SNS.
3. (Opcional) Se você estiver usando notificações do Amazon SNS, crie um tópico do Amazon SNS e adicione um ou mais assinantes ou endpoints a ele.
4. Ao usar a AWS CLI para criar o pipeline ou depois de usar a CLI ou o console para criar o pipeline, você adiciona uma ação de aprovação a um estágio no pipeline.

   Se você estiver usando notificações, incluirá o nome do recurso da Amazon (ARN) do tópico do Amazon SNS na configuração da ação. (Um ARN é um identificador exclusivo para um recurso da Amazon. Os ARNs dos tópicos do Amazon SNS são estruturados da mesma forma. *arn:aws:sns:us-east-2:80398EXAMPLE:MyApprovalTopic* Para obter mais informações, consulte Nomes de recursos da Amazon (ARNs) e AWS service (Serviço da AWS) namespaces no.) Referência geral da Amazon Web Services
5. O pipeline para quando acessa a ação de aprovação. Se um ARN do tópico do Amazon SNS foi incluído na configuração da ação, uma notificação será publicada no tópico e uma mensagem será entregue a todos os assinantes do tópico ou endpoints inscritos, com um link para analisar a ação de aprovação no console.
6. Um responsável pela aprovação examinará o URL de destino e os comentários, se houver.
7. Usando o console, CLI ou SDK, o responsável pela aprovação fará um comentário de resumo e enviará uma resposta:
   - Aprovado: a execução do pipeline é retomada.
   - Rejeitado: o status do estágio é alterado para "Reprovado", e a execução do pipeline não é retomada.

   Se nenhuma resposta for enviada no prazo de sete dias, a ação será marcada como "Reprovado".

Conceda permissões de aprovação a um usuário do IAM em CodePipeline

Para que os usuários do IAM de sua organização consigam aprovar ou rejeitar ações de aprovação, eles devem receber permissões para acessar os pipelines e atualizar o status das ações de aprovação. Você pode conceder permissão de acesso a todos os pipelines e ações de aprovação em sua conta anexando a política gerenciada `AWSCodePipelineApproverAccess` a um usuário, perfil ou grupo do IAM ou pode conceder permissões limitadas especificando recursos individuais que podem ser acessados por um usuário, perfil ou grupo do IAM.

> **Note**
>
> As permissões descritas neste tópico concedem acesso bem limitado. Para permitir que um usuário, função ou grupo façam mais do que aprovar ou rejeitar ações de aprovação, é possível anexar outras políticas gerenciadas. Para obter informações sobre as políticas gerenciadas disponíveis para CodePipeline, consulteAWS políticas gerenciadas para AWS CodePipeline.

Conceder permissão de aprovação a todos os pipelines e ações de aprovação

Para usuários que precisam realizar ações de aprovação em CodePipeline, use a política `AWSCodePipelineApproverAccess` gerenciada.

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em Centro de Identidade do AWS IAM:

  Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do Centro de Identidade do AWS IAM .

- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criando um perfil para um provedor de identidades de terceiros (federação) no Guia do Usuário do IAM.

- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.

- (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

Definir uma permissão de aprovação para pipelines e ações de aprovação específicos

Para usuários que precisam realizar ações de aprovação no CodePipeline, use a política personalizada a seguir. Na política abaixo, especifique os recursos individuais que um usuário pode acessar. Por exemplo, a política a seguir concede aos usuários a autoridade para aprovar ou rejeitar apenas a ação `MyApprovalAction` no pipeline `MyFirstPipeline` na região Leste dos EUA (Ohio) (us-east-2):

> **Note**
>
> A `codepipeline:ListPipelines` permissão é necessária somente se os usuários do IAM precisarem acessar o CodePipeline painel para visualizar essa lista de pipelines. Se o acesso ao console não for necessário, omita `codepipeline:ListPipelines`.

Para usar o editor de políticas JSON para criar uma política

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Políticas.

   Se essa for a primeira vez que você escolhe Políticas, a página Bem-vindo às políticas gerenciadas será exibida. Escolha Começar.
3. Na parte superior da página, escolha Criar política.
4. Na seção Editor de políticas, escolha a opção JSON.
5. Insira o seguinte 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. Escolha Próximo.

> **Note**
>
> É possível alternar entre as opções de editor Visual e JSON a qualquer momento. Porém, se você fizer alterações ou escolher Próximo no editor Visual, o IAM poderá reestruturar a política a fim de otimizá-la para o editor visual. Para obter mais informações, consulte Restruturação de política no Guia do usuário do IAM.

7. Na página Revisar e criar, insira um Nome de política e uma Descrição (opcional) para a política que você está criando. Revise Permissões definidas nessa política para ver as permissões que são concedidas pela política.
8. Escolha Criar política para salvar sua nova política.

Conceder permissões ao Amazon SNS para uma função CodePipeline de serviço

Se você planeja usar o Amazon SNS para publicar notificações em tópicos quando as ações de aprovação precisarem ser revisadas, a função de serviço que você usa em suas CodePipeline operações deve ter permissão para acessar os recursos do Amazon SNS. Você pode usar o console do IAM para adicionar essa permissão ao seu perfil de serviço.

Na política abaixo, especifique a política para publicação com o SNS. Você pode nomeá-la como `SNSPublish`. Use a política a seguir anexando-a ao seu perfil de serviço.

> **Important**
>
> Certifique-se de estar conectado ao Console de gerenciamento da AWS com as mesmas informações da conta que você usouComeçando com CodePipeline.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "*"
        }
    ]
}
```

Para usar o editor de políticas JSON para criar uma política

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Políticas.

   Se essa for a primeira vez que você escolhe Políticas, a página Bem-vindo às políticas gerenciadas será exibida. Escolha Começar.

3. Na parte superior da página, escolha Criar política.
4. Na seção Editor de políticas, escolha a opção JSON.
5. Insira ou cole um documento de política JSON. Para obter detalhes sobre a linguagem da política do IAM, consulte a referência de política JSON do IAM.
6. Resolva os avisos de segurança, erros ou avisos gerais gerados durante a validação de política e depois escolha Próximo.

> **Note**
>
> Você pode alternar entre as opções de editor Visual e JSON a qualquer momento. Porém, se você fizer alterações ou escolher Próximo no editor Visual, o IAM poderá reestruturar a política a fim de otimizá-la para o editor visual. Para obter mais informações, consulte Restruturação de política no Guia do usuário do IAM.

7. (Opcional) Ao criar ou editar uma política no Console de gerenciamento da AWS, você pode gerar um modelo de política JSON ou YAML que pode ser usado nos CloudFormation modelos.

   Para fazer isso, no editor de políticas, escolha Ações e, em seguida, escolha Gerar CloudFormation modelo. Para saber mais CloudFormation, consulte a referência AWS Identity and Access Management de tipo de recurso no Guia AWS CloudFormation do usuário.
8. Quando terminar de adicionar as permissões à política, escolha Avançar.
9. Na página Revisar e criar, insira um Nome de política e uma Descrição (opcional) para a política que você está criando. Revise Permissões definidas nessa política para ver as permissões que são concedidas pela política.
10. (Opcional) Adicione metadados à política associando tags como pares de chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Tags para AWS Identity and Access Management recursos no Guia do usuário do IAM.
11. Escolha Criar política para salvar sua nova política.

Adicione uma ação de aprovação manual a um funil no CodePipeline

Você pode adicionar uma ação de aprovação a um estágio em um CodePipeline funil no ponto em que deseja que o funil pare para que alguém possa aprovar ou rejeitar manualmente a ação.

> **Note**
>
> As ações de aprovação não podem ser adicionadas aos estágios Origem. Os estágios Origem podem conter somente ações de origem.

Para usar o Amazon SNS para enviar notificações quando uma ação de aprovação estiver pronta para revisão, primeiro você deverá estar em conformidade com estes pré-requisitos:

- Conceda permissão à sua função CodePipeline de serviço para acessar os recursos do Amazon SNS. Para mais informações, consulte Conceder permissões ao Amazon SNS para uma função CodePipeline de serviço.
- Conceder permissão a uma ou mais identidades do IAM em sua organização para atualizar o status de uma ação de aprovação. Para mais informações, consulte Conceda permissões de aprovação a um usuário do IAM em CodePipeline.

Neste exemplo, você cria um novo estágio de aprovação e adiciona uma ação de aprovação manual ao estágio. Você também pode adicionar uma ação de aprovação manual a um estágio existente que contém outras ações.

Adicionar uma ação de aprovação manual a um CodePipeline pipeline (console)

Você pode usar o CodePipeline console para adicionar uma ação de aprovação a um CodePipeline funil existente. Você deve usar a AWS CLI se quiser adicionar ações de aprovação ao criar um novo funil.

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Em Nome, selecione o pipeline.
3. Na página de detalhes do pipeline, selecione Editar.
4. Se você deseja adicionar uma ação de aprovação a um novo estágio, selecione +Add stage (+Adicionar estágio) no ponto do pipeline onde você deseja adicionar uma solicitação de aprovação e insira um nome para o estágio. Na página Add stage (Adicionar estágio) em Stage name (Nome do estágio), insira o novo nome de estágio. Por exemplo, adicione um novo estágio e nomeie-o `Manual_Approval`.

   Se você deseja adicionar uma ação de aprovação a um estágio existente, selecione Edit stage (Editar estágio).

5. No estágio em que você deseja adicionar a ação de aprovação, escolha + Add action group (Adicionar grupo de ações).
6. Na página Edit action (Editar ação), faça o seguinte:
    1. Em Action name (Nome da ação), insira um nome para identificar a ação.
    2. Em Action provider (Fornecedor de ação), em Approval (Aprovação), escolha Manual approval (Aprovação manual).
    3. (Opcional) Em SNS topic ARN (ARN do tópico do SNS), selecione o nome do tópico que é usado para enviar notificações para a ação de aprovação.
    4. (Opcional) Em URL para revisão, insira a URL da página ou aplicativo que você deseja que o aprovador examine. Os aprovadores podem acessar essa URL através de um link incluído no console do pipeline.
    5. (Opcional) Em Comments (Comentários), insira as outras informações que deseja compartilhar com o revisor.
    6. Escolha Salvar.

Adicionar uma ação de aprovação manual a um CodePipeline pipeline (CLI)

Você pode usar a CLI para adicionar uma ação de aprovação a um pipeline existente ou ao criar um pipeline. Para fazer isso, inclua uma ação de aprovação com o tipo manual em um estágio que você está criando ou editando.

Para obter mais informações sobre como criar e editar pipelines, consulte Criar um pipeline, estágios e ações e Editar um pipeline no CodePipeline.

Para adicionar um estágio a um pipeline que inclua somente uma ação de aprovação, você inclui algo semelhante ao exemplo a seguir ao criar ou atualizar o pipeline.

> **Note**
>
> A seção `configuration` é opcional. Essa é apenas uma parte do arquivo, não a estrutura completa. Para obter mais informações, consulte CodePipeline referência de estrutura de tubulação.

```
{
```

```
    "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
        }
    ]
}
```

Se a ação de aprovação estiver em um estágio em que há outras ações, a seção do arquivo JSON que contém o estágio poderá ser parecido com o exemplo a seguir.

> **Note**
>
> A seção `configuration` é opcional. Essa é apenas uma parte do arquivo, não a estrutura completa. Para obter mais informações, consulte CodePipeline referência de estrutura de tubulação.

```
,
{
    "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
        }
    ]
}
```

Aprovar ou rejeitar uma ação de aprovação no CodePipeline

Quando um pipeline inclui uma ação de aprovação, a execução do pipeline é interrompida no ponto em que a ação foi adicionada. O pipeline não reiniciará a menos que alguém aprove manualmente a ação. Se um responsável pela autorização rejeitar a ação ou se nenhuma resposta de aprovação for recebida no prazo de sete dias da interrupção do pipeline para a ação de aprovação, o status do pipeline será "Reprovado".

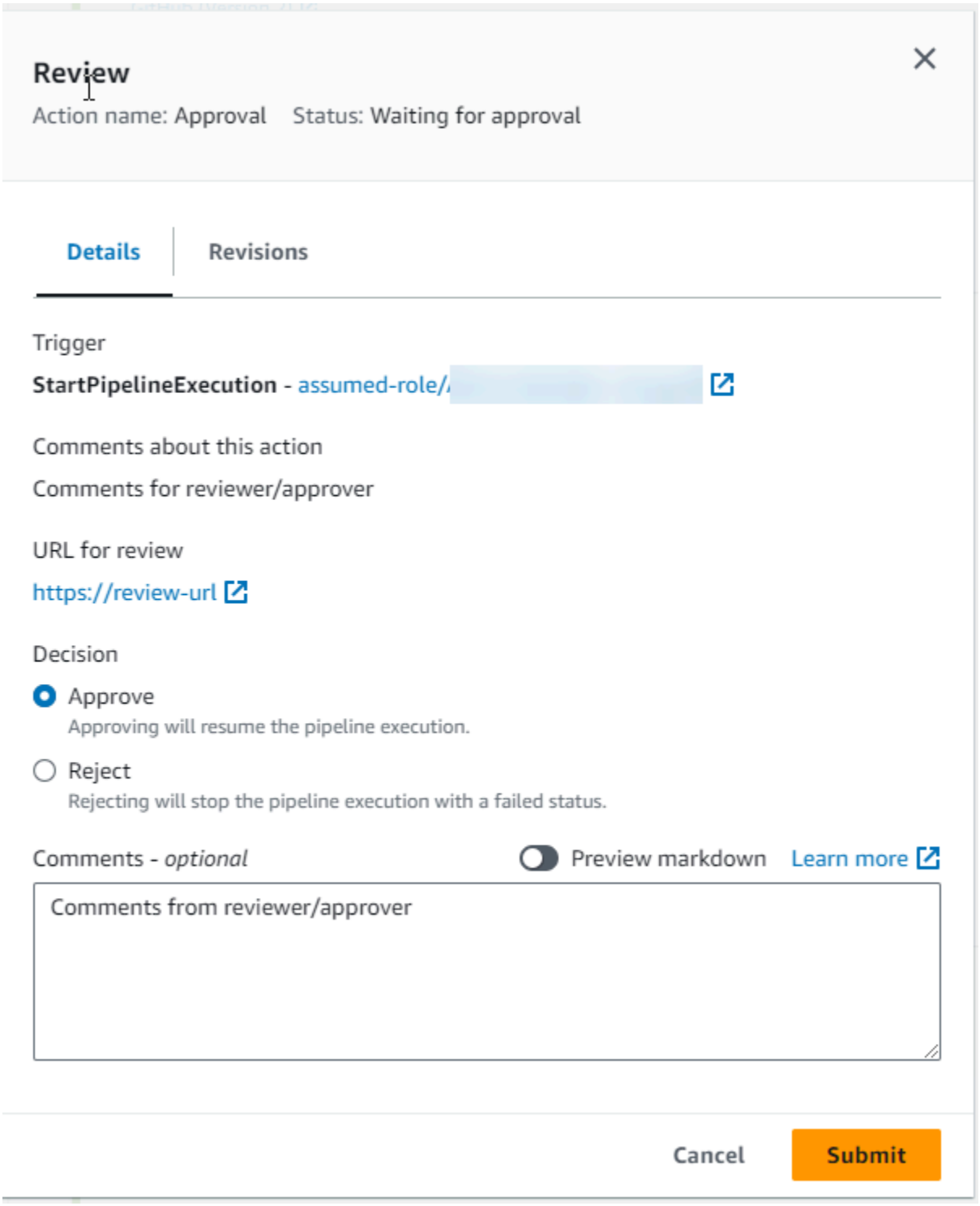
Se a pessoa que adicionou a ação de aprovação ao pipeline configurou notificações, você poderá receber um e-mail com as informações do pipeline e o status da aprovação.

Aprovar ou rejeitar uma ação de aprovação (console)

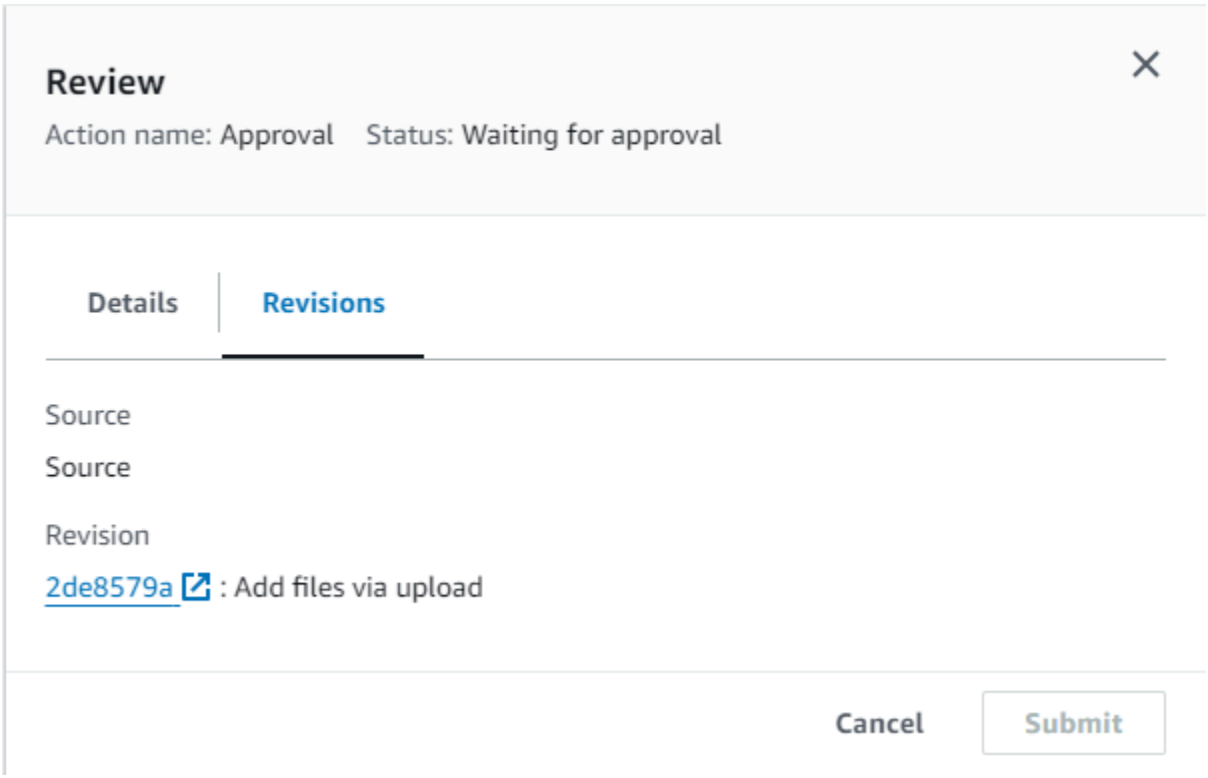
Se você receber uma notificação que inclui um link direto para uma ação de aprovação, selecione o link de Approve or reject (Aprovar ou rejeitar), faça login no console, e passe para a etapa 7 aqui. Do contrário, siga todas estas etapas.

1. Abra o CodePipeline console em https://console.aws.amazon.com/codepipeline/.
2. Na página All Pipelines, selecione o nome do pipeline.
3. Localize o estágio com a ação de aprovação. Escolha Revisar.

   A caixa de diálogo Revisar é exibida. A guia Detalhes mostra o conteúdo e os comentários da revisão.



A guia Revisões mostra as revisões de origem para a execução.



4. Na guia Detalhes, visualize os comentários e o URL, se houver. A mensagem também exibe a URL de conteúdo para sua revisão, se ela tiver sido incluída.
5. Se um URL foi fornecido, selecione o link URL para revisar na ação para abrir a página da web de destino e depois revise o conteúdo.
6. Na janela Revisar, insira comentários de revisão, por exemplo, por que você está aprovando ou rejeitando a ação e, depois, selecione Aprovar ou Rejeitar.
7. Selecione Enviar.

Aprovar ou rejeitar uma solicitação de aprovação (CLI)

Para usar a ILC para responder a uma ação de aprovação, primeiro você deve usar o comando get-pipeline-state para recuperar o token associado à execução mais recente da ação de aprovação.

1. Em um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows), execute o comando get-pipeline-state no pipeline que contém a ação de aprovação. Por exemplo, para um pipeline nomeado *MyFirstPipeline*, insira o seguinte:

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

2. Na resposta ao comando, localize o valor `token`, que aparece em `latestExecution` na seção `actionStates` da ação de aprovação, como exibido aqui:

```
{
    "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. Em um editor de texto plano, crie um arquivo para adicionar o seguinte, no formato JSON:

   - O nome do pipeline que contém a ação de aprovação.
   - O nome do estágio que contém a ação de aprovação.
   - O nome da ação de aprovação.
   - O valor do token que você coletou na etapa anterior.
   - Sua resposta à ação (Aprovada ou Rejeitada). A resposta deve ser capitalizada.
   - Seu resumo de comentários.

   Para o exemplo anterior *MyFirstPipeline*, seu arquivo deve ser algo parecido com isto:

```
{
  "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. Salve o arquivo com um nome como **`approvalstage-approved.json`**.
5. Execute o comando put-approval-result, especificando o nome do arquivo JSON de aprovação, semelhante ao seguinte:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

```
aws codepipeline put-approval-result --cli-input-json file://approvalstage-approved.json
```

Formato de dados JSON para notificações de aprovação manual em CodePipeline

No caso de ações de aprovação que usam notificações do Amazon SNS, os dados JSON sobre a ação são criados e publicados no Amazon SNS quando o pipeline é interrompido. Você pode usar a saída JSON para enviar mensagens às filas do Amazon SQS ou invocar funções no AWS Lambda.

> **Note**
>
> Este guia não aborda como configurar notificações usando o JSON. Para obter informações, consulte Enviar mensagens do Amazon SNS para filas do Amazon SQS e Invocar funções do Lambda usando notificações do Amazon SNS no Guia de desenvolvedor do Amazon SNS.

O exemplo a seguir mostra a estrutura do resultado JSON disponível com aprovações do 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."
    }
}
```

Adicione uma ação entre regiões em CodePipeline

AWS CodePipeline inclui várias ações que ajudam você a configurar, criar, testar e implantar recursos para seu processo de lançamento automatizado. Você pode adicionar ações ao seu funil que estejam em uma AWS região diferente do seu funil. Quando an AWS service (Serviço da AWS) é o provedor de uma ação e esse type/provider tipo de ação está em uma AWS região diferente do seu funil, essa é uma ação entre regiões.

> **Note**
>
> Cross-region as ações são suportadas e só podem ser criadas nas AWS regiões em que CodePipeline há suporte. Para obter uma lista das AWS regiões suportadas para CodePipeline, consulteCotas em AWS CodePipeline.

Você pode usar o console ou CloudFormation adicionar ações entre regiões em pipelines. AWS CLI

> **Note**
>
> Alguns tipos de ação CodePipeline podem estar disponíveis somente em determinadas AWS regiões. Observe também que pode haver AWS regiões em que um tipo de ação está disponível, mas um AWS provedor específico para esse tipo de ação não está disponível.

Ao criar ou editar um pipeline, é necessário ter um bucket de artefato na região do pipeline e ter um bucket de artefato por região em que planeja executar uma ação. Para obter mais informações sobre o parâmetro `ArtifactStores`, consulte CodePipeline referência de estrutura de tubulação.

> **Note**
>
> CodePipeline manipula a cópia de artefatos de uma AWS região para outras regiões ao realizar ações entre regiões.

Se você usar o console para criar um pipeline ou ações entre regiões, os buckets de artefatos padrão serão configurados nas regiões CodePipeline em que você tem ações. Ao usar o AWS CLI, CloudFormation, ou um SDK para criar um funil ou ações entre regiões, você fornece o bucket de artefatos para cada região em que você tem ações.

> **Note**
>
> Você deve criar o bucket de artefatos e a chave de criptografia na mesma AWS região da ação entre regiões e na mesma conta do seu funil.

Você não pode criar ações entre regiões para os seguintes tipos de ação:

- Ações de origem
- Third-party ações
- Ações personalizadas

> **Note**
>
> Ao usar a ação de invocação do Lambda entre regiões em CodePipeline, o status da execução do Lambda usando o PutJobSuccessResult e PutJobFailureResult deve ser enviado para a AWS região em que a função Lambda está presente e não para a região em que existe. CodePipeline

Quando um pipeline inclui uma ação entre regiões como parte de um estágio, CodePipeline replica somente os artefatos de entrada da ação entre regiões da região do pipeline para a região da ação.

> **Note**
>
> A região do pipeline e a região em que seus recursos de detecção de alterações de CloudWatch eventos são mantidos permanecem as mesmas. A região em que o pipeline é hospedado não se altera.

Gerencie ações entre regiões em um pipeline (console)

Você pode usar o CodePipeline console para adicionar uma ação entre regiões a um pipeline existente. Para criar um novo pipeline com ações entre regiões usando o assistente Create pipeline (Criar pipeline), consulte Criar um pipeline personalizado (console).

No console, você cria uma ação entre regiões em um estágio de funil escolhendo o provedor de ação e o campo Região, que lista os recursos que você criou nessa região para esse provedor. Quando você adiciona uma ação entre regiões, CodePipeline usa um bucket de artefatos separado na região da ação. Para obter mais informações sobre buckets de artefatos entre regiões, consulte CodePipeline referência de estrutura de tubulação.

Adicionar uma ação entre regiões a um estágio do pipeline (console)

Use o console para adicionar uma ação entre regiões a um pipeline.

> **Note**
>
> Se o pipeline estiver em execução quando as alterações forem salvas, essa execução não será concluída.

Adicionar uma ação entre regiões

1. Faça login no console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Selecione o pipeline e escolha Edit (Editar).
3. Na parte inferior do diagrama, escolha + Add stage (+ Adicionar estágio) se estiver adicionando um novo estágio ou escolha Edit stage (Editar estágio) se quiser adicionar a ação a um estágio existente.
4. Em Edit: <Stage> (Editar: <estágio>), escolha + Add action group (+ Adicionar grupo de ações) para adicionar uma ação serial. Ou escolha + Add action (+ Adicionar ação) para adicionar uma ação paralela.
5. Na página Edit action (Editar ação):
   a. Em Action name (Nome da ação), insira um nome para a ação entre regiões.
   b. Em Action provider (Provedor de ação), escolha o provedor de ação.
   c. Em Região, escolha a AWS região em que você criou ou planeja criar o recurso para a ação. Quando a região for selecionada, os recursos disponíveis para essa região serão listados para seleção. O campo Região designa onde os AWS recursos são criados para esse tipo de ação e tipo de provedor. Esse campo é exibido apenas para ações em que o provedor de ação é um AWS service (Serviço da AWS). O campo Região assume como valor padrão a mesma Região da AWS do seu pipeline.
   d. Em Input artifacts (Artefatos de entrada) escolha a entrada adequada do estágio anterior. Por exemplo, se o estágio anterior for um estágio de origem, escolha SourceArtifact.
   e. Preencha todos os campos obrigatórios para o provedor de ação que está configurando.
   f. Em Output artifacts (Artefatos de saída) escolha a saída adequada para o próximo estágio. Por exemplo, se o próximo estágio for um estágio de implantação, escolha BuildArtifact.
   g. Escolha Salvar.
6. Em Edit: <Stage> (Editar: <estágio>), escolha Done (Concluído).
7. Escolha Salvar.

Editar uma ação entre regiões em um estágio de pipeline (console)

Use o console para editar uma ação entre regiões existente em um pipeline.

> **Note**
>
> Se o pipeline estiver em execução quando as alterações forem salvas, essa execução não será concluída.

Para editar uma ação entre regiões

1. Faça login no console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Selecione o pipeline e escolha Edit (Editar).
3. Selecione Edit stage (Editar estágio).
4. Em Edit: <Stage> (Editar: <estágio>), escolha o ícone para editar uma ação existente.
5. Na página Edit action (Editar ação), faça alterações nos campos, conforme apropriado.
6. Em Edit: <Stage> (Editar: <estágio>), escolha Done (Concluído).
7. Escolha Salvar.

Excluir uma ação entre regiões de um estágio de pipeline (console)

Use o console para excluir uma ação entre regiões existente de um pipeline.

> **Note**
>
> Se o pipeline estiver em execução quando as alterações forem salvas, essa execução não será concluída.

Para excluir uma ação entre regiões

1. Faça login no console em http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Selecione o pipeline e escolha Edit (Editar).
3. Selecione Edit stage (Editar estágio).
4. Em Edit: <Stage> (Editar: <estágio>), escolha o ícone para excluir uma ação existente.
5. Em Edit: <Stage> (Editar: <estágio>), escolha Done (Concluído).
6. Escolha Salvar.

Adicionar uma ação entre regiões a um pipeline (CLI)

Você pode usar o AWS CLI para adicionar uma ação entre regiões a um pipeline existente.

Para criar uma ação entre regiões em um estágio de pipeline com o AWS CLI, você adiciona a ação de configuração junto com um `region` campo opcional. Você também deve já ter criado um balde de artefatos na região da ação. Em vez de fornecer o parâmetro `artifactStore` do pipeline de região única, use o parâmetro `artifactStores` para incluir uma listagem de cada bucket de artefatos da região.

> **Note**
>
> Neste passo a passo e em seus exemplos, *`RegionA`* está a região em que o pipeline é criado. Ele tem acesso ao bucket do *`RegionA`* Amazon S3 usado para armazenar artefatos do pipeline e à função de serviço usada pelo. CodePipeline *`RegionB`*é a região em que o CodeDeploy aplicativo, o grupo de implantação e a função de serviço usados por CodeDeploy são criados.

Pré-requisitos

É necessário criar os seguintes itens:

- Um oleoduto em*`RegionA`*.
- Um bucket de artefato do Amazon S3 em. *`RegionB`*
- Os recursos para sua ação, como seu CodeDeploy aplicativo e grupo de implantação para uma ação de implantação entre regiões, em*`RegionB`*.

Adicionar uma ação entre regiões a um pipeline (CLI)

Use o AWS CLI para adicionar uma ação entre regiões a um pipeline.

Adicionar uma ação entre regiões

1. Para entrar um pipeline*`RegionA`*, execute o get-pipeline comando para copiar a estrutura do pipeline em um arquivo JSON. Por exemplo, para um pipeline nomeado `MyFirstPipeline`, execute o seguinte comando:

```
aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
```

Este comando retorna nada, mas o arquivo que você criou deve aparecer no diretório onde você executou o comando.

2. Adicione o campo `region` para adicionar um novo estágio com sua ação entre regiões que inclui a região e os recursos para sua ação. O exemplo de JSON a seguir adiciona um estágio de implantação com uma ação de implantação entre regiões em que o provedor está CodeDeploy, em uma nova região. `us-east-1`

```
{
                "name": "Deploy",
                "actions": [
                    {
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "name": "Deploy",
                        "region": "RegionB",
                        "actionTypeId": {
                            "category": "Deploy",
                            "owner": "AWS",
                            "version": "1",
                            "provider": "CodeDeploy"
                        },
                        "outputArtifacts": [],
                        "configuration": {
                            "ApplicationName": "name",
                            "DeploymentGroupName": "name"
                        },
                        "runOrder": 1
                    }
```

3. Na estrutura do pipeline, remova o campo `artifactStore` e adicione o mapa `artifactStores` para sua nova ação entre regiões. O mapeamento deve incluir uma entrada para cada AWS região na qual você tem ações. Para cada entrada no mapeamento, os recursos devem estar na respectiva AWS região. No exemplo abaixo, `ID-A` é o ID da chave de criptografia para *`RegionA`* e `ID-B` é o ID da chave de criptografia para*`RegionB`*.

```
"artifactStores":{
   "RegionA":{
      "encryptionKey":{
         "id":"ID-A",
         "type":"KMS"
      },
      "location":"Location1",
      "type":"S3"
   },
   "RegionB":{
      "encryptionKey":{
         "id":"ID-B",
         "type":"KMS"
      },
      "location":"Location2",
      "type":"S3"
   }
}
```

O exemplo de JSON a seguir exibe o bucket da us-west-2 como `my-storage-bucket` e adiciona o novo bucket da us-east-1 chamado `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. Se você estiver trabalhando com a estrutura do pipeline recuperada por meio do comando get-pipeline, remova as linhas `metadata` do arquivo JSON. Caso contrário, o comando update-pipeline não é capaz de utilizá-la. Remova as linhas `"metadata": { }`, `"created"`, `"pipelineARN"` e os campos `"updated"`.

   Por exemplo, remova as seguintes linhas da estrutura:

```
"metadata": {
```

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

Salve o arquivo.

5. Para aplicar suas alterações, execute o comando update-pipeline especificando o arquivo JSON do pipeline:

> **Important**
>
> Não se esqueça de incluir `file://` antes do nome de arquivo. Ele é obrigatório nesse comando.

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

Este comando retorna toda a estrutura do pipeline editado. A saída é semelhante à seguinte.

```
{
    "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**
>
> O comando update-pipeline interrompe o pipeline. Se uma revisão estiver sendo executada pelo pipeline quando você executar o comando update-pipeline, essa execução será interrompida. Você deve iniciar manualmente o pipeline para executar a revisão através do pipeline atualizado. Use o comando **`start-pipeline-execution`** para iniciar manualmente o pipeline.

6. Depois de atualizar seu funil, a ação entre regiões é exibida no console.



Adicione uma ação entre regiões a um pipeline (CloudFormation)

Você pode usar CloudFormation para adicionar uma ação entre regiões a um funil existente.

Para adicionar uma ação entre regiões com CloudFormation

1. Adicione o parâmetro `Region` ao recurso `ActionDeclaration` em seu modelo, conforme mostrado no exemplo a seguir:

```
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. Em `Mappings`, adicione o mapa de região como mostrado neste exemplo para um mapeamento chamado `SecondRegionMap` que mapeia valores para as chaves `RegionA` e `RegionB`. No recurso `Pipeline`, no campo `artifactStore`, adicione o mapa `artifactStores` para sua nova ação entre regiões da seguinte forma:

```
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
```

O exemplo de YAML a seguir mostra o *RegionA* bucket como `us-west-2` e adiciona o novo *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. Salve o modelo atualizado em seu computador local e abra o console do CloudFormation .
4. Selecione sua pilha e clique em Create Change Set for Current Stack (Criar conjunto de alterações para a pilha atual).
5. Carregue o modelo e visualize as alterações listadas no CloudFormation. Essas são as alterações a serem feitas na pilha. Seus novos recursos devem ser exibidos na lista.
6. Clique em Executar.

Trabalhar com variáveis

Algumas ações na CodePipeline geração de variáveis. Para usar variáveis:

- Atribua um namespace a uma ação a fim de disponibilizar as variáveis que ele produz para uma configuração de ação downstream.
- Configure a ação downstream para consumir as variáveis geradas pela ação.

  Você pode visualizar os detalhes de cada execução de ação para ver os valores de cada variável de saída gerada pela ação em tempo de execução.

Para ver step-by-step exemplos de uso de variáveis:

- Para ver um tutorial com uma ação Lambda que usa variáveis de uma ação upstream (CodeCommit) e gera variáveis de saída, consulte. Tutorial: Usar variáveis com ações de invocação do Lambda
- Para ver um tutorial com uma CloudFormation ação que faz referência às variáveis de saída da pilha de uma CloudFormation ação upstream, consulte. Tutorial: criar um pipeline que usa variáveis de AWS CloudFormation ações de implantação
- Para obter um exemplo de ação de aprovação manual com texto de mensagem que faz referência a variáveis de saída que são resolvidas para o CodeCommit ID de confirmação e a mensagem de confirmação, consulteExemplo: Usar variáveis em aprovações manuais.
- Para obter um exemplo de CodeBuild ação com uma variável de ambiente que é resolvida com o nome da GitHub ramificação, consulteExemplo: use uma BranchName variável com variáveis de CodeBuild ambiente.
- CodeBuild as ações produzem como variáveis todas as variáveis de ambiente que foram exportadas como parte da construção. Para obter mais informações, consulte CodeBuild variáveis de saída da ação. Para obter uma lista das variáveis de ambiente que você pode usar CodeBuild, consulte Variáveis de ambiente em ambientes de compilação no Guia AWS CodeBuild do usuário.

Tópicos

- Configurar ações de variáveis
- Visualizar variáveis de saída
- Exemplo: Usar variáveis em aprovações manuais
- Exemplo: use uma BranchName variável com variáveis de CodeBuild ambiente

Configurar ações de variáveis

Ao adicionar uma ação ao pipeline, você pode atribuir-lhe um namespace e configurá-lo para consumir variáveis de ações anteriores.

Configurar ações com variáveis (console)

Este exemplo cria um pipeline com uma ação CodeCommit de origem e uma ação de CodeBuild construção. A CodeBuild ação é configurada para consumir as variáveis produzidas pela CodeCommit ação.

Se o namespace não for especificado, as variáveis não estarão disponíveis para referência na configuração da ação. Quando você usa o console para criar um pipeline, o namespace de cada ação é gerado automaticamente.

Como criar um pipeline com variáveis

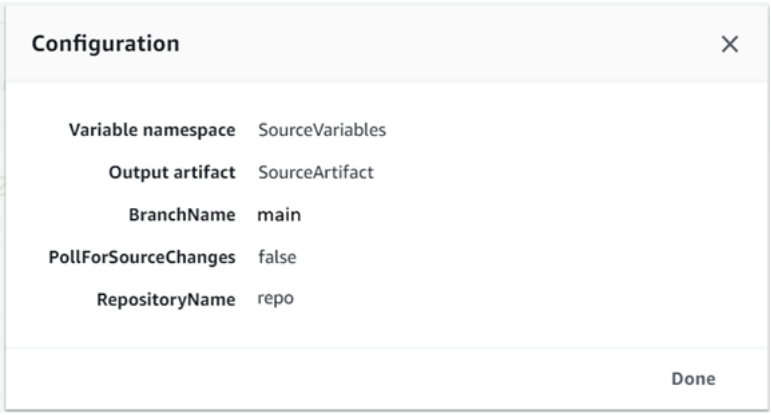
1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http:// console.aws.amazon. com/codesuite/codepipeline/home.
2. Selecione Criar pipeline. Insira um nome para o pipeline e escolha Next (Próximo).
3. Em Fonte, em Provedor, escolha CodeCommit. Escolha o CodeCommit repositório e a ramificação para a ação de origem e, em seguida, escolha Avançar.
4. Em Build, em Provider, escolha CodeBuild. Escolha um nome de projeto de CodeBuild construção existente ou escolha Criar projeto. Em Criar projeto de compilação, crie um projeto de compilação e escolha Retornar para CodePipeline.

   Em Environment variables (Variáveis de ambiente), escolha Add environment variables (Adicionar variáveis de ambiente). Por exemplo, insira o ID de execução com a sintaxe de variável `#{codepipeline.PipelineExecutionId}` e o ID de confirmação com a sintaxe da variável `#{SourceVariables.CommitId}`.

   > **Note**
   >
   > Você pode inserir sintaxe variável em qualquer campo de configuração de ação no assistente.

5. Escolha Criar.

6. Depois que o pipeline é criado, você pode visualizar o namespace criado pelo assistente. No pipeline, escolha o ícone do estágio cujo namespace você deseja visualizar. Neste exemplo, o namespace gerado automaticamente da ação de origem, `SourceVariables`, é exibido.



Para editar o namespace de uma ação existente

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Escolha o pipeline que você deseja editar e escolha Edit (Editar). No estágio de origem, escolha Edit stage (Editar estágio). Adicione a CodeCommit ação.
3. Em Edit action (Editar ação), visualize o campo Variable namespace (Namespace de variável) . Se a ação existente foi criada anteriormente ou sem usar o assistente, você deverá adicionar um namespace. Em Variable namespace (Namespace de variável), insira um nome de namespace e escolha Save (Salvar).

Como visualizar as variáveis de saída

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Depois que o pipeline for criado e executado com êxito, você poderá visualizar as variáveis na página Action execution details (Detalhes da execução da ação). Para mais informações, consulte Visualizar variáveis (console).

Configurar ações para variáveis (CLI)

Ao usar o create-pipeline comando para criar um pipeline ou o update-pipeline comando para editar um pipeline, você pode ver reference/use variáveis na configuração de uma ação.

Se o namespace não for especificado, as variáveis produzidas pela ação não estarão disponíveis para serem referenciadas em qualquer configuração de ação downstream.

Como configurar uma ação com um namespace

1. Siga as etapas em Criar um pipeline, estágios e ações para criar um pipeline usando a CLI. Inicie um arquivo de entrada para fornecer o comando create-pipeline com o parâmetro `--cli-input-json`. Na estrutura do pipeline, adicione o parâmetro `namespace` e especifique um nome, como `SourceVariables`.

```
. . .
{
          "inputArtifacts": [],
          "name": "Source",
          "region": "us-west-2",
          "namespace": "SourceVariables",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeCommit"
          },
          "outputArtifacts": [

. . .
```

2. Salve o arquivo com um nome como **`MyPipeline.json`**.
3. Em um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows), execute o comando create-pipeline e crie o pipeline.

   Invoque o arquivo que você criou quando executou o comando create-pipeline. Por exemplo:

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

Como configurar ações downstream para consumir variáveis

1. Edite um arquivo de entrada para fornecer o comando update-pipeline com o parâmetro `--cli-input-json`. Na ação downstream, adicione a variável à configuração dessa ação. Uma variável é composta de um namespace e uma chave, separados por um ponto. Por exemplo, para adicionar variáveis para o ID de execução do pipeline e o ID

de confirmação de origem, especifique o namespace `codepipeline` para a variável `#{codepipeline.PipelineExecutionId}`. Especifique o namespace `SourceVariables` para a variável `#{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. Salve o arquivo com um nome como **`MyPipeline.json`**.
3. Em um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows), execute o comando create-pipeline e crie o pipeline.

   Invoque o arquivo que você criou quando executou o comando create-pipeline. Por exemplo:

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

Visualizar variáveis de saída

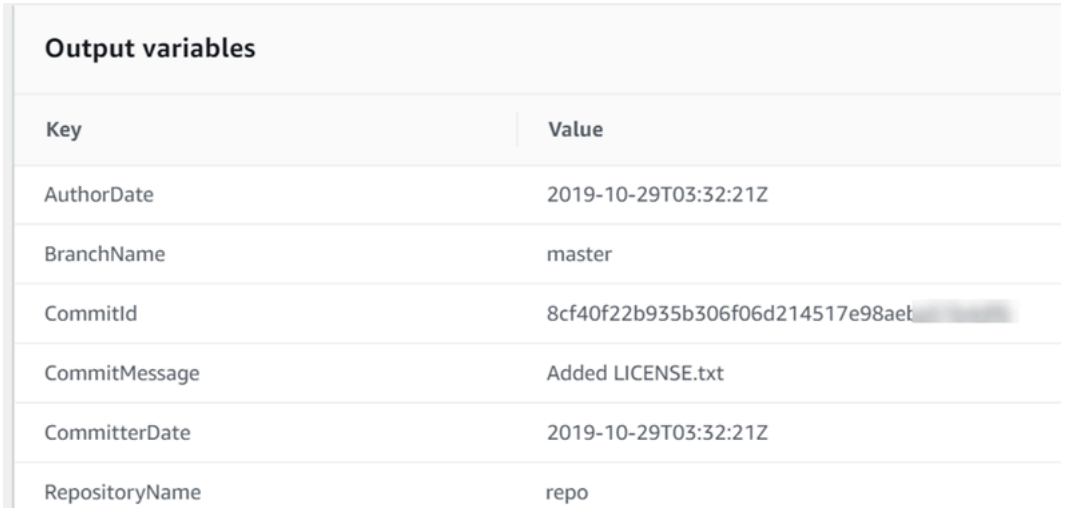
Você pode visualizar os detalhes da execução da ação para visualizar as variáveis dessa ação, específicas de cada execução.

Visualizar variáveis (console)

Você pode usar o console para visualizar variáveis de uma ação.

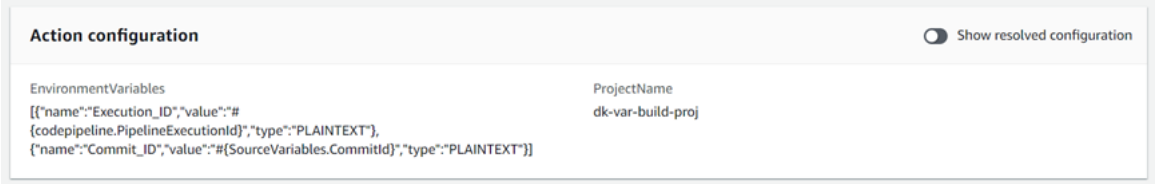
1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http:// console.aws.amazon. com/codesuite/codepipeline/home.

   Os nomes de todos os pipelines associados à sua AWS conta são exibidos.
2. Em Nome, selecione o nome do pipeline.
3. Selecione Visualizar histórico.
4. Depois que o pipeline for executado com êxito, você poderá exibir as variáveis produzidas pela ação de origem. Selecione Visualizar histórico. Escolha Origem na lista de ações para a execução do pipeline para ver os detalhes da execução da CodeCommit ação. Na tela de detalhes da ação, exiba as variáveis em Output variables (Variáveis de saída).

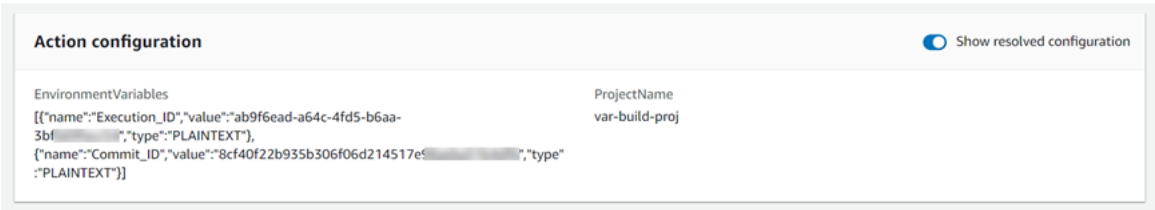


5. Depois que o pipeline for executado com êxito, você poderá exibir as variáveis consumidas pela ação de compilação. Selecione Visualizar histórico. Na lista de ações para a execução do pipeline, escolha Criar para visualizar os detalhes da execução da CodeBuild ação. Na página

de detalhes da ação, exiba as variáveis em Action configuration (Configuração de ação). O namespace gerado automaticamente é exibido.



Por padrão, a Action configuration (Configuração de ação) exibe a sintaxe da variável. Você pode escolher Show resolved configuration (Mostrar configuração resolvida) para alternar a lista para exibir os valores que foram produzidos durante a execução da ação



Visualizar variáveis (CLI)

Você pode usar o comando list-action-executions para exibir as variáveis de uma ação.

1. Use o seguinte comando:

```
aws codepipeline list-action-executions
```

A saída mostra o parâmetro `outputVariables` conforme mostrado aqui.

```
"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. Use o seguinte comando:

```
aws codepipeline get-pipeline --name <pipeline-name>
```

Na configuração da CodeBuild ação, você pode visualizar as variáveis:

```
{
    "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
        }
    ]
},
```

Exemplo: Usar variáveis em aprovações manuais

Quando você especifica um namespace para uma ação e essa ação produz variáveis de saída, você pode adicionar uma aprovação manual que exibe variáveis na mensagem de aprovação. Este exemplo mostra como adicionar sintaxe variável a uma mensagem de aprovação manual.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http:// console.aws.amazon. com/codesuite/codepipeline/home.

   Os nomes de todos os pipelines associados à sua AWS conta são exibidos. Escolha o pipeline ao qual deseja adicionar a aprovação.
2. Para editar o pipeline, escolha Edit (Editar). Adicione uma aprovação manual após a ação de origem. Em Action name (Nome da ação), insira o nome da ação de aprovação.
3. Em Action provider (Provedor de ação), escolha Manual approval (Aprovação manual).
4. Em URL para revisão, adicione a sintaxe da variável `CommitId` para sua CodeCommit URL. Certifique-se de usar o namespace atribuído à ação de origem. Por exemplo, a sintaxe da variável para uma ação CodeCommit com o namespace padrão `SourceVariables` é `#{SourceVariables.CommitId}`.

   Em Comentários, em `CommitMessage`, insira a mensagem de confirmação:

   ```
   Please approve this change. Commit message: #{SourceVariables.CommitMessage}
   ```
5. Depois que o pipeline for executado com êxito, você poderá exibir os valores das variáveis na mensagem de aprovação.

Exemplo: use uma BranchName variável com variáveis de CodeBuild ambiente

Ao adicionar uma CodeBuild ação ao seu pipeline, você pode usar variáveis de CodeBuild ambiente para referenciar uma variável de `BranchName` saída de uma ação de origem ascendente. Com uma variável de saída de uma ação em CodePipeline, você pode criar suas próprias variáveis de CodeBuild ambiente para uso em seus comandos de compilação.

Este exemplo mostra como adicionar a sintaxe da variável de saída de uma ação de GitHub origem a uma variável de CodeBuild ambiente. A sintaxe da variável de saída neste exemplo representa a

variável de saída da ação de GitHub origem para`BranchName`. Depois que a ação for executada com êxito, a variável será resolvida para mostrar o nome da GitHub ramificação.

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http:// console.aws.amazon. com/codesuite/codepipeline/home.

   Os nomes de todos os pipelines associados à sua AWS conta são exibidos. Escolha o pipeline ao qual deseja adicionar a aprovação.

2. Para editar o pipeline, escolha Edit (Editar). No estágio que contém sua CodeBuild ação, escolha Editar estágio.

3. Escolha o ícone para editar sua CodeBuild ação.

4. Na página Editar ação, em Variáveis de ambiente, insira o seguinte:

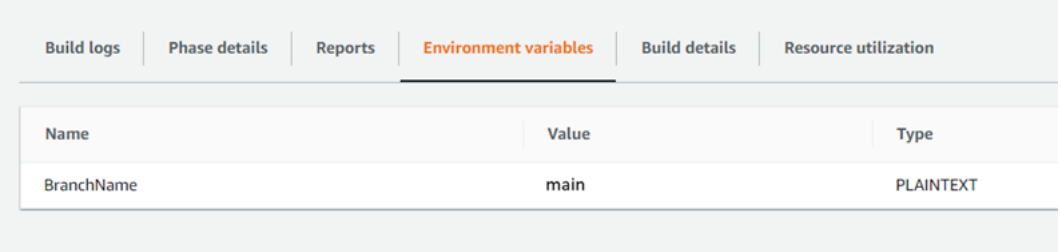
   - Em Nome, insira um nome para a variável de ambiente.
   - Em Valor, insira a sintaxe da variável de saída do pipeline, que inclui o namespace atribuído à sua ação de origem. Por exemplo, a sintaxe da variável de saída para uma GitHub ação com o namespace `SourceVariables` padrão é. `#{SourceVariables.BranchName}`
   - Em Tipo, escolha Texto sem formatação.

5. Depois que o pipeline for executado com êxito, você poderá ver como a variável de saída resolvida é o valor na variável de ambiente. Escolha uma das seguintes opções:

   - CodePipeline console: escolha seu funil e, em seguida, escolha Histórico. Escolha a execução mais recente do pipeline.
     - Em Linha do tempo, escolha o seletor de Origem. Essa é a ação de origem que gera variáveis GitHub de saída. Escolha Visualizar detalhes da execução. Em Variáveis de saída, visualize a lista de variáveis de saída geradas por essa ação.
     - Em Linha do tempo, escolha o seletor de Origem. Essa é a ação de compilação que especifica as variáveis de CodeBuild ambiente para seu projeto de compilação. Escolha Visualizar detalhes da execução. Em Configuração da ação, visualize suas variáveis de CodeBuild ambiente. Escolha Mostrar a configuração resolvida. O valor da variável de ambiente é a variável `BranchName` de saída resolvida da ação de GitHub origem. Neste exemplo, esse valor resolvido é `main`.

       Para obter mais informações, consulte Visualizar variáveis (console).
   - CodeBuild console: escolha seu projeto de compilação e escolha o link para sua execução de compilação. Em Variáveis de ambiente, sua variável de saída resolvida é o valor da variável

de CodeBuild ambiente. Neste exemplo, a variável de ambiente Name é `BranchName` e o Value é a variável `BranchName` de saída resolvida da ação de GitHub origem. Neste exemplo, o valor resolvido é `main`.



Trabalhando com transições de estágio no CodePipeline

Transições são links entre os estágios do pipeline que podem ser ativados ou desativados. Por padrão, as transições estão ativadas. Ao permitir novamente uma transição desativada, a revisão mais recente é executada nos estágios restantes do pipeline, a menos que mais de 30 dias tenham se passado. A execução do pipeline não recomeçará no caso de uma transição que foi desativada a mais de 30 dias, a menos que uma nova alteração seja detectada ou que você execute manualmente o pipeline outra vez.

Você pode usar o AWS CodePipeline console ou o AWS CLI para desativar ou ativar as transições entre os estágios em um pipeline.

> **Note**
>
> Você pode usar uma ação de aprovação para pausar a execução de um pipeline até que haja a aprovação manual para continuar. Para obter mais informações, consulte Adicionar uma ação de aprovação manual a uma fase.

Tópicos

- Habilitar ou desabilitar transições (console)
- Habilitar ou desabilitar transições (CLI)

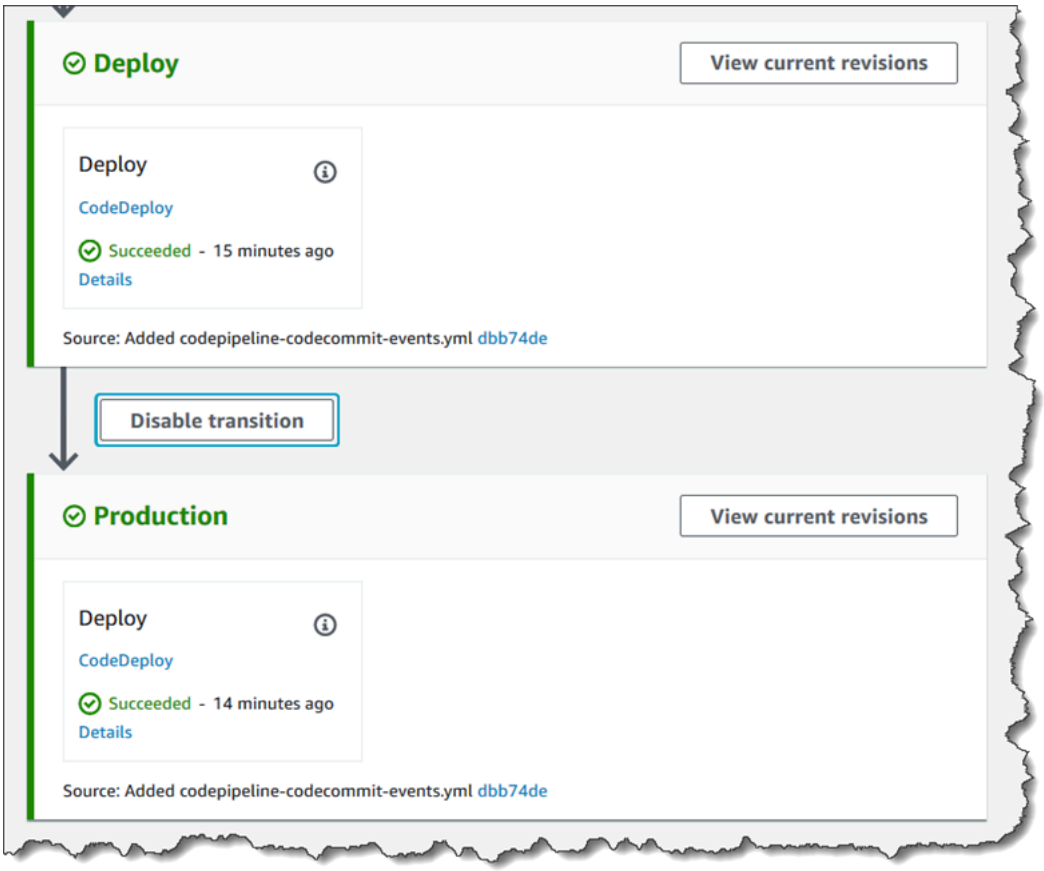
Habilitar ou desabilitar transições (console)

Para desabilitar ou permitir transições em um pipeline

1. Faça login no Console de gerenciamento da AWS e abra o CodePipeline console em http://console.aws.amazon.com/codesuite/codepipeline/home.

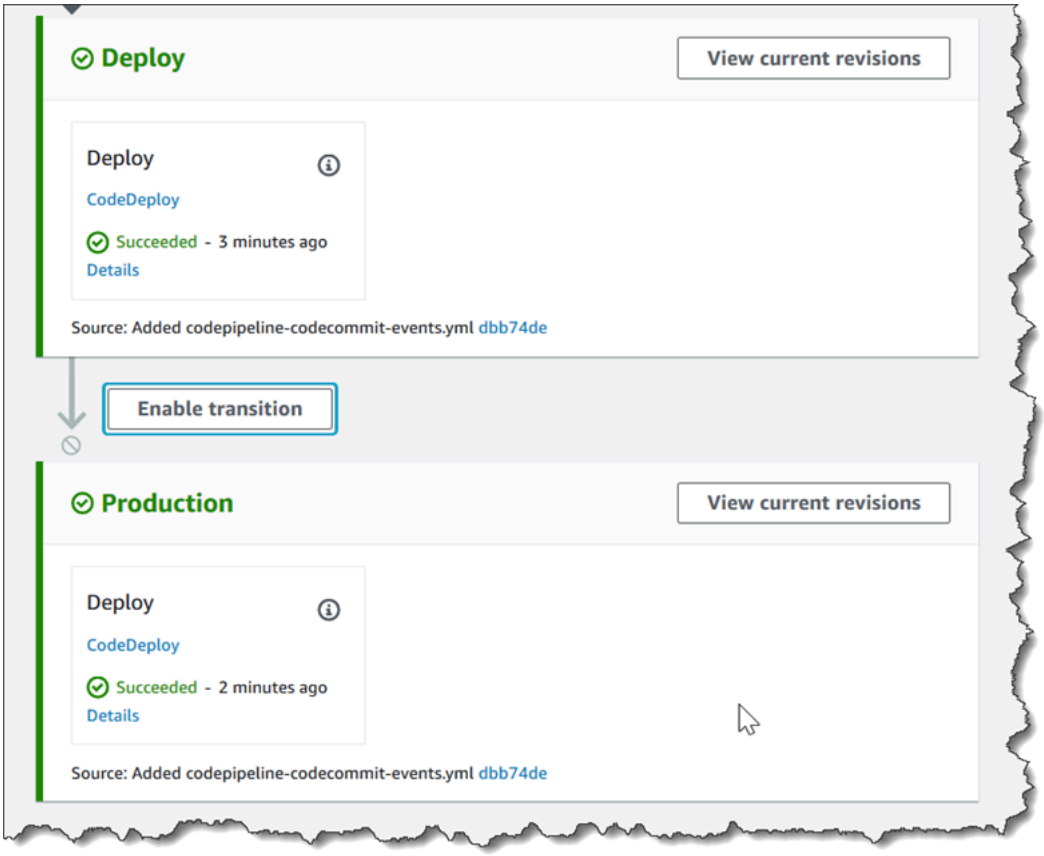
   Os nomes de todos os pipelines associados à sua conta da AWS são exibidos.
2. Em Nome, selecione o nome do pipeline para o qual deseja permitir ou desabilitar transições. Isso abre uma visão detalhada do pipeline, incluindo as transições entre os estágios do pipeline.
3. Encontre a seta após o último estágio que você deseja executar e depois selecione o botão ao lado dele. Por exemplo, no pipeline a seguir, se desejar que as ações no estágio Staging

(Estagiamento) sejam executadas, mas não as ações no estágio chamado Production (Produção), selecione o botão Disable transition (Desabilitar transição)entre estas duas etapas:



4. Na caixa de diálogo Desabilitar transição, insira um motivo para desabilitar a transição e depois selecione Desabilitar.

   O botão muda para mostrar que transições entre o estágio anterior e posterior à seta estão desabilitadas. Quaisquer revisões que já estavam em execução em estágios que vêm após a transição desabilitada continuarão ao longo do pipeline, mas quaisquer revisões subsequentes não continuarão além da transição desabilitada.



5. Escolha o botão Enable transition (Habilitar transição) próximo a seta. Na caixa de diálogo Permitir transição, selecione Permitir. O pipeline permite, imediatamente, a transição entre os dois estágios. Se todas as revisões foram executadas por meio dos estágios anteriores após a transição ter sido desativada, em alguns minutos o pipeline começa a executar a revisão mais recente através de estágios após a transição desativada anteriormente. O pipeline executa a revisão através de todos os estágios restantes no pipeline.

> **Note**
>
> Pode levar alguns segundos para que as alterações apareçam no CodePipeline console após você ativar a transição.

Habilitar ou desabilitar transições (CLI)

Para desativar a transição entre os estágios usando o AWS CLI, execute o disable-stage-transition comando. Para habilitar uma transição desabilitada, execute o comando enable-stage-transition.

Para desabilitar uma transição

1. Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use o AWS CLI para executar o comando disable-stage-transition, especificando o nome do pipeline, o nome do estágio no qual você deseja desativar as transições, o tipo de transição e o motivo pelo qual você está desativando as transições para esse estágio. Diferente do uso do console, você também deve especificar se estiver desabilitando transições para dentro do estágio (transições de entrada) ou para fora do estágio depois que todas as ações forem concluídas (de saída).

   Por exemplo, para desativar a transição para um estágio nomeado *Staging* em um pipeline chamado*MyFirstPipeline*, você digitaria um comando semelhante ao seguinte:

   ```
   aws codepipeline disable-stage-transition --pipeline-name MyFirstPipeline --stage-
   name Staging --transition-type Inbound --reason "My Reason"
   ```

   O comando não retorna nada.
2. Para verificar se a transição foi desativada, visualize o pipeline no console do CodePipeline ou execute o comando get-pipeline-state. Para obter mais informações, consulte Visualizar pipelines (console) e Visualizar detalhes e histórico do pipeline (CLI).

Para permitir uma transição

1. Abra um terminal (Linux, macOS ou Unix) ou prompt de comando (Windows) e use o AWS CLI para executar o comando enable-stage-transition, especificando o nome do pipeline, o nome do estágio para o qual você deseja ativar as transições e o tipo de transição.

   Por exemplo, para ativar a transição para um estágio nomeado *Staging* em um pipeline chamado*MyFirstPipeline*, você digitaria um comando semelhante ao seguinte:

   ```
   aws codepipeline enable-stage-transition --pipeline-name MyFirstPipeline --stage-
   name Staging  --transition-type Inbound
   ```

   O comando não retorna nada.
2. Para verificar se a transição foi ativada, visualize o pipeline no CodePipeline console ou execute o get-pipeline-state comando. Para obter mais informações, consulte Visualizar pipelines (console) e Visualizar detalhes e histórico do pipeline (CLI).

Monitorar pipelines

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e o desempenho do AWS CodePipeline. Você deve coletar dados de monitoramento de todas as partes de sua solução da AWS para facilitar a depuração de uma falha multipontos, caso ela ocorra. Antes de começar a monitorar, você deve criar um plano de monitoramento que responda às seguintes perguntas:

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?
- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento estão disponíveis para você usar?
- Quem realizará o monitoramento das tarefas?
- Quem deve ser notificado se algo der errado?

Você pode usar as seguintes ferramentas para monitorar os pipelines do CodePipeline e respectivos recursos:

- Eventos de barramento de eventos do EventBridge: você pode monitorar eventos do CodePipeline no EventBridge, que detecta alterações no status de execução do pipeline, do estágio ou da ação. O EventBridge encaminha esses dados para destinos como o AWS Lambda e o Amazon Simple Notification Service. Os eventos do EventBridge são os mesmos que aparecem no Amazon CloudWatch Events.
- Notificações para eventos do pipeline no console do Developer Tools: você pode monitorar eventos do CodePipeline com as notificações configuradas no console e, em seguida, criar um tópico e uma assinatura do Amazon Simple Notification Service. Para obter mais informações, consulte O que são notificações? no Guia do usuário do console do Developer Tools.
- AWS CloudTrail: use o CloudTrail para capturar as chamadas de API feitas por ou em nome do CodePipeline em sua conta da AWS e forneça os arquivos de log a um bucket do Amazon S3. Você pode optar por fazer com que o CloudWatch publique notificações do Amazon SNS quando novos arquivos de log forem fornecidos, a fim de que você possa executar ações rapidamente.
- Console e CLI: você pode usar o console do CodePipeline e a CLI para visualizar detalhes sobre o status de um pipeline ou da execução de um pipeline específico.

Tópicos

- Monitoramento de CodePipeline eventos
- Referência do bucket de espaço reservado para eventos
- Registrando chamadas de CodePipeline API com AWS CloudTrail
- CodePipeline CloudWatch métricas

Monitoramento de CodePipeline eventos

Você pode monitorar CodePipeline eventos em EventBridge, o que fornece um fluxo de dados em tempo real de seus próprios aplicativos, aplicativos de software como serviço (SaaS) e. Serviços da AWS EventBridge encaminha esses dados para destinos como o AWS Lambda Amazon Simple Notification Service. Esses eventos são os mesmos que aparecem no Amazon CloudWatch Events, que fornece um fluxo quase em tempo real de eventos do sistema que descrevem mudanças nos AWS recursos. Para obter mais informações, consulte O que é a Amazon EventBridge? no Guia do EventBridge usuário da Amazon.

> **Note**
>
> A Amazon EventBridge é a forma preferida de gerenciar seus eventos. Amazon CloudWatch Events e EventBridge são o mesmo serviço e API subjacentes, mas EventBridge oferecem mais recursos. As alterações que você fizer nos CloudWatch Eventos ou EventBridge aparecerão em cada console.

Os eventos são compostos por regras. Uma regra é configurada escolhendo o seguinte:

- Padrão de evento. Cada regra é expressa como um padrão de evento com a origem e o tipo de eventos a serem monitorados e os destinos do evento. Para monitorar eventos, você cria uma regra com o serviço que está monitorando como fonte do evento, como CodePipeline. Por exemplo, você pode criar uma regra com um padrão de evento que usa CodePipeline como fonte de eventos para acionar a regra quando há mudanças no estado de um pipeline, estágio ou ação.
- Destinos. A nova regra recebe um serviço selecionado como o destino do evento. Você pode configurar um serviço de destino para enviar notificações, capturar informações de status, tomar medidas corretivas, iniciar eventos ou realizar outras ações. Ao adicionar seu alvo, você também deve conceder permissões EventBridge para permitir que ele invoque o serviço de destino selecionado.

Cada tipo de evento de alteração de estado de execução emite notificações com um conteúdo de mensagem específico, em que:

- A entrada `version` inicial mostra o número da versão do evento.
- A entrada `version` no pipeline `detail` mostra o número de versão da estrutura do pipeline.
- A entrada `execution-id` no pipeline `detail` mostra o ID de execução do pipeline que provocou a alteração de estado. Consulte a chamada de API GetPipelineExecution na Referência da API do AWS CodePipeline.
- A entrada `pipeline-execution-attempt` mostra o número de tentativas, ou novas tentativas, do ID de execução específico.

CodePipeline reporta um evento para EventBridge sempre que o estado de um recurso em seu for Conta da AWS alterado. Os eventos são emitidos de forma garantida, pelo menos uma vez, para os seguintes recursos:

- Execuções de pipeline
- Execuções de estágio
- Execuções de ação

Os eventos são emitidos EventBridge com o padrão de eventos e o esquema detalhados acima. Para eventos processados, como eventos que você recebe por meio de notificações que configurou no console do Developer Tools, a mensagem do evento inclui campos de padrão de evento com alguma variação. Por exemplo, o campo `detail-type` é convertido em `detailType`. Para obter mais informações, consulte a chamada de PutEvents API no Amazon EventBridge API Reference.

Os exemplos a seguir mostram eventos para CodePipeline. Sempre que possível, cada exemplo mostra o esquema de um evento emitido junto com o esquema de um evento processado.

Tópicos

- Tipos de detalhes
- Pipeline-level eventos
- Stage-level eventos
- Action-level eventos
- Crie uma regra que envie uma notificação sobre um evento de pipeline

Tipos de detalhes

Ao configurar eventos a serem monitorados, você pode escolher o tipo de detalhe do evento.

Você pode configurar notificações para que sejam enviadas quando o estado muda para:

- Pipelines especificados ou todos os seus pipelines. Para controlar isso, use `"detail-type": "CodePipeline Pipeline Execution State Change"`.
- Estágios especificados ou todos os seus estágios, dentro de um pipeline especificado ou em todos os seus pipelines. Para controlar isso, use `"detail-type": "CodePipeline Stage Execution State Change"`.
- Ações especificadas ou todas as ações, dentro de um estágio especificado ou todos os estágios, dentro de um pipeline especificado ou todos os seus pipelines. Para controlar isso, use `"detail-type": "CodePipeline Action Execution State Change"`.

> **Note**
>
> Os eventos emitidos por EventBridge contêm o `detail-type` parâmetro, que é convertido em `detailType` quando os eventos são processados.

Tipo de detalhe	Estado	Description
CodePipeline Alteração do estado de execução do pipeline	CANCELED	A execução do pipeline foi cancelada porque a estrutura do pipeline foi atualizada.
	FAILED	A execução do pipeline não foi concluída com êxito.
	RESUMED	Uma execução malsucedida do pipeline foi retomada em resposta à chamada de API `RetryStageExecution` .
	STARTED	O pipeline está em execução no momento.
	STOPPED	O processo de interrupção é concluído e a execução do pipeline é interrompida.

Tipo de detalhe	Estado	Description
	STOPPING	A execução do pipeline está sendo interrompida devido a uma solicitação para interromper e aguardar ou interromper e abandonar a execução do pipeline.
	SUCCEEDED	A execução do pipeline foi concluída com êxito.
	SUPERSEDED	Embora a conclusão da execução desse pipeline estivesse programada para o estágio seguinte, uma nova execução se antecipou e prosseguiu pelo pipeline.
CodePipeline Alteração do estado de execução do estágio	CANCELED	O estágio foi cancelado porque a estrutura do pipeline foi atualizada.
	FAILED	O estágio não foi concluído com êxito.
	RESUMED	Um estágio malsucedido foi retomado em resposta à chamada de API `RetryStageExecution` .
	STARTED	O estágio está em execução no momento.
	STOPPED	O processo de interrupção é concluído e a execução do estágio é interrompida.
	STOPPING	A execução do estágio é interrompida devido a uma solicitação para interromper e aguardar ou interromper e abandonar a execução do pipeline.
	SUCCEEDED	O estágio foi concluído com êxito.
CodePipeline Alteração do estado de execução da ação	ABANDONADO	A ação é abandonada devido a uma solicitação para interromper e abandonar a execução do pipeline.
	CANCELED	A ação foi cancelada porque a estrutura do pipeline foi atualizada.
	FAILED	Para ações de aprovação, o estado FAILED significa que a ação foi rejeitada pelo revisor ou falhou devido a uma ação de configuração incorreta.

Tipo de detalhe	Estado	Description
	STARTED	A ação está em execução no momento.
	SUCCEEDED	A ação foi concluída com êxito.

Pipeline-level eventos

Pipeline-level eventos são emitidos quando há uma mudança de estado para a execução de um pipeline.

Tópicos

- Evento Pipeline INICIADO
- Evento Pipeline PARANDO
- Evento Pipeline BEM-SUCEDIDO
- Pipeline BEM-SUCEDIDO (exemplo com tags Git)
- Evento Pipeline COM FALHA
- Pipeline BEM-SUCEDIDO (exemplo com tags Git)

Evento Pipeline INICIADO

Quando a execução de um pipeline é iniciada, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo refere-se ao pipeline chamado `"myPipeline"` na região `us-east-1`. O campo `id` representa o ID do evento e o campo `account` representa o ID da conta em que o pipeline é criado.

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 Pipeline PARANDO

Quando a execução de um pipeline está sendo interrompida, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo refere-se ao pipeline chamado `myPipeline` na região `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 Pipeline BEM-SUCEDIDO

Quando a execução de um pipeline é bem-sucedida, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo refere-se ao pipeline chamado `myPipeline` na região `us-east-1`.

Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
```

```
    "time": "2020-01-24T22:03:44Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:codepipeline:us-east-1:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "SUCCEEDED",
        "version": 3.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-east-1",
    "source": "aws.codepipeline",
    "time": "2021-06-30T22:13:51Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "SUCCEEDED",
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {}
}
```

Pipeline BEM-SUCEDIDO (exemplo com tags Git)

Quando a execução de um pipeline tem um estágio que foi repetido e bem-sucedido, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo destina-se ao pipeline chamado `myPipeline` na região `eu-central-1` em que `execution-trigger` está configurado para as tags Git.

> **Note**
>
> O campo `execution-trigger` terá `tag-name` ou `branch-name`, dependendo do tipo de evento que acionou o pipeline.

```
{
    "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 Pipeline COM FALHA

Quando a execução de um pipeline falha, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo refere-se ao pipeline chamado "`myPipeline`" na região `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"
    }
```

Pipeline BEM-SUCEDIDO (exemplo com tags Git)

A menos que falhe no estágio de origem, para um pipeline configurado com gatilhos, ele emite um evento que envia notificações com o conteúdo a seguir. Este exemplo destina-se ao pipeline chamado `myPipeline` na região `eu-central-1` em que `execution-trigger` está configurado para as tags Git.

> **Note**
>
> O campo `execution-trigger` terá `tag-name` ou `branch-name`, dependendo do tipo de evento que acionou o pipeline.

Emitted event

```
{
    "version": "0",
```

```
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:55:43Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "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 eventos são emitidos quando há uma mudança de estado para a execução de um estágio.

Tópicos

- Evento Estágio INICIADO

- Evento Estágio PARANDO
- Evento Estágio PARADO
- Estágio RETOMADO após o evento de repetição de estágio

Evento Estágio INICIADO

Quando a execução de um estágio é iniciada, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo refere-se ao pipeline chamado `"myPipeline"` na região `us-east-1`, para o estágio `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 Estágio PARANDO

Quando a execução de um estágio está sendo interrompida, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo refere-se ao pipeline chamado `myPipeline` na região `us-west-2`, para o estágio `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 Estágio PARADO

Quando a execução de um estágio é interrompida, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo refere-se ao pipeline chamado `myPipeline` na região `us-west-2`, para o estágio `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
    }
```

```
}
```

Estágio RETOMADO após o evento de repetição de estágio

Quando a execução de um estágio é retomada e tem um estágio que foi repetido, ela emite um evento que envia notificações com o conteúdo a seguir.

Quando um estágio é repetido, o campo `stage-last-retry-attempt-time` é exibido, conforme mostrado no exemplo. O campo é exibido em todos os eventos do estágio se uma nova tentativa for realizada.

> **Note**
>
> O campo `stage-last-retry-attempt-time` estará presente em todos os eventos de estágio subsequentes após a repetição de um estágio.

```
{
    "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 eventos são emitidos quando há uma mudança de estado para a execução de uma ação.

Tópicos

- Evento Ação INICIADA
- Evento Ação BEM-SUCEDIDA
- Evento Ação COM FALHA
- Evento Ação ABANDONADA

Evento Ação INICIADA

Quando a execução de uma ação é iniciada, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo refere-se ao pipeline chamado `myPipeline` na região `us-east-1`, para a ação de implantação `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 Ação BEM-SUCEDIDA

Quando a execução de uma ação é bem-sucedida, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo refere-se ao pipeline chamado `"myPipeline"` na região `us-west-2`, para a ação de origem `"Source"`. Para esse tipo de evento, há dois campos `region` diferentes. O campo `region` do evento especifica a região do evento do pipeline. O campo `region` abaixo da seção `detail` especifica a região da ação.

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 Ação COM FALHA

Quando a execução de uma ação apresenta falha, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo refere-se ao pipeline chamado `"myPipeline"` na região `us-west-2`, para a ação `"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 Ação ABANDONADA

Quando a execução de uma ação é abandonada, ela emite um evento que envia notificações com o conteúdo a seguir. Este exemplo refere-se ao pipeline chamado "myPipeline" na região us-west-2, para a ação "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
    }
}
```

Crie uma regra que envie uma notificação sobre um evento de pipeline

Uma regra observa determinados eventos e os encaminha para os AWS alvos que você escolher. Você pode criar uma regra que executa uma AWS ação automaticamente quando outra AWS ação acontece ou uma regra que executa uma AWS ação regularmente em um cronograma definido.

Tópicos

- Enviar uma notificação quando o estado do pipeline é alterado (console)
- Enviar uma notificação quando o estado do pipeline é alterado (CLI)

Enviar uma notificação quando o estado do pipeline é alterado (console)

Essas etapas mostram como usar o EventBridge console para criar uma regra para enviar notificações de alterações no CodePipeline.

Para criar uma EventBridge regra que vise seu pipeline com uma fonte do Amazon S3

1. Abra o EventBridge console da Amazon em https://console.aws.amazon.com/events/.
2. No painel de navegação, escolha Regras. Deixe o barramento padrão selecionado ou escolha um barramento de eventos. Escolha Criar regra.
3. Em Nome, insira um nome para a regra.
4. Em Tipo de regra, escolha Regra com um padrão de eventos. Escolha Próximo.
5. Em Padrão de evento, escolha Serviços da AWS .
6. Na lista suspensa Event Type, escolha o nível de alteração de estado para a notificação.
   - Para uma regra que se aplica a eventos no nível do pipeline, escolha Mudança de estado de execução do CodePipeline pipeline.
   - Para uma regra que se aplica a eventos em nível de CodePipeline estágio, escolha Stage Execution State Change.
   - Para uma regra que se aplica a eventos em nível de ação, escolha Alteração do estado de execução da CodePipeline ação.
7. Especifique as alterações de estado às quais a regra se aplica:
   - Para uma regra que se aplique a todas as alterações de estado, escolha Any state.
   - Para uma regra que se aplique a algumas alterações de estado, escolha Specific state(s) e, em seguida, um ou mais valores de estado na lista.

8. Para padrões de evento mais detalhados do que os seletores permitem, você pode usar também a opção Editar padrão na janela Padrão de evento para designar um padrão de evento no formato JSON.

> **Note**
>
> Se não for especificado de outra forma, o padrão de evento será criado para todos os pipelines/stages /actions e states.

Para obter padrões de evento mais detalhados, você pode copiar e colar o exemplo de padrões de evento a seguir na janela Padrão de evento.

- Example

  Use esse exemplo de padrão de evento para capturar ações malsucedidas de implantação e compilação em todos os pipelines.

```
{
"source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Deploy", "Build"]
    }
  }
}
```

- Example

  Use esse exemplo de padrão de evento para capturar todas as ações de aprovação rejeitadas ou malsucedidas em todos os pipelines.

```
{
```

```
 "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Approval"]
    }
  }
}
```

- Example

  Use esse exemplo de padrão de evento para capturar todos os eventos de pipelines especificados.

```
{
"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. Escolha Próximo.
10. Em Tipos de destino, escolha Serviço da AWS .
11. Em Selecionar um alvo, escolha CodePipeline. Em ARN do pipeline, insira o ARN do pipeline a ser iniciado por esta regra.

> **Note**
>
> Para obter o ARN do pipeline, execute o comando get-pipeline. O ARN do pipeline é exibido na saída. Ele é construído neste formato:
> arn:aws:codepipeline::: *region account pipeline-name*
> Exemplo de ARN do pipeline:
> arn: aws: codepipeline: us-east- 2:80398 EXAMPLE:MyFirstPipeline

12. Para criar ou especificar uma função de serviço do IAM que conceda EventBridge permissões para invocar a meta associada à sua EventBridge regra (nesse caso, a meta é CodePipeline):
    - Escolha Criar uma nova função para esse recurso específico para criar uma função de serviço que dê EventBridge permissões para iniciar suas execuções de pipeline.
    - Escolha Usar função existente para inserir uma função de serviço que dê EventBridge permissões para iniciar suas execuções de pipeline.
13. Escolha Próximo.
14. Na página Tags, selecione Próximo.
15. Na página Revisar e criar, revise a configuração da regra. Se você estiver satisfeito com a regra, escolha Create rule.

Enviar uma notificação quando o estado do pipeline é alterado (CLI)

Essas etapas mostram como usar a CLI para criar uma regra de CloudWatch eventos para enviar notificações de alterações em CodePipeline.

Para usar o AWS CLI para criar uma regra, chame o put-rule comando, especificando:

- Um nome que identifique de forma exclusiva a regra que você está criando. Esse nome deve ser exclusivo em todos os funis que você cria CodePipeline associados à sua AWS conta.
- O padrão de evento para a origem e os campos detalhados usados pela regra. Para obter mais informações, consulte Amazon EventBridge e Event Patterns.

Para criar uma EventBridge regra com CodePipeline como fonte do evento

1. Chame o comando put-rulepara criar uma regra, especificando o padrão de evento. (Consulte as tabelas anteriores para ver os estados válidos.)

O exemplo de comando a seguir é usado --event-pattern para criar uma regra chamada "MyPipelineStateChanges" que emite o CloudWatch evento quando a execução de um pipeline falha para o pipeline chamado “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. Chame o comando put-targets e inclua os seguintes parâmetros:

   - O parâmetro `--rule` é usado com `rule_name` criado por meio de put-rule.
   - O parâmetro `--targets` é usado com o `Id` de lista do destino na lista de destinos e o `ARN` do tópico do Amazon SNS.

   O exemplo de comando a seguir especifica que, para a regra chamada `MyPipelineStateChanges`, o `Id` do destino é composto do número um, indicando que, em uma lista de destinos para a regra, esse é o destino 1. O exemplo de comando também especifica um exemplo de `ARN` para o tópico do Amazon SNS.

```
aws events put-targets --rule MyPipelineStateChanges --targets
 Id=1,Arn=arn:aws:sns:us-west-2:11111EXAMPLE:MyNotificationTopic
```

3. Adicione permissões EventBridge para usar o serviço de destino designado para invocar a notificação. Para obter mais informações, consulte Usando políticas baseadas em recursos para a Amazon. EventBridge

Referência do bucket de espaço reservado para eventos

Esta seção é apenas uma referência. Para obter informações sobre como criar um pipeline com recursos de detecção de eventos, consulte Ações de origem e métodos de detecção de alterações.

As ações de origem fornecidas pelo Amazon S3 e pelo CodeCommit usam recursos de detecção de alterações baseados em eventos para acionar o pipeline quando uma alteração é feita no bucket de origem ou no repositório. Esses recursos são as regras do CloudWatch Events configuradas para responder a eventos na origem do pipeline, como uma alteração de código no repositório do CodeCommit. Ao usar o CloudWatch Events para uma origem do Amazon S3, é necessário ativar o CloudTrail para que os eventos sejam registrados. O CloudTrail exige um bucket do S3 para o

qual possa enviar os resumos. É possível acessar os arquivos de log dos recursos do CloudWatch Events a partir do bucket personalizado, mas não é possível acessar os dados do bucket de espaço reservado.

- Se você usou a CLI ou o CloudFormation para configurar os recursos do CloudWatch Events, é possível encontrar os arquivos do CloudTrail no bucket especificado ao configurar o pipeline.
- Se você usou o console para configurar o pipeline com uma origem do S3, o console usa um bucket de espaço reservado do CloudTrail ao criar recursos do CloudWatch Events para você. Os resumos do CloudTrail são armazenados no bucket no Região da AWS em que o pipeline é criado.

É possível alterar a configuração se quiser usar um bucket diferente do bucket de espaço reservado.

> **Note**
>
> Os dados gravados nos buckets de espaço reservado do CloudTrail expiram automaticamente após um dia e não são retidos.

Para obter mais informações sobre como localizar e gerenciar os arquivos de log do CloudTrail, consulte Obter e visualizar os arquivos de log do CloudTrail.

Tópicos

- Nomes de bucket de espaço reservado para eventos por região

Nomes de bucket de espaço reservado para eventos por região

Esta tabela lista os nomes dos buckets de espaço reservado do S3 que contêm arquivos de log que rastreiam eventos de detecção de alteração para pipelines com ações de origem do Amazon S3.

Nome da região	Nome do bucket do espaço reservado	Identificador da região
Leste dos EUA (Ohio)	codepipeline-cloudtrail-placeholder-bucket-us-east-2	us-east-2
Leste dos EUA (Norte da Virgínia)	codepipeline-cloudtrail-placeholder-bucket-us-east-1	us-east-1

Nome da região	Nome do bucket do espaço reservado	Identificador da região
Oeste dos EUA (Norte da Califórnia)	codepipeline-cloudtrail-placeholder-bucket-us-west-1	us-west-1
Oeste dos EUA (Oregon)	codepipeline-cloudtrail-placeholder-bucket-us-west-2	us-west-2
Canadá (Central)	codepipeline-cloudtrail-placeholder-bucket-ca-central-1	ca-central-1
Europa (Frankfurt)	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 (Paris)	codepipeline-cloudtrail-placeholder-bucket-eu-west-3	eu-west-3
Europa (Estocolmo)	codepipeline-cloudtrail-placeholder-bucket-eu-north-1	eu-north-1
Ásia-Pacífico (Hong Kong)	codepipeline-cloudtrail-placeholder-bucket-ap-east-1	ap-east-1
Ásia-Pacífico (Hyderabad)	codepipeline-cloudtrail-placeholder-bucket-ap-south-2	ap-south-2
Ásia-Pacífico (Jacarta)	codepipeline-cloudtrail-placeholder-bucket-ap-southeast-3	ap-southeast-3

Nome da região	Nome do bucket do espaço reservado	Identificador da região
Ásia-Pacífico (Melbourne)	codepipeline-cloudtrail-placeholder-bucket-ap-southeast-4	ap-southeast-4
Ásia-Pacífico (Mumbai)	codepipeline-cloudtrail-placeholder-bucket-ap-south-1	ap-south-1
Ásia-Pacífico (Osaka)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-3-prod	ap-northeast-3
Ásia-Pacífico (Tóquio)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-1	ap-northeast-1
Ásia-Pacífico (Seul)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-2	ap-northeast-2
Ásia-Pacífico (Singapura)	codepipeline-cloudtrail-placeholder-bucket-ap-southeast-1	ap-southeast-1
Ásia-Pacífico (Sydney)	codepipeline-cloudtrail-placeholder-bucket-ap-southeast-2	ap-southeast-2
Ásia-Pacífico (Tóquio)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-1	ap-northeast-1
Canadá (Central)	codepipeline-cloudtrail-placeholder-bucket-ca-central-1	ca-central-1
Europa (Frankfurt)	codepipeline-cloudtrail-placeholder-bucket-eu-central-1	eu-central-1

Nome da região	Nome do bucket do espaço reservado	Identificador da região
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 (Milão)	codepipeline-cloudtrail-placeholder-bucket-eu-south-1	eu-south-1
Europa (Paris)	codepipeline-cloudtrail-placeholder-bucket-eu-west-3	eu-west-3
Europa (Espanha)	codepipeline-cloudtrail-placeholder-bucket-eu-south-2	eu-south-2
Europa (Estocolmo)	codepipeline-cloudtrail-placeholder-bucket-eu-north-1	eu-north-1
Europa (Zurique)*	codepipeline-cloudtrail-placeholder-bucket-eu-central-2	eu-central-2
Israel (Tel Aviv)	codepipeline-cloudtrail-placeholder-bucket-il-central-1	il-central-1
Oriente Médio (Bahrein)*	codepipeline-cloudtrail-placeholder-bucket-me-south-1	me-south-1
Oriente Médio (Emirados Árabes Unidos)	codepipeline-cloudtrail-placeholder-bucket-me-central-1	me-central-1
América do Sul (São Paulo)	codepipeline-cloudtrail-placeholder-bucket-sa-east-1	sa-east-1

Registrando chamadas de CodePipeline API com AWS CloudTrail

AWS CodePipeline é integrado a AWS CloudTrail, um serviço que fornece um registro das ações tomadas por um usuário, função ou usuário CodePipeline. AWS service (Serviço da AWS) CloudTrail captura todas as chamadas de API CodePipeline como eventos. As chamadas capturadas incluem chamadas do CodePipeline console e chamadas de código para as operações CodePipeline da API. Se você criar uma trilha, poderá habilitar a entrega contínua de CloudTrail eventos para um bucket do Amazon S3, incluindo eventos para CodePipeline. Se você não configurar uma trilha, ainda poderá ver os eventos mais recentes no CloudTrail console no Histórico de eventos. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita CodePipeline, o endereço IP a partir do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais.

Para saber mais sobre isso CloudTrail, consulte o Guia AWS CloudTrail do usuário.

CodePipeline informações em CloudTrail

CloudTrail é habilitado em você Conta da AWS quando você cria a conta. Quando a atividade ocorre em CodePipeline, essa atividade é registrada em um CloudTrail evento junto com outros AWS service (Serviço da AWS) eventos no histórico de eventos. Você pode visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Visualização de eventos com histórico de CloudTrail eventos.

Para um registro contínuo dos eventos em seu Conta da AWS, incluindo eventos para CodePipeline, crie uma trilha. Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela se aplica a todas as AWS regiões. A trilha registra eventos de todas as regiões na AWS partição e entrega os arquivos de log para o bucket do Amazon S3 que você especificar. Além disso, você pode configurar outros Serviços da AWS para analisar e agir mais detalhadamente com base nos dados do evento coletados nos CloudTrail registros. Para saber mais, consulte:

- Visão Geral para Criar uma Trilha
- CloudTrail Serviços e integrações compatíveis
- Configurando as notificações do Amazon SNS para CloudTrail
- Recebendo arquivos de CloudTrail log de várias regiões e recebendo arquivos de CloudTrail log de várias contas

Todas CodePipeline as ações são registradas CloudTrail e documentadas na Referência da CodePipeline API. Por exemplo, chamadas para o `CreatePipeline` `GetPipelineExecution` e `UpdatePipeline` ações geram entradas nos arquivos de CloudTrail log.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais raiz ou AWS Identity and Access Management (IAM).
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS service (Serviço da AWS).

Para obter mais informações, consulte Elemento userIdentity do CloudTrail .

Entendendo as entradas do arquivo de CodePipeline log

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log para um bucket do Amazon S3 que você especificar. CloudTrail os arquivos de log contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a ação solicitada, a data e a hora da ação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas de API públicas, portanto, eles não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de CloudTrail registro para um evento de atualização do pipeline, em que um canal chamado MyFirstPipeline foi editado pelo usuário nomeado JaneDoe-CodePipeline com a ID da conta 80398EXAMPLE. O usuário alterou o nome do estágio de origem de um pipeline de `Source` para `MySourceStage`. Como os elementos `requestParameters` e `responseElements` no log do CloudTrail têm a estrutura completa do pipeline editado, esses elementos foram abreviados no exemplo a seguir. Foi adicionada ênfase à parte `requestParameters` do pipeline em que a mudança ocorreu, ao número de versão anterior do pipeline e à parte `responseElements` que mostra o número de versão incrementado por 1. As partes editadas estão marcadas com reticências (...) para ilustrar em que partes mais dados são exibidos em uma entrada de log real.

```
{
 "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

Você pode usar métricas CloudWatch para medir CodePipeline pipelines. Você pode usar as seguintes métricas para medir a duração do pipeline e as falhas nas execuções de pipeline.

Você pode monitorar os pipelines em dois níveis:

Nível do pipeline

Essas métricas estão no nível do pipeline. Escolha Por pipeline em CloudWatch. Os pipelines disponíveis serão listados em CloudWatch.

AWS nível da conta

Essas métricas se destinam a todos os pipelines em uma conta. Para ver as métricas no nível da AWS conta, escolha Métricas da conta em CloudWatch.

Para obter mais informações sobre métricas de visualização, consulte Visualizar métricas disponíveis no Guia CloudWatch do usuário da Amazon.

Tópicos

- PipelineDuration
- FailedPipelineExecutions

PipelineDuration

A métrica `PipelineDuration` mede a duração da execução do pipeline. Essa métrica está disponível somente para o nível do pipeline.

Unidades: segundos

FailedPipelineExecutions

A métrica `FailedPipelineExecutions` mede o número das execuções de pipeline com falha. Essa métrica está disponível nos níveis de pipeline e de conta.

Essa métrica inclui uma contagem à parte para execuções de pipeline com uma nova tentativa de uma ação com falha. Em outras palavras, quando uma ação com falha é repetida em um pipeline, isso contará como uma métrica de execução do pipeline à parte. Por exemplo, para um pipeline com uma ação de origem do S3 em que a ação de origem do S3 falha na primeira vez, isso faz uma execução e, em seguida, a ação de origem é repetida e falha novamente. Para este exemplo, a métrica será a seguinte:

```
FailedPipelineExecutionAttempts: 2
```

Unidades: contagem

Solução de problemas CodePipeline

As informações a seguir podem ajudar a solucionar problemas comuns no AWS CodePipeline.

Tópicos

- Erro de pipeline: um pipeline configurado com AWS Elastic Beanstalk retorna uma mensagem de erro: “Falha na implantação. A função fornecida não tem permissões suficientes: Service:AmazonElasticLoadBalancing”
- Erro de implantação: um pipeline configurado com um AWS Elastic Beanstalk a ação de implantação trava em vez de falhar se a permissão "DescribeEvents" estiver ausente
- Erro de pipeline: uma ação de origem retorna a mensagem de permissões insuficientes: “Não foi possível acessar o nome do CodeCommit repositório. Verifique se a função do IAM do pipeline tem permissões suficientes para acessar o repositório"
- Erro de pipeline: uma ação de compilação ou de teste do Jenkins é executada por muito tempo e falha devido a falta de credenciais ou de permissões
- Erro de pipeline: um pipeline criado em um AWS Região usando um bucket criado em outro AWS A região retorna um "InternalError" com o código "JobFailed”
- Erro de implantação: um arquivo ZIP que contém um arquivo WAR é implantado com êxito no AWS Elastic Beanstalk, mas o URL do aplicativo relata um erro 404 não encontrado
- Os nomes de pasta de artefatos do pipeline parecem estar truncados
- Adicione CodeBuild GitClone permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com
- Adicionar CodeBuild GitClone permissões para ações CodeCommit de origem
- Erro de pipeline: uma implantação com a CodeDeployToECS ação retorna uma mensagem de erro: “Exceção ao tentar ler o arquivo de artefato de definição de tarefa de: nome do artefato de < origem” >
- GitHub (via aplicativo OAuth) ação de origem: a lista de repositórios mostra diferentes repositórios
- GitHub (via GitHub aplicativo) ação de origem: não é possível concluir a conexão com um repositório
- Erro do Amazon S3: o < ARN da CodePipeline função de serviço > está negando o acesso ao S3 para o bucket do S3 < BucketName >
- Pipelines com Amazon S3, Amazon ECR CodeCommit ou fonte não são mais iniciados automaticamente

- Erro de conexão ao se conectar a GitHub: “Ocorreu um problema, verifique se os cookies estão habilitados em seu navegador” ou “O proprietário de uma organização deve instalar o GitHub aplicativo”
- Pipelines com o modo de execução alterado para o modo QUEUED ou PARALLEL falham quando o limite de execução é atingido
- Ao alterar para os modos QUEUED ou SUPERSEDED, pipelines configurados no modo PARALLEL podem apresentar uma definição desatualizada.
- Os pipelines alterados do modo PARALLEL exibirão um modo de execução anterior
- Pipelines configurados com conexões que filtram gatilhos por caminhos de arquivos podem falhar ao iniciar durante a criação da ramificação.
- Pipelines com conexões que usam filtragem de gatilho por caminhos de arquivo podem não iniciar quando o limite de arquivos é atingido
- CodeCommit ou as revisões de origem do S3 no modo PARALELO podem não corresponder ao evento EventBridge
- A ação de implantação do EC2 falha com uma mensagem de erro Não existe tal arquivo
- A ação EKS Deploy falha com uma mensagem de erro de cluster inacessível
- O pipeline não é acionado para todas as ramificações quando várias ações de origem fazem referência ao mesmo repositório
- Precisa de ajuda com outro problema?

Erro de pipeline: um pipeline configurado com AWS Elastic Beanstalk retorna uma mensagem de erro: “Falha na implantação. A função fornecida não tem permissões suficientes: Service:AmazonElasticLoadBalancing”

Problema: a função de serviço de CodePipeline não tem permissões suficientes para AWS Elastic Beanstalk, incluindo, mas não se limita a, algumas operações no Elastic Load Balancing. A função de serviço do CodePipeline foi atualizada em 6 de agosto de 2015 para resolver esse problema. Os clientes que criaram a função de serviço antes dessa data devem modificar a declaração de política da função de serviço para adicionar as permissões necessárias.

Correções possíveis: a solução mais fácil é editar a declaração de política para sua função de serviço, conforme detalhado em Adicionar permissões à função CodePipeline de serviço.

Após aplicar a política editada, siga as etapas em Iniciar um pipeline manualmente para executar novamente e de maneira manual todos os pipelines que usam o Elastic Beanstalk.

Dependendo das necessidades de segurança, também é possível modificar as permissões de outras maneiras.

Erro de implantação: um pipeline configurado com um AWS Elastic Beanstalk a ação de implantação trava em vez de falhar se a permissão "DescribeEvents" estiver ausente

Problema: a função de serviço de CodePipeline deve incluir a `"elasticbeanstalk:DescribeEvents"` ação de qualquer pipeline que use AWS Elastic Beanstalk. Sem essa permissão, as ações de AWS Elastic Beanstalk implantação são interrompidas sem falhar ou indicar um erro. Se essa ação estiver ausente da sua função de serviço, CodePipeline não terá permissões para executar o estágio de implantação do pipeline AWS Elastic Beanstalk em seu nome.

Possíveis correções: revise sua função CodePipeline de serviço. Se a ação `"elasticbeanstalk:DescribeEvents"` estiver ausente, use as etapas em Adicionar permissões à função CodePipeline de serviço para adicioná-la usando o recurso Editar política no console do IAM.

Após aplicar a política editada, siga as etapas em Iniciar um pipeline manualmente para executar novamente e de maneira manual todos os pipelines que usam o Elastic Beanstalk.

Erro de pipeline: uma ação de origem retorna a mensagem de permissões insuficientes: “Não foi possível acessar o nome do CodeCommit **`repositório`**. Verifique se a função do IAM do pipeline tem permissões suficientes para acessar o repositório"

Problema: a função de serviço para CodePipeline não tem permissões suficientes CodeCommit e provavelmente foi criada antes da adição do suporte para o uso de CodeCommit repositórios em 18 de abril de 2016. Os clientes que criaram a função de serviço antes dessa data devem modificar a declaração de política da função de serviço para adicionar as permissões necessárias.

Possíveis correções: adicione as permissões necessárias CodeCommit à política da sua função de CodePipeline serviço. Para obter mais informações, consulte Adicionar permissões à função CodePipeline de serviço.

Erro de pipeline: uma ação de compilação ou de teste do Jenkins é executada por muito tempo e falha devido a falta de credenciais ou de permissões

Problema: Se o servidor Jenkins estiver instalado em uma instância do Amazon EC2, a instância pode não ter sido criada com uma função de instância que tenha as permissões necessárias. CodePipeline Caso você esteja usando um usuário do IAM em um servidor Jenkins, em uma instância on-premises ou em uma instância do Amazon EC2 criada sem perfil do IAM necessário, isso significa que o usuário do IAM não tem as permissões necessárias ou o servidor Jenkins não consegue acessar essas credenciais por meio do perfil configurado no servidor.

Correções possíveis: verifique se o perfil de instância do Amazon EC2 ou o usuário do IAM estão configurados com a política gerenciada `AWSCodePipelineCustomActionAccess` ou com as permissões equivalentes. Para obter mais informações, consulte AWS políticas gerenciadas para AWS CodePipeline.

Se você estiver usando um usuário do IAM, verifique se o AWS perfil configurado na instância usa o usuário do IAM configurado com as permissões corretas. Talvez seja necessário fornecer as credenciais de usuário do IAM que você configurou para integração entre o Jenkins e CodePipeline diretamente na interface do usuário do Jenkins. Essa não é uma melhor prática recomendada. Se precisar fazer isso, verifique se o servidor Jenkins é seguro a usa HTTPS em vez de HTTP.

Erro de pipeline: um pipeline criado em um AWS Região usando um bucket criado em outro AWS A região retorna um "InternalError" com o código "JobFailed”

Problema: o download de um artefato armazenado em um bucket do Amazon S3 falhará se o pipeline e o bucket forem criados em AWS regiões diferentes.

Possíveis correções: certifique-se de que o bucket do Amazon S3 onde seu artefato está armazenado esteja na mesma AWS região do pipeline que você criou.

Erro de implantação: um arquivo ZIP que contém um arquivo WAR é implantado com êxito no AWS Elastic Beanstalk, mas o URL do aplicativo relata um erro 404 não encontrado

Problema: um arquivo WAR é implantado com êxito em um ambiente AWS Elastic Beanstalk mas o URL do aplicativo resulta em um erro "404 Not Found".

Possíveis correções: AWS Elastic Beanstalk pode descompactar um arquivo ZIP, mas não um arquivo WAR contido em um arquivo ZIP. Em vez de especificar um arquivo WAR no arquivo `buildspec.yml`, especifique uma pasta que contém o conteúdo a ser implantado. Por exemplo:

```
version: 0.2

phases:
  post_build:
    commands:
      - mvn package
      - mv target/my-web-app ./
artifacts:
  files:
    - my-web-app/**/*
 discard-paths: yes
```

Para ver um exemplo, consulte Amostra do AWS Elastic Beanstalk para o CodeBuild.

Os nomes de pasta de artefatos do pipeline parecem estar truncados

Problema: quando você visualiza os nomes dos artefatos do pipeline em CodePipeline, os nomes parecem estar truncados. Isso pode fazer com que os nomes pareçam ser semelhantes ou não conter todo o nome do pipeline.

Explicação: CodePipeline trunca os nomes dos artefatos para garantir que o caminho completo do Amazon S3 não exceda os limites de tamanho da política ao CodePipeline gerar credenciais temporárias para funcionários.

Mesmo que o nome do artefato pareça estar truncado, CodePipeline mapeia para o compartimento de artefatos de uma forma que não seja afetada por artefatos com nomes truncados. O pipeline pode

funcionar normalmente. Isso não é um problema com a pasta ou os artefatos. Há um limite de 100 caracteres para nomes de pipelines. Embora o nome da pasta do artefato possa parecer reduzido, ele ainda é exclusivo para o pipeline.

Adicione CodeBuild GitClone permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com

Quando você usa uma ação AWS CodeConnections de origem e uma CodeBuild ação, há duas maneiras pelas quais o artefato de entrada pode ser passado para a compilação:

- Padrão: a ação de origem produz um arquivo zip que contém o código que foi CodeBuild baixado.
- Clone completo: o código-fonte pode ser baixado diretamente para o ambiente de compilação.

  O modo de clone completo permite que você interaja com o código-fonte como um repositório Git funcional. Para usar esse modo, você deve conceder permissões ao seu CodeBuild ambiente para usar a conexão.

Para adicionar permissões à sua política CodeBuild de função de serviço, você cria uma política gerenciada pelo cliente que anexa à sua função de CodeBuild serviço. As etapas a seguir criam uma política em que a `UseConnection` permissão é especificada no `action` campo, o ARN da conexão é especificado no `Resource` campo e o ID do repositório de origem é limitado por. `Condition`

Para usar o console para adicionar as UseConnection permissões

1. Para encontrar o ARN da conexão e o ID do repositório de origem do seu pipeline, abra seu pipeline, clique no ícone (i) na ação de origem e vá para a guia Entrada.

   Um exemplo de ARN de conexão é:

   ```
   arn:aws:codeconnections:eu-central-1:123456789123:connection/
   sample-1908-4932-9ecc-2ddacee15095
   ```

   Um exemplo do ID do repositório de origem:

   ```
   owner/test-app
   ```

Você adiciona o ARN da conexão e o ID do repositório à sua política de função de CodeBuild serviço.

2. Para encontrar sua função CodeBuild de serviço, escolha o projeto de compilação usado em seu pipeline e navegue até a guia Detalhes da compilação.
3. Escolha o link Service role (Função de serviço) . Isso abre o console do IAM, onde você pode adicionar uma nova política que concede acesso à sua conexão.
4. No console do IAM, escolha Adicionar permissões e, em seguida, escolha Criar política em linha.

   Use o seguinte exemplo de modelo de política. Adicione o ARN da conexão no `Resource` campo e o ID `codeconnections:FullRepositoryId` do repositório no `Condition` campo, conforme mostrado neste exemplo:

   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"
                }
            }
        }
    ]
}
```

   Use o `Condition` campo para definir ainda mais suas permissões de política com base em seus requisitos de especificação de compilação (consulte a documentação `CodeConnection` das condições).

   Na guia JSON, cole sua política.

5. Escolha Próximo. Insira um nome para a política (por exemplo, **`connection-permissions`**) e escolha Create policy (Criar política).

   Você verá a **`connection-permissions`** política anexada às políticas de permissões da sua função.

Adicionar CodeBuild GitClone permissões para ações CodeCommit de origem

Quando seu pipeline tem uma ação de CodeCommit origem, há duas maneiras de passar o artefato de entrada para a construção:

- Padrão — A ação de origem produz um arquivo zip que contém o código que foi CodeBuild baixado.
- Clone completo: o código-fonte pode ser obtido por download diretamente para o ambiente de compilação.

  A opção Clone completo permite que você interaja com o código-fonte como um repositório Git em funcionamento. Para usar esse modo, você deve adicionar permissões para que seu CodeBuild ambiente extraia do seu repositório.

Para adicionar permissões à sua política CodeBuild de função de serviço, você cria uma política gerenciada pelo cliente que anexa à sua função de CodeBuild serviço. As etapas a seguir criam uma política que especifica a permissão `codecommit:GitPull` no campo `action`.

Para usar o console para adicionar as GitPull permissões

1. Para encontrar sua função CodeBuild de serviço, abra o projeto de compilação usado em seu pipeline e navegue até a guia Detalhes da compilação.
2. Escolha o link Service role (Função de serviço) . Isso abre o console do IAM, no qual você pode adicionar uma nova política que concede acesso ao seu repositório.
3. No console do IAM, escolha Attach policies (Anexar políticas) e selecione Create policy (Criar política).
4. Na guia JSON, cole a política de exemplo a seguir.

```
{
```

```
    "Action": [
        "codecommit:GitPull"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
```

5. Selecione Revisar política. Insira um nome para a política (por exemplo, **`codecommit-gitpull`**) e escolha Create policy (Criar política).
6. Retorne à página onde você estava anexando permissões, atualize a lista de políticas e selecione a política que acabou de criar. Escolha Anexar políticas.

Erro de pipeline: uma implantação com a CodeDeployToECS ação retorna uma mensagem de erro: “Exceção ao tentar ler o arquivo de artefato de definição de tarefa de: nome do artefato de < origem” >

Problema:

O arquivo de definição da tarefa é um artefato necessário para a ação de CodePipeline implantação no Amazon ECS por meio de CodeDeploy (a `CodeDeployToECS` ação). O tamanho máximo do ZIP do artefato na ação de implantação `CodeDeployToECS` é de 3 MB. A seguinte mensagem de erro é retornada quando o arquivo não é encontrado ou o tamanho do artefato excede 3 MB:

Exception while trying to read the task definition artifact file from: <source artifact name> (Exceção ao tentar ler o arquivo de artefato de definição de tarefa de: <nome do artefato de origem>)

Possíveis correções: verifique se o arquivo de definição de tarefa está incluído como um artefato. Se o arquivo já existir, verifique se o tamanho compactado é menor que 3 MB.

GitHub (via aplicativo OAuth) ação de origem: a lista de repositórios mostra diferentes repositórios

Problema:

Depois de uma autorização bem-sucedida para uma ação GitHub (via aplicativo OAuth) no CodePipeline console, você pode escolher em uma lista de seus GitHub repositórios. Se a lista não

incluir os repositórios que você esperava ver, você poderá solucionar o problema da conta usada para autorização.

Possíveis correções: a lista de repositórios fornecida no CodePipeline console é baseada na GitHub organização à qual a conta autorizada pertence. Verifique se a conta que você está usando para autorizar GitHub é a conta associada à GitHub organização em que seu repositório foi criado.

GitHub (via GitHub aplicativo) ação de origem: não é possível concluir a conexão com um repositório

Problema:

Como uma conexão com um GitHub repositório usa o AWS Conector para GitHub, você precisa de permissões de proprietário da organização ou permissões de administrador no repositório para criar a conexão.

Possíveis correções: para obter informações sobre os níveis de permissão de um GitHub repositório, consulte https://docs.github.com/en/free-pro-team@ latest/github teams/permission /setting-up-and-managing-organizations-and- -levels-for-an-organization.

Erro do Amazon S3: o < ARN da CodePipeline função de serviço > está negando o acesso ao S3 para o bucket do S3 < BucketName >

Problema:

Enquanto estiver em andamento, a CodeCommit ação CodePipeline verifica se o bucket de artefatos do pipeline existe. Se a ação não tiver permissão para verificação, ocorre um `AccessDenied` erro no Amazon S3 e a seguinte mensagem de erro é exibida em: CodePipeline

CodePipeline a função de serviço “arn:aws:iam:*`AccountID`*: role/service -role/*`RoleID`*" está tendo o acesso do S3 negado para o bucket do S3 "" *`BucketName`*

Os CloudTrail registros da ação também registram o `AccessDenied` erro.

Possíveis correções: Faça o seguinte:

- Para a política anexada à sua função de CodePipeline serviço, adicione `s3:ListBucket` à lista de ações em sua política. Para obter instruções sobre como visualizar sua política de função de serviço, consulte Visualizar o ARN do pipeline e o ARN do perfil de serviço (console). Edite a declaração de política para seu perfil de serviço, conforme detalhado em Adicionar permissões à função CodePipeline de serviço.
- Para a política baseada em recursos anexada ao bucket de artefatos do Amazon S3 para seu pipeline, também chamada de política de compartimento de artefatos, adicione uma declaração para permitir que `s3:ListBucket` a permissão seja usada por sua função de serviço. CodePipeline

  Para adicionar sua política ao bucket de artefatos

  1. Siga as etapas em Visualizar o ARN do pipeline e o ARN do perfil de serviço (console) para escolher seu bucket de artefatos na página Configurações do pipeline e, em seguida, visualize-o no console do Amazon S3.
  2. Escolha Permissions (Permissões).
  3. Em Bucket policy (Política de bucket), escolha Edit (Editar).
  4. No campo de texto Política, insira uma nova política de bucket ou edite a política existente, conforme mostrado no exemplo a seguir. A política de bucket é um arquivo JSON; portanto, você deve inserir um JSON válido.

     O exemplo a seguir mostra uma declaração de política de bucket para um bucket de artefatos em que está *AROAEXAMPLEID* o exemplo de ID da função de serviço.

     ```
     {
         "Effect": "Allow",
         "Principal": "*",
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::BucketName",
         "Condition": {
             "StringLike": {
                 "aws:userid": "AROAEXAMPLEID:*"
             }
         }
     }
     ```

     O exemplo a seguir mostra a mesma declaração de política de bucket após a adição da permissão.

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 obter mais informações, consulte as etapas em https://aws.amazon.com/blogs/security/writing-iam-policies-how-to-grant-access-to-an-amazon-s3-bucket/.

5. Escolha Salvar.

Após aplicar a política editada, siga as etapas em Iniciar um pipeline manualmente para executar novamente e de maneira manual o seu pipeline.

Pipelines com Amazon S3, Amazon ECR CodeCommit ou fonte não são mais iniciados automaticamente

Problema:

Depois de fazer uma alteração nas configurações de uma ação que usa regras de eventos (EventBridgeou CloudWatch Eventos) para detecção de alterações, o console pode não detectar uma alteração em que os identificadores de gatilho de origem sejam semelhantes e tenham caracteres iniciais idênticos. Como a nova regra de evento não é criada pelo console, o pipeline não é mais iniciado automaticamente.

Um exemplo de uma pequena alteração no final do nome do parâmetro para CodeCommit seria alterar o nome da CodeCommit ramificação `MyTestBranch-1` para`MyTestBranch-2`. Como a alteração está no final do nome da ramificação, a regra de evento para a ação de origem pode não atualizar ou criar uma regra para as novas configurações de origem.

Isso se aplica às ações de origem que usam eventos CWE para detecção de alterações, da seguinte maneira:

Ação de origem	Parâmetros/identificadores de gatilho (console)
Amazon ECR	Nome do repositório Tag da imagem
Amazon S3	Bucket Chave de objeto do S3

Ação de origem	Parâmetros/identificadores de gatilho (console)
CodeCommit	Nome do repositório Nome da ramificação

Correções possíveis:

Execute um destes procedimentos:

- Altere as CodeCommit/S3/ECR configurações para que sejam feitas alterações na parte inicial do valor do parâmetro.

  Exemplo: altere o nome da filial de `release-branch` para `2nd-release-branch`. Evite alterações no fim do nome, como `release-branch-2`.
- Altere CodeCommit/S3/ECR as configurações de cada pipeline.

  Exemplo: altere o nome da filial de `myRepo/myBranch` para `myDeployRepo/myDeployBranch`. Evite alterações no fim do nome, como `myRepo/myBranch2`.
- Em vez do console, use a CLI ou CloudFormation crie e atualize suas regras de eventos de detecção de alterações. Para obter instruções sobre como criar regras de eventos para uma ação de origem do S3, consulte Conectando-se às ações de origem do Amazon S3 que usam EventBridge e AWS CloudTrail. Para obter instruções sobre como criar regras de eventos para uma ação do Amazon ECR, consulte Ações e recursos de origem do Amazon ECR EventBridge. Para obter instruções sobre como criar regras de eventos para uma CodeCommit ação, consulteCodeCommit ações de origem e EventBridge.

Após editar sua configuração de ação no console, aceite os recursos atualizados de detecção de alterações criados pelo console.

Erro de conexão ao se conectar a GitHub: “Ocorreu um problema, verifique se os cookies estão habilitados em seu navegador” ou “O proprietário de uma organização deve instalar o GitHub aplicativo”

Problema:

Para criar a conexão para uma ação GitHub de origem em CodePipeline, você deve ser o proprietário GitHub da organização. Para repositórios que não estão em uma organização, você deve ser o proprietário do repositório. Quando uma conexão é criada por alguém que não seja o proprietário da organização, é criada uma solicitação para o proprietário da organização e um dos seguintes erros é exibido:

Ocorreu um problema, verifique se os cookies estão habilitados em seu navegador

OU

O proprietário da organização deve instalar o GitHub aplicativo

Possíveis correções: Para repositórios em uma GitHub organização, o proprietário da organização deve criar a conexão com o GitHub repositório. Para repositórios que não estão em uma organização, você deve ser o proprietário do repositório.

Pipelines com o modo de execução alterado para o modo QUEUED ou PARALLEL falham quando o limite de execução é atingido

Problema: no modo Em FILA, um pipeline pode ter no máximo 50 execuções simultâneas. Quando esse limite é atingido, o pipeline falha sem uma mensagem de status.

Possíveis correções: realize a edição do modo de execução na definição do pipeline de forma isolada, sem combinar com outras alterações.

Para ter mais informações sobre o modo de execução QUEUED ou PARALLEL, consulte CodePipeline conceitos.

Ao alterar para os modos QUEUED ou SUPERSEDED, pipelines configurados no modo PARALLEL podem apresentar uma definição desatualizada.

Problema: ao modificar o modo de execução de pipelines configurados no modo PARALLEL para QUEUED ou SUPERSEDED, a definição do pipeline para o modo PARALLEL permanece desatualizada. A definição atualizada do pipeline ao atualizar o modo PARALLEL não é usada no modo SUPERSEDED ou QUEUED.

Possíveis correções: quando ajustar o modo de execução de pipelines no modo PARALLEL para QUEUED ou SUPERSEDED, não faça alterações simultâneas na definição do pipeline.

Para ter mais informações sobre o modo de execução QUEUED ou PARALLEL, consulte CodePipeline conceitos.

Os pipelines alterados do modo PARALLEL exibirão um modo de execução anterior

Problema: ao ajustar o modo de execução de pipelines no modo PARALLEL para QUEUED ou SUPERSEDED, o estado do pipeline não refletirá o modo atualizado como PARALLEL. Caso o pipeline seja alterado de PARALLEL para QUEUED ou SUPERSEDED, o estado no modo SUPERSEDED ou QUEUED corresponderá ao último estado registrado em um desses modos. Se o pipeline nunca tiver sido executado nesse modo antes, o estado estará vazio.

Possíveis correções: ao ajustar o modo de execução de pipelines no modo PARALLEL para QUEUED ou SUPERSEDED, esteja ciente de que o estado PARALLEL não será exibido no modo de execução.

Para ter mais informações sobre o modo de execução QUEUED ou PARALLEL, consulte CodePipeline conceitos.

Pipelines configurados com conexões que filtram gatilhos por caminhos de arquivos podem falhar ao iniciar durante a criação da ramificação.

Descrição: Para pipelines com ações de origem que usam conexões, como uma ação de BitBucket origem, você pode configurar um gatilho com uma configuração do Git que permite filtrar por caminhos de arquivo para iniciar seu pipeline. Em certos casos, para pipelines com acionadores filtrados em caminhos de arquivo, o pipeline pode não iniciar quando uma ramificação com um filtro de caminho de arquivo é criada pela primeira vez, pois isso não permite que a AWS CodeConnections conexão resolva os arquivos que foram alterados. Quando a configuração do Git para o gatilho estiver configurada para filtrar os caminhos dos arquivos, o pipeline não será iniciado quando a ramificação com o filtro tiver acabado de ser criada no repositório de origem. Para obter mais informações sobre a filtragem em caminhos de arquivo, consulteAdicionar gatilho com tipos de eventos code push ou pull request.

Resultado: por exemplo, pipelines CodePipeline que tenham um filtro de caminho de arquivo em uma ramificação “B” não serão acionados quando a ramificação “B” for criada. Se não houver filtros de caminho de arquivo, o pipeline ainda será iniciado.

Pipelines com conexões que usam filtragem de gatilho por caminhos de arquivo podem não iniciar quando o limite de arquivos é atingido

Descrição: Para pipelines com ações de origem que usam conexões, como uma ação de BitBucket origem, você pode configurar um gatilho com uma configuração do Git que permite filtrar por caminhos de arquivo para iniciar seu pipeline. CodePipeline recupera até os primeiros 100 arquivos; portanto, quando a configuração do Git para o gatilho é configurada para filtrar os caminhos dos arquivos, o pipeline pode não iniciar se houver mais de 100 arquivos. Para obter mais informações sobre a filtragem em caminhos de arquivo, consulteAdicionar gatilho com tipos de eventos code push ou pull request.

Resultado: por exemplo, se um diff contém 150 arquivos, CodePipeline examina os primeiros 100 arquivos (em nenhuma ordem específica) para verificar o filtro de caminho de arquivo especificado. Se o arquivo que corresponde ao filtro do caminho do arquivo não estiver entre os 100 arquivos recuperados pelo CodePipeline, o pipeline não será invocado.

CodeCommit ou as revisões de origem do S3 no modo PARALELO podem não corresponder ao evento EventBridge

Descrição: Para execuções de pipeline no modo PARALELO, uma execução pode começar com a alteração mais recente, como a confirmação do CodeCommit repositório, que pode não ser a mesma alteração do evento. EventBridge Em alguns casos, quando uma fração de segundo pode ocorrer entre confirmações ou tags de imagem que iniciam o pipeline, quando CodePipeline recebe o evento e inicia a execução, outra confirmação ou tag de imagem foi enviada CodePipeline (por exemplo, a CodeCommit ação) clonará a confirmação HEAD naquele momento.

Resultado: para pipelines no modo PARALLEL com uma fonte CodeCommit ou S3, independentemente da alteração que acionou a execução do pipeline, a ação de origem sempre clonará o HEAD no momento em que for iniciada. Por exemplo, em um pipeline no modo PARALLEL, o primeiro commit inicia a execução 1, enquanto o segundo será utilizado para a segunda execução do pipeline.

A ação de implantação do EC2 falha com uma mensagem de erro `Não existe tal arquivo`

Descrição: depois que a ação de implantação do EC2 descompacta os artefatos no diretório de destino nas instâncias, a ação executará o script. Se o script estiver no diretório de destino, mas a ação não conseguir executá-lo, a ação falhará nessa instância, e as instâncias restantes falharão na implantação.

Um erro semelhante às mensagens de erro a seguir é exibido nos logs de uma implantação na qual o diretório de destino seja `/home/ec2-user/deploy/` e o caminho do repositório de origem seja `myRepo/postScript.sh`.

- ```
  Instance i-0145a2d3f3EXAMPLE is FAILED on event AFTER_DEPLOY, message:
  ----------ERROR-------
  chmod: cannot access '/home/ec2-user/deploy/myRepo/postScript.sh': No such file or
   directory
  /var/lib/<path>/_script.sh: line 2: /home/ec2-user/deploy/myRepo/postScript.sh: No
   such file or directory
  failed to run commands: exit status 127
  ```
- ```
  Executing commands on instances i-0145a2d3f3EXAMPLE, SSM command id <ID>, commands:
   chmod u+x /home/ec2-user/deploy/script.sh
  ----------ERROR-------: No such file or directory
  ```

Resultado: a ação de implantação falha no pipeline.

Correções possíveis: para solucionar o problema, use as etapas a seguir.

1. Exiba os logs para verificar qual instância causou a falha no script.
2. Altere o diretório (`cd`) para o diretório de destino na instância. Teste a execução do script na instância.
3. No repositório de origem, edite o arquivo de script para remover eventuais comentários ou comandos que possam estar causando o problema.

A ação EKS Deploy falha com uma mensagem de erro de `cluster inacessível`

Descrição: depois que a ação de implantação do EKS for executada, a ação falhará com uma mensagem de erro `cluster unreachable`. A mensagem mostra um problema de acesso no cluster por causa de permissões não encontradas. Com base no tipo de arquivos (chart do Helm ou arquivos de manifesto do Kubernetes), a mensagem de erro é exibida da seguinte maneira:

- Em uma ação de implantação do EKS que esteja usando um chart do Helm, um erro semelhante à mensagem de erro a seguir é exibido.

```
error message:

helm upgrade --install my-release test-chart --wait
Error: Kubernetes cluster unreachable: the server has asked for the client to provide
 credentials
```

- Para uma ação de implantação do EKS que esteja usando arquivos de manifesto do Kubernetes, um erro semelhante à mensagem de erro a seguir é exibido.

```
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: a ação de implantação falha no pipeline.

Possíveis correções: se estiver usando uma função existente, a função de CodePipeline serviço deve ser atualizada com as permissões necessárias para usar a ação de implantação do EKS. Além disso, para permitir que a função de CodePipeline serviço acesse seu cluster, você deve adicionar uma entrada de acesso ao seu cluster e especificar a função de serviço para a entrada de acesso.

1. Verifique se a função CodePipeline de serviço tem as permissões necessárias para a ação de implantação do EKS. Para acessar a referência de permissões, consulte Permissões de política de perfil de serviço.
2. Adicione uma entrada de acesso ao seu cluster e especifique a função CodePipeline de serviço para o acesso. Para ver um exemplo, consulte Etapa 4: criar uma entrada de acesso para a função CodePipeline de serviço.

O pipeline não é acionado para todas as ramificações quando várias ações de origem fazem referência ao mesmo repositório

Problema: um pipeline tem várias ações de origem que usam conexões (como um Bitbucket ou GitLab conexão) GitHub, com cada ação de origem apontando para uma ramificação diferente do mesmo repositório. Somente uma das ramificações aciona o pipeline quando você envia alterações. A assinatura do webhook da conexão é registrada para a combinação de pipeline e repositório, não por ramificação. Como resultado, várias ações de origem direcionadas a diferentes ramificações do mesmo repositório em um único pipeline não são suportadas.

Possíveis correções: use um pipeline separado para cada ramificação que você deseja acionar de forma independente.

Precisa de ajuda com outro problema?

Veja estes outros recursos:

- Entrar em contato com o AWS Support.
- Faça uma pergunta no Fórum do CodePipeline.
- Solicite um aumento da cota. Para obter mais informações, consulte Cotas em AWS CodePipeline.

> **Note**
>
> Até duas semanas podem ser necessárias para o processamento das solicitações de aumento da cota.

Segurança em AWS CodePipeline

A segurança na nuvem AWS é a maior prioridade. Como AWS cliente, você se beneficia de data centers e arquiteturas de rede criados para atender aos requisitos das organizações mais sensíveis à segurança.

A segurança é uma responsabilidade compartilhada entre você AWS e você. O modelo de responsabilidade compartilhada descreve isso como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa AWS os serviços no Nuvem AWS. AWS também fornece serviços que você pode usar com segurança. Third-party auditores testam e verificam regularmente a eficácia de nossa segurança como parte dos Programas de Conformidade Programas de AWS de . Para saber mais sobre os programas de conformidade que se aplicam AWS CodePipeline, consulte AWS Serviços no escopo do programa de conformidade AWS .
- Segurança na nuvem — Sua responsabilidade é determinada pelo AWS serviço que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da sua empresa e as leis e normas aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar CodePipeline. Os tópicos a seguir mostram como configurar para atender CodePipeline aos seus objetivos de segurança e conformidade. Você também aprenderá a usar outros AWS serviços que ajudam a monitorar e proteger seus CodePipeline recursos.

Tópicos

- Proteção de dados em AWS CodePipeline
- Gerenciamento de identidade e acesso para AWS CodePipeline
- Registro e monitoramento em CodePipeline
- Validação de conformidade AWS CodePipeline
- Resiliência em AWS CodePipeline
- Segurança da infraestrutura em AWS CodePipeline
- Práticas recomendadas de segurança

Proteção de dados em AWS CodePipeline

O modelo de responsabilidade compartilhada da AWS se aplica à proteção de dados no . Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre o conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre privacidade de dados, consulte Perguntas frequentes sobre privacidade de dados . Para obter informações sobre proteção de dados na Europa, consulte o Centro de Regulamento Geral sobre a Proteção de Dados (RGPD).

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com Centro de Identidade do AWS IAM ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail. Para obter informações sobre o uso de CloudTrail trilhas para capturar AWS atividades, consulte Como trabalhar com CloudTrail trilhas no Guia AWS CloudTrail do usuário.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sensíveis armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para saber mais sobre os endpoints FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-3.

É altamente recomendável que nunca sejam colocadas informações confidenciais ou sensíveis, como endereços de e-mail de clientes, em tags ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com ou Serviços da AWS usa o console, a API ou AWS os SDKs. AWS CLI Quaisquer dados inseridos em tags ou em campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL

para um servidor externo, é fortemente recomendável que não sejam incluídas informações de credenciais no URL para validar a solicitação nesse servidor.

As melhores práticas de segurança a seguir também abordam a proteção de dados no CodePipeline:

- Configure a criptografia do lado do servidor para artefatos armazenados no Amazon S3 para CodePipeline
- Use AWS Secrets Manager para rastrear senhas de bancos de dados ou chaves de API de terceiros

Privacidade do tráfego entre redes

A Amazon VPC é uma AWS service (Serviço da AWS) que você pode usar para lançar AWS recursos em uma rede virtual (nuvem privada virtual) que você define. CodePipelinesuporta endpoints Amazon VPC baseados em AWS PrivateLink, uma AWS tecnologia que facilita a comunicação privada entre o Serviços da AWS uso de uma interface de rede elástica com endereços IP privados. Isso significa que você pode se conectar diretamente CodePipeline por meio de um endpoint privado em sua VPC, mantendo todo o tráfego dentro da VPC e da rede. AWS Anteriormente, era necessário que os aplicativos executados dentro de uma VPC estivessem conectados à Internet para que fosse estabelecida uma conexão com o CodePipeline. Com uma VPC, você possui controle sobre suas configurações de rede, como:

- Intervalo de endereços IP
- Sub-redes
- Tabelas de rotas e
- Gateways de rede.

Para conectar sua VPC a CodePipeline, você define uma interface para a qual VPC endpoint. CodePipeline Esse tipo de endpoint permite que você conecte a VPC aos Serviços da AWS. O endpoint fornece conectividade confiável e escalável CodePipeline sem a necessidade de um gateway de internet, instância de tradução de endereços de rede (NAT) ou conexão VPN. Para obter mais informações sobre como configurar uma VPC, consulte o Guia do usuário da VPC.

Criptografia em repouso

Os dados inseridos CodePipeline são criptografados em repouso usando AWS KMS keys. Os artefatos de código são armazenados em um bucket S3 de propriedade do cliente e criptografados

com a chave Chave gerenciada pela AWS ou com uma chave gerenciada pelo cliente. Para obter mais informações, consulte Configure a criptografia do lado do servidor para artefatos armazenados no Amazon S3 para CodePipeline.

Criptografia em trânsito

Toda comunicação entre serviços é criptografada em trânsito usando. SSL/TLS

Gerenciamento das chaves de criptografia

Se você escolher a opção padrão para criptografar artefatos de código, CodePipeline use o. Chave gerenciada pela AWS Você não pode alterar ou excluir isso Chave gerenciada pela AWS. Se você usar uma chave gerenciada pelo cliente AWS KMS para criptografar ou descriptografar artefatos no bucket do S3, poderá alterar ou alternar essa chave gerenciada pelo cliente conforme necessário.

> **Important**
>
> CodePipeline só oferece suporte a chaves KMS simétricas. Não use uma chave assimétrica do KMS para criptografar os dados no bucket do S3.

Configure a criptografia do lado do servidor para artefatos armazenados no Amazon S3 para CodePipeline

Há duas maneiras de configurar a criptografia do lado do servidor para artefatos do Amazon S3:

- CodePipeline cria um bucket de artefatos do S3 e usa como padrão Chave gerenciada pela AWS quando você cria um pipeline usando o assistente Create Pipeline. Chave gerenciada pela AWS É criptografado junto com os dados do objeto e gerenciado por AWS.
- É possível criar e gerenciar sua própria chave gerenciada pelo cliente.

> **Important**
>
> CodePipeline só oferece suporte a chaves KMS simétricas. Não use uma chave assimétrica do KMS para criptografar os dados no bucket do S3.

Se você estiver usando a chave padrão do S3, não será possível alterar ou excluir essa Chave gerenciada pela AWS. Se você estiver usando uma chave gerenciada pelo cliente AWS KMS para criptografar ou descriptografar artefatos no bucket do S3, poderá alterar ou alternar essa chave gerenciada pelo cliente conforme necessário.

O Amazon S3 comporta políticas de bucket que você pode usar se precisar de criptografia de servidor para todos os objetos que estão armazenados em seu bucket. Por exemplo, a política de bucket a seguir nega a permissão de upload object (`s3:PutObject`) a todos se a solicitação não incluir o `x-amz-server-side-encryption` cabeçalho que solicita a criptografia do lado do servidor com. 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 obter mais informações sobre criptografia do lado do servidor e AWS KMS, consulte Proteção de dados usando criptografia e Proteção de dados usando Server-Side criptografia do lado do servidor com chaves KMS armazenadas em (). AWS Key Management Service SSE-KMS

Para obter mais informações sobre AWS KMS, consulte o Guia do AWS Key Management Service desenvolvedor.

Tópicos

- Visualize o seu Chave gerenciada pela AWS
- Configure a criptografia do lado do servidor para buckets S3 usando CloudFormation ou o AWS CLI

Visualize o seu Chave gerenciada pela AWS

Quando você usa o assistente Create Pipeline (Criar pipeline) para criar seu primeiro pipeline, um bucket do S3 é criado na mesma região em que criou o pipeline. O bucket é usado para armazenar artefatos do pipeline. Quando um pipeline é executado, os artefatos são colocados e recuperados do bucket do S3. Por padrão, CodePipeline usa criptografia do lado do servidor com o AWS KMS uso do para Amazon Chave gerenciada pela AWS S3 (a chave). `aws/s3` Isso Chave gerenciada pela AWS é criado e armazenado em sua AWS conta. Quando os artefatos são recuperados do bucket do S3, CodePipeline usa o mesmo SSE-KMS processo para descriptografar o artefato.

Para ver informações sobre seu Chave gerenciada pela AWS

1. Faça login no Console de gerenciamento da AWS e abra o AWS KMS console.
2. Se uma página de boas-vindas for exibida, escolha Comece a usar agora.
3. No painel de navegação do serviço, escolha Chaves gerenciadas pela AWS .
4. Escolha a região do seu pipeline. Por exemplo, se o pipeline foi criado em `us-east-2`, verifique se o filtro está definido como Leste dos EUA (Ohio).

   Para obter mais informações sobre as regiões e os endpoints disponíveis CodePipeline, consulte AWS CodePipeline endpoints e cotas.
5. Na lista, escolha a chave com o alias usado para seu pipeline (por padrão, aws/s3). As informações básicas sobre a chave são exibidas.

Configure a criptografia do lado do servidor para buckets S3 usando CloudFormation ou o AWS CLI

Ao usar CloudFormation ou AWS CLI para criar um pipeline, você deve configurar a criptografia do lado do servidor manualmente. Use a política de bucket de amostra acima e crie a própria chave gerenciada pelo cliente. Você também pode usar suas próprias chaves em vez da Chave gerenciada pela AWS. Veja a seguir alguns motivos para escolher sua própria chave:

- Você deseja fazer a rotação das chaves em uma programação para atender aos requisitos comerciais ou de segurança de sua organização.
- Você deseja criar um pipeline que usa recursos associados a outra conta da AWS . Isso requer o uso de uma chave gerenciada pelo cliente. Para obter mais informações, consulte Crie um pipeline CodePipeline que use recursos de outro AWS account.

As melhores práticas criptográficas desencorajam a reutilização extensiva de chaves de criptografia. Uma prática recomendada é alternar sua chave em períodos regulares. Para criar um novo material criptográfico para suas AWS KMS chaves, você pode criar uma chave gerenciada pelo cliente e, em seguida, alterar seus aplicativos ou aliases para usar a nova chave gerenciada pelo cliente. Ou é possível habilitar a alternância automática de chaves para uma chave gerenciada pelo cliente existente.

Para alternar sua chave gerenciada pelo cliente, consulte Como alternar chaves.

> **Important**
>
> CodePipeline só oferece suporte a chaves KMS simétricas. Não use uma chave assimétrica do KMS para criptografar os dados no bucket do S3.

Use AWS Secrets Manager para rastrear senhas de bancos de dados ou chaves de API de terceiros

Recomendamos que você use AWS Secrets Manager para alternar, gerenciar e recuperar credenciais de banco de dados, chaves de API e outros segredos em todo o ciclo de vida. O Secrets Manager permite substituir credenciais codificadas, incluindo senhas, por uma chamada de API ao Secrets Manager para recuperar o segredo de modo programático. Para obter mais informações, consulte O que é o AWS Secrets Manager? no Guia do usuário do AWS Secrets Manager.

Para pipelines em que você passa parâmetros que são segredos (como credenciais OAuth) em um CloudFormation modelo, você deve incluir referências dinâmicas em seu modelo que acessem os segredos armazenados no Secrets Manager. Para obter padrão e exemplos de ID de referência, consulte Segredos do Secrets Manager no Guia do usuário do AWS CloudFormation . Para ver um exemplo que usa referências dinâmicas em um trecho de modelo para GitHub webhook em um pipeline, consulte Configuração de recursos de webhook.

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com gerenciamento de segredos.

- O Secrets Manager pode alternar as credenciais de banco de dados automaticamente, por exemplo, para alternar os segredos do Amazon RDS. Para obter mais informações, consulte Rotating Your AWS Secrets Manager Secrets Secrets no Guia do Usuário do AWS Secrets Manager.
- Para exibir instruções para adicionar referências dinâmicas do Secrets Manager aos modelos do CloudFormation , consulte https://aws.amazon.com/blogs/security/how-to-create-and-retrieve-secrets-managed-in-aws-secrets-manager-using-aws-cloudformation-template/.

Gerenciamento de identidade e acesso para AWS CodePipeline

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar CodePipeline os recursos. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Tópicos

- Público
- Autenticação com identidades
- Gerenciar o acesso usando políticas
- Como AWS CodePipeline funciona com o IAM
- AWS CodePipeline exemplos de políticas baseadas em identidade
- AWS CodePipeline exemplos de políticas baseadas em recursos

- Solução de problemas AWS CodePipeline identidade e acesso
- referência de permissões
- Gerenciar a função CodePipeline de serviço

Público

A forma como você usa AWS Identity and Access Management (IAM) difere com base na sua função:

- Usuário do serviço: solicite permissões ao seu administrador se você não conseguir acessar os atributos (consulte Solução de problemas AWS CodePipeline identidade e acesso).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte Como AWS CodePipeline funciona com o IAM)
- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte AWS CodePipeline exemplos de políticas baseadas em identidade)

Autenticação com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado como usuário do IAM ou assumindo uma função do IAM. Usuário raiz da conta da AWS

Você pode fazer login como uma identidade federada usando credenciais de uma fonte de identidade como Centro de Identidade do AWS IAM (IAM Identity Center), autenticação de login único ou credenciais. Google/Facebook Para ter mais informações sobre como fazer login, consulte Como fazer login em sua Conta da AWS no Guia do usuário do Início de Sessão da AWS .

Para acesso programático, AWS fornece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte AWS Signature Version 4 para solicitações de API no Guia do usuário do IAM.

Usuário raiz da conta da AWS

Ao criar um Conta da AWS, você começa com uma identidade de login chamada usuário Conta da AWS raiz que tem acesso completo a todos Serviços da AWS os recursos. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em Tarefas que exigem credenciais de usuário-raiz no Guia do usuário do IAM.

Usuários e grupos do IAM

Um usuário do IAM é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para obter mais informações, consulte Exigir que usuários humanos usem a federação com um provedor de identidade para acessar AWS usando credenciais temporárias no Guia do usuário do IAM.

Um grupo do IAM especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte Casos de uso de usuários do IAM no Guia do usuário do IAM.

Perfis do IAM

Uma perfil do IAM é uma identidade com permissões específicas que oferece credenciais temporárias. Você pode assumir uma função mudando de um usuário para uma função do IAM (console) ou chamando uma operação de AWS API AWS CLI ou. Para saber mais, consulte Métodos para assumir um perfil no Manual do usuário do IAM.

Os perfis do IAM são úteis para acesso de usuário federado, permissões de usuário do IAM temporárias, acesso entre contas, acesso entre serviços e aplicações em execução no Amazon EC2. Consulte mais informações em Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Gerenciar o acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política define permissões quando associada a uma identidade ou recurso. AWS avalia essas políticas quando um diretor faz uma solicitação. A maioria das políticas é armazenada AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

Identity-based políticas

Identity-based políticas são documentos de políticas de permissões JSON que você anexa a uma identidade (usuário, grupo ou função). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente no Guia do Usuário do IAM.

Identity-based as políticas podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do usuário do IAM.

Resource-based políticas

Resource-based políticas são documentos de política JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário especificar uma entidade principal em uma política baseada em recursos.

Resource-based as políticas são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais que podem definir o máximo de permissões concedidas por tipos de políticas mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de Controle de Serviços (SCPs): as SCPs especificam o número máximo de permissões para uma organização ou uma unidade organizacional no AWS Organizations. Para saber mais, consulte Políticas de controle de serviço no Guia do usuário do AWS Organizations .
- Políticas de controle de recursos (RCPs): definem o número máximo de permissões disponíveis para recursos em suas contas. Consulte mais informações em Resource control policies (RCPs) no Guia do usuário do AWS Organizations .

- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte Políticas de sessão no Guia do usuário do IAM.

Como AWS CodePipeline funciona com o IAM

Antes de usar o IAM para gerenciar o acesso CodePipeline, você deve entender quais recursos do IAM estão disponíveis para uso CodePipeline. Para ter uma visão de alto nível de como CodePipeline e de outras coisas Serviços da AWS funcionam com o IAM, consulte Serviços da AWS "Trabalho com o IAM" no Guia do usuário do IAM.

Tópicos

- CodePipeline políticas baseadas em identidade
- CodePipeline políticas baseadas em recursos
- Autorização baseada em CodePipeline tags
- CodePipeline Funções do IAM

CodePipeline políticas baseadas em identidade

Com as políticas baseadas em identidade do IAM, é possível especificar ações permitidas ou negadas e recursos, bem como as condições sob as quais as ações são permitidas ou negadas. O CodePipeline oferece suporte a ações, recursos e chaves de condição específicos. Para conhecer todos os elementos usados em uma política JSON, consulte Referência de elementos de política JSON do IAM no Guia do usuário do IAM.

Ações

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. Incluem ações em uma política para conceder permissões para executar a operação associada.

As ações políticas CodePipeline usam o seguinte prefixo antes da ação:`codepipeline:`.

Por exemplo, para conceder a alguém permissão para exibir os pipelines existentes na conta, inclua a ação `codepipeline:GetPipeline` em sua política. As declarações de política devem incluir

um `NotAction` elemento `Action` ou. CodePipeline define seu próprio conjunto de ações que descrevem as tarefas que você pode executar com esse serviço.

Para especificar várias ações em uma única instrução, separe-as com vírgulas, como segue:

```
"Action": [
      "codepipeline:action1",
      "codepipeline:action2"
```

Você também pode especificar várias ações usando caracteres curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Get`, inclua a seguinte ação:

```
"Action": "codepipeline:Get*"
```

Para ver uma lista de CodePipeline ações, consulte Ações definidas por AWS CodePipeline no Guia do usuário do IAM.

Recursos

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Para ações que não oferecem compatibilidade com permissões em nível de recurso, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

recursos e operações

No , o principal recurso é um pipeline. Em uma política, você usa um Nome de recurso da Amazon (ARN) para identificar o recurso ao qual a política de aplica. oferece suporte a outros recursos que podem ser usados com o recurso principal, como estágios, ações e ações personalizadas. Estes são chamados de sub-recursos. Esses recursos e sub-recursos têm Nomes de recurso da Amazon (ARNs) exclusivos associados a eles. Para obter mais informações sobre ARNs, consulte Amazon Resource Names (ARN) AWS service (Serviço da AWS) e namespaces no. Referência geral da

Amazon Web Services Para obter o ARN de pipeline associado ao seu pipeline, localize o ARN do pipeline em Configurações no console. Para obter mais informações, consulte Visualizar o ARN do pipeline e o ARN do perfil de serviço (console).

Tipo de recurso	Formato do ARN
Pipeline	arn: aws:codepipeline::: *region account pipeline-name*
Estágio	arn: aws:codepipeline:::/*regionaccountpipeline-name stage-name*
Ação	arn: aws:codepipeline:::///*regionaccountpipeline-name stage-name action-name*
Ação personalizada	arn:aws:codepipeline: ::actiont ype:///*regionaccountownercategoryproviderversion*
Todos os recursos	arn:aws:codepipeline:*
Todos os recursos do pertencentes à conta especificada na região especificada	arn: aws:codepipeline::: * *region account*

> **Note**
>
> A maioria dos serviços em AWS trata dois pontos (:) ou uma barra invertida (/) como o mesmo caractere em ARNs. No entanto, o usa uma correspondência exata nos padrões e regras dos eventos. Use os caracteres do ARN corretos ao criar padrões de evento para que eles correspondam à sintaxe ARN no pipeline a que você quer corresponder.

Em, há chamadas de API que oferecem suporte a permissões em nível de recurso. Resource-level as permissões indicam se uma chamada de API pode especificar o ARN de um recurso ou se a chamada de API só pode especificar todos os recursos usando o caractere curinga. Consulte referência de permissões para obter uma descrição detalhada das permissões em nível de recurso e uma lista das chamadas de CodePipeline API que oferecem suporte a permissões em nível de recurso.

Por exemplo, você pode indicar um pipeline específico (*myPipeline*) em sua declaração usando seu ARN da seguinte forma:

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:myPipeline"
```

Você também pode especificar todos os pipelines pertencentes a uma conta específica usando o caractere curinga (*), conforme o seguinte:

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:*"
```

Para especificar todos os recursos ou se uma ação de API específica não for compatível com ARNs, use o caractere curinga (*) no elemento `Resource`, da seguinte maneira:

```
"Resource": "*"
```

> **Note**
>
> Ao criar políticas do IAM, siga as dicas de segurança padrão de concessão de privilégio mínimo, ou seja, conceda apenas as permissões necessárias para executar uma tarefa. Se uma chamada de API der suporte a Nomes de recursos da Amazon (ARNs), ela dará suporte a permissões no nível do recurso e você não precisará usar o caractere curinga (*).

Algumas chamadas de API aceitam vários recursos (por exemplo,`GetPipeline`). Para especificar vários recursos em uma única declaração, separe seus ARNs com vírgulas, como se segue:

```
"Resource": ["arn1", "arn2"]
```

fornece um conjunto de operações para trabalhar com os recursos. Para ver uma lista das operações disponíveis, consulte referência de permissões.

Chaves de condição

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem agentes de condição, como “igual a”

ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

CodePipeline define seu próprio conjunto de chaves de condição e também oferece suporte ao uso de algumas chaves de condição globais. Para ver todas as chaves de condição AWS globais, consulte Chaves de contexto de condição AWS global no Guia do usuário do IAM.

Todas as ações do Amazon EC2 oferecem suporte às chaves de condição `aws:RequestedRegion` e `ec2:Region`. Para obter mais informações, consulte Exemplo: restrição de acesso a uma Região específica.

Para ver uma lista de chaves de CodePipeline condição, consulte Chaves de condição AWS CodePipeline no Guia do usuário do IAM. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte Ações definidas por AWS CodePipeline.

Exemplos

Para ver exemplos de políticas CodePipeline baseadas em identidade, consulte. AWS CodePipeline exemplos de políticas baseadas em identidade

CodePipeline políticas baseadas em recursos

CodePipeline não oferece suporte a políticas baseadas em recursos. No entanto, é fornecido um exemplo de política baseada em recursos para o serviço S3 relacionado ao CodePipeline .

Exemplos

Para ver exemplos de políticas CodePipeline baseadas em recursos, consulte AWS CodePipeline exemplos de políticas baseadas em recursos

Autorização baseada em CodePipeline tags

Você pode anexar tags a CodePipeline recursos ou passar tags em uma solicitação para CodePipeline. Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `codepipeline:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`. Para obter mais informações sobre recursos de marcação do CodePipeline , consulte Marcando atributos.

Para visualizar um exemplo de política baseada em identidade que visa limitar o acesso a um recurso baseado nas tags desse recurso, consulte Usando tags para controlar o acesso aos CodePipeline recursos.

CodePipeline Funções do IAM

Uma função do IAM é uma entidade na sua AWS conta que tem permissões específicas.

Usando credenciais temporárias com CodePipeline

É possível usar credenciais temporárias para fazer login com federação, assumir um perfil do IAM ou assumir um perfil entre contas. Você obtém credenciais de segurança temporárias chamando operações de AWS STS API, como AssumeRoleou GetFederationToken.

CodePipeline suporta o uso de credenciais temporárias.

Perfis de serviço

CodePipeline permite que um serviço assuma uma função de serviço em seu nome. O perfil permite que o serviço acesse recursos em outros serviços para concluir uma ação em seu nome. Os perfis de serviço aparecem em sua conta do IAM e são de propriedade da conta. Isso significa que um administrador do IAM pode alterar as permissões para esse perfil. Porém, fazer isso pode alterar a funcionalidade do serviço.

CodePipeline suporta funções de serviço.

AWS CodePipeline exemplos de políticas baseadas em identidade

Por padrão, os usuários e os perfis do IAM não têm permissão para criar ou modificar recursos do CodePipeline . Eles também não podem realizar tarefas usando a AWS API Console de gerenciamento da AWS AWS CLI, ou. Um administrador do IAM deve criar políticas do IAM que concedam aos usuários e perfis permissão para executarem operações de API específicas nos recursos especificados de que precisam. O administrador deve anexar essas políticas aos usuários ou grupos do IAM que exigem essas permissões.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documentos de política JSON, consulte Criar políticas na guia JSON no Manual do usuário do IAM.

Para saber como criar um pipeline que usa recursos de outra conta e para conferir exemplos de políticas relacionados, consulte Crie um pipeline CodePipeline que use recursos de outro AWS account.

Tópicos

- Práticas recomendadas de política
- Visualizar recursos no console
- Permitir que os usuários visualizem suas próprias permissões
- Identity-based exemplos de políticas (IAM)
- Usando tags para controlar o acesso aos CodePipeline recursos
- Permissões necessárias para usar o console
- Permissões necessárias para visualizar registros computacionais no console
- AWS políticas gerenciadas para AWS CodePipeline
- Exemplos de política gerenciada pelo cliente

Práticas recomendadas de política

Identity-based as políticas determinam se alguém pode criar, acessar ou excluir CodePipeline recursos em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para saber mais, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como CloudFormation.

Para saber mais, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.

- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para saber mais, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para saber mais sobre as práticas recomendadas do IAM, consulte Práticas recomendadas de segurança no IAM no Guia do usuário do IAM.

Visualizar recursos no console

O console exige a `ListRepositories` permissão para exibir uma lista de repositórios para sua AWS conta na AWS região em que você está conectado. O console também inclui uma função Go to resource (Acessar recurso) para realizar uma pesquisa por recursos que diferencia letras maiúsculas de minúsculas. Essa pesquisa é realizada em sua AWS conta na AWS região em que você está conectado. Os seguintes recursos são exibidos nos seguintes serviços:

- AWS CodeBuild: projetos de compilação
- AWS CodeCommit: repositórios
- AWS CodeDeploy: aplicativos
- AWS CodePipeline: pipelines

Para realizar essa pesquisa nos recursos em todos os serviços, você deve ter as seguintes permissões:

- AWS CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`

- CodePipeline: `ListPipelines`

Os resultados não serão retornados para os recursos de um serviço se você não tiver permissões para esse serviço. Mesmo se você tiver permissões para visualizar recursos, alguns recursos não serão retornados se houver um `Deny` explícito para visualizar esses recursos.

Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou 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 exemplos de políticas (IAM)

É possível anexar políticas a identidades do IAM. Por exemplo, você pode fazer o seguinte:

- Anexe uma política de permissões a um usuário ou grupo em sua conta — Para conceder a um usuário permissões para visualizar pipelines no console, você pode anexar uma política de permissões a um usuário ou grupo ao qual o usuário pertence.
- Anexar uma política de permissões a uma função: você pode anexar uma política de permissões baseada em identidade a um perfil do IAM para conceder permissões entre contas. Por exemplo, o administrador na Conta A pode criar uma função para conceder permissões entre contas para outra AWS conta (por exemplo, Conta B) ou da AWS service (Serviço da AWS) seguinte forma:
  1. Um administrador da Conta A cria uma função do IAM e anexa uma política de permissões à função que concede permissões em recursos da Conta A.
  2. Um administrador da Conta A anexa uma política de confiança à função identificando a Conta B como a entidade principal, que pode assumir a função.
  3. O administrador da Conta B pode então delegar permissões para assumir a função a qualquer usuário na Conta B. Isso permite que os usuários da Conta B criem ou acessem recursos na Conta A. O principal na política de confiança também pode ser um AWS service (Serviço da AWS) diretor se você quiser conceder AWS service (Serviço da AWS) permissões para assumir a função.

  Para obter mais informações sobre o uso do IAM para delegar permissões, consulte Gerenciamento de acesso no Guia do usuário do IAM.

O exemplo a seguir mostra um exemplo de política de permissões que permite que um usuário desabilite e habilite transições entre todos os estágios no pipeline `MyFirstPipeline` na `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/*"
      ]
    }
  ]
}
```

O exemplo a seguir mostra uma política na conta 111222333444 que permite que os usuários visualizem, mas não alterem, o pipeline nomeado no console. `MyFirstPipeline` Essa política é baseada na política gerenciada `AWSCodePipeline_ReadOnlyAccess`, mas, como ela é específica ao pipeline `MyFirstPipeline`, a política gerenciada não pode ser usada diretamente. Se você não desejar restringir a política a um pipeline específico, leve em consideração usar uma das políticas gerenciadas criadas e mantidas pelo . Para obter mais informações, consulte Como trabalhar com políticas gerenciadas. Você deve anexar essa política a um perfil do IAM criado para acesso, por exemplo, a um perfil chamado `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*"
        }
      }
    }
  ]
}
```

Após criar essa política, crie o perfil do IAM na conta 111222333444 e anexe a política a esse perfil. Nas relações de confiança da função, você deve adicionar a AWS conta que assumirá essa função. O exemplo a seguir mostra uma política que permite que os usuários da *111111111111* AWS conta assumam funções definidas na conta 111222333444:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

O exemplo a seguir mostra uma política criada na *111111111111* AWS conta que permite que os usuários assumam a função nomeada *CrossAccountPipelineViewers* na conta 111222333444:

JSON

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

```
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource":
 "arn:aws:iam::111222333444:role/CrossAccountPipelineViewers"
        }
    ]
}
```

Você pode criar políticas do IAM para restringir as chamadas e os recursos a que os usuários em sua conta têm acesso e anexar essas políticas ao seu usuário administrativo. Para obter mais informações sobre como criar funções do IAM e explorar exemplos de declarações de política do IAM para, consulteExemplos de política gerenciada pelo cliente.

Usando tags para controlar o acesso aos CodePipeline recursos

As condições nas declarações de política do IAM fazem parte da sintaxe que você usa para especificar as permissões para os recursos exigidos pelas CodePipeline ações. O uso de tags em condições é uma forma de controlar o acesso a recursos e solicitações. Para obter informações sobre a marcação de CodePipeline recursos, consulteMarcando atributos. Este tópico discute o controle de acesso com base em tags.

Ao criar políticas do IAM, você pode definir permissões granulares concedendo acesso a recursos específicos. À medida que o número de recursos que você gerencia aumenta, essa tarefa se torna mais difícil. Atribuir etiquetas a recursos e usá-las em condições de declaração de política pode facilitar essa tarefa. Você concede acesso em massa a qualquer recurso utilizando determinada etiqueta. Depois, você a aplica repetidamente a recursos relevantes durante a criação ou posteriormente.

As tags podem ser anexadas ao recurso ou passadas na solicitação para serviços compatíveis com tags. Em CodePipeline, os recursos podem ter tags e algumas ações podem incluir tags. Ao criar uma política do IAM, você poderá usar chaves de condição de tag para controlar:

- Quais usuários podem executar ações em um recurso de pipeline, com base nas tags que o recurso já tem.
- Quais tags podem ser transmitidas na solicitação de uma ação.
- Se chaves de tags específicas podem ser usadas em uma solicitação.

Operadores de condição de string permitem que você construa elementos `Condition` que restringem o acesso com base na comparação de uma chave a um valor de string. Você pode adicionar `IfExists` ao final de qualquer nome de operador de condição, exceto a condição Null. Isso é feito para dizer "Se a chave de política estiver presente no contexto da solicitação, processar a chave conforme especificado na política. Se a chave não estiver presente, avalie o elemento da condição como verdadeiro." Por exemplo, você pode usar `StringEqualsIfExists` para restringir por chaves de condição que podem não estar presentes em outros tipos de recursos.

Para obter a sintaxe e a semântica completas das chaves de condição de tag, consulte Como controlar o acesso usando tags. Para obter mais informações sobre as chaves de condição, veja os recursos a seguir. Os exemplos CodePipeline de políticas nesta seção se alinham às seguintes informações sobre chaves de condição e as expandem com exemplos de nuances, CodePipeline como aninhamento de recursos.

- Operadores de condição de strings
- Serviços da AWS que funcionam com o IAM
- Sintaxe de SCP
- Elementos de política JSON do IAM: Condition
- leis: RequestTag/tag-key
- Chaves de condição para CodePipeline

Os exemplos a seguir demonstram como especificar condições de tag em políticas para usuários do CodePipeline .

Example 1: Limitar ações com base em tags na solicitação

A política de usuário `AWSCodePipeline_FullAccess` gerenciado dá aos usuários permissão ilimitada para realizar qualquer CodePipeline ação em qualquer recurso.

A política a seguir limita esse poder e nega a usuários não autorizados a permissão para criar pipelines quando tags específicas estão listadas na solicitação. Para fazer isso, ela negará a ação `CreatePipeline` se a solicitação especificar uma tag chamada `Project` com um dos valores `ProjectA` ou `ProjectB`. (A `aws:RequestTag` chave de condição é usada para controlar quais tags podem ser transmitidas em uma solicitação do IAM.)

No exemplo a seguir, a intenção da política é negar a usuários não autorizados a permissão para criar um pipeline com os valores de tag especificados. No entanto, a criação de um pipeline requer

o acesso a recursos além do próprio pipeline (por exemplo, ações e estágios do pipeline). Como o `'Resource'` especificado na política é `'*'`, a política é avaliada com base em cada recurso que tem um ARN e é criada quando o pipeline está sendo criado. Esses recursos adicionais não têm a chave de condição de tag; portanto, a verificação `StringEquals` falha e o usuário não recebe a permissão para criar nenhum pipeline. Para resolver isso, use o operador de condição `StringEqualsIfExists`. Dessa forma, o teste só acontece se a chave de condição existir.

Você pode entender isso da seguinte maneira: "Se o recurso que está sendo verificado tiver uma chave de condição de tag `"RequestTag/Project"`, permita a ação apenas se o valor de chave começar com `projectA`. Se o recurso que está sendo verificado não tiver essa chave de condição, não se preocupe com isso."

Além disso, a política impede que esses usuários não autorizados interfiram nos recursos usando a chave de condição `aws:TagKeys` para não permitir que ações de modificação de tag incluam esses mesmos valores de tag. O administrador de um cliente deve anexar essa política do IAM a usuários administrativos não autorizados, além da política de usuário gerenciada.

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 ações de marcação com base em tags de recursos

A política de usuário `AWSCodePipeline_FullAccess` gerenciado dá aos usuários permissão ilimitada para realizar qualquer CodePipeline ação em qualquer recurso.

A seguinte política limita esse poder e nega a usuários não autorizados permissão para realizar ações em pipelines específicos do projeto. Para fazer isso, ela negará ações específicas se o recurso tiver uma tag denominada `Project` com um dos valores `ProjectA` ou `ProjectB`. (A chave de condição `aws:ResourceTag` é usada para controlar o acesso a recursos com base nas tags desses recursos.) O administrador de um cliente deve anexar essa política do IAM a usuários não autorizados do IAM, além da política de usuário gerenciada.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": ["ProjectA", "ProjectB"]
        }
      }
    }
  ]
}
```

Example 3: Permitir ações com base em tags na solicitação

A política a seguir concede aos usuários permissão para criar pipelines de desenvolvimento no CodePipeline.

Para fazer isso, ela permitirá as ações `CreatePipeline` e `TagResource` se a solicitação especificar uma tag denominada `Project` com o valor `ProjectA`. Em outras palavras, a única chave de tag que pode ser especificada é `Project`, e seu valor deve ser `ProjectA`.

A chave de condição `aws:RequestTag` é usada para controlar quais tags podem ser transmitidas em uma solicitação do IAM. A `aws:TagKeys` condição garante que a chave de tag faça diferenciação de letras maiúsculas e minúsculas. Essa política é útil para usuários ou perfis que não têm a política de usuário gerenciada `AWSCodePipeline_FullAccess` anexada. A política gerenciada dá aos usuários permissão ilimitada para realizar qualquer CodePipeline ação em qualquer 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 ações de marcação com base em tags de recursos

A política de usuário `AWSCodePipeline_FullAccess` gerenciado dá aos usuários permissão ilimitada para realizar qualquer CodePipeline ação em qualquer recurso.

A seguinte política limita esse poder e nega a usuários não autorizados permissão para realizar ações em pipelines específicos do projeto. Para fazer isso, ela negará ações específicas se o recurso tiver uma tag denominada `Project` com um dos valores `ProjectA` ou `ProjectB`.

Além disso, a política impede que esses usuários não autorizados interfiram nos recursos, usando a chave de condição `aws:TagKeys` para não permitir que ações de modificação de tag removam completamente a tag `Project`. O administrador de um cliente deve anexar essa política do IAM a usuários ou perfis não autorizados, além da política de usuário gerenciada.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

Permissões necessárias para usar o console

Para usar o no console do , é necessário ter um conjunto mínimo de permissões dos seguintes serviços:

- AWS Identity and Access Management

- Amazon Simple Storage Service

Essas permissões permitem que você descreva outros AWS recursos para sua AWS conta.

Dependendo dos outros serviços que você incorpora aos pipelines, podem ser necessárias permissões de um ou mais dos seguintes itens:

- AWS CodeCommit
- AWS CodeBuild
- CloudFormation
- AWS CodeDeploy
- AWS Elastic Beanstalk
- AWS Lambda
- AWS OpsWorks

Se você criar uma política do IAM que seja mais restritiva que as permissões mínimas necessárias, o console do não funcionará como pretendido para os usuários com essa política do IAM. Para garantir que esses usuários ainda consigam usar o console, associe também a política gerenciada `AWSCodePipeline_ReadOnlyAccess` ao usuário, conforme descrito em AWS políticas gerenciadas para AWS CodePipeline.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas para a API AWS CLI ou para a API.

Permissões necessárias para visualizar registros computacionais no console

Para visualizar os registros na ação Comandos no CodePipeline console, a função do console deve ter permissões. Para visualizar logs no console, adicione as permissões `logs:GetLogEvents` ao perfil do console.

Na declaração de política do perfil do console, reduza as permissões até o nível do pipeline, conforme mostrado no exemplo a seguir.

```
{
    "Effect": "Allow",
    "Action": [
        "Action": "logs:GetLogEvents"
```

```
    ],
    "Resource": "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
}
```

AWS políticas gerenciadas para AWS CodePipeline

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque elas estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo as políticas gerenciadas pelo cliente que são específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas operações de API são disponibilizadas para serviços existentes.

Para saber mais, consulte AWS Políticas gerenciadas pela no Guia do usuário do IAM.

> **Important**
>
> As políticas gerenciadas pela AWS `AWSCodePipelineFullAccess` e `AWSCodePipelineReadOnlyAccess` foram substituídas. Use as políticas `AWSCodePipeline_FullAccess` e `AWSCodePipeline_ReadOnlyAccess`.

AWS política gerenciada: **`AWSCodePipeline_ FullAccess`**

Esta é uma política que concede acesso total CodePipeline a. Para ver o documento de política JSON no console do IAM, consulte AWSCodePipeline_ FullAccess.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `codepipeline`— Concede permissões para CodePipeline.
- `chatbot`: concede permissões para permitir que as entidades principais gerenciem recursos no Amazon Q Developer em aplicações de chat.
- `cloudformation`— Concede permissões para permitir que os diretores gerenciem as pilhas de recursos no. CloudFormation
- `cloudtrail`— Concede permissões para permitir que os diretores gerenciem CloudTrail os recursos de registro.
- `codebuild`— Concede permissões para permitir que os diretores acessem os recursos de construção no CodeBuild.
- `codecommit`— Concede permissões para permitir que os diretores acessem os recursos de origem em CodeCommit.
- `codedeploy`— Concede permissões para permitir que os diretores acessem os recursos de implantação no CodeDeploy.
- `codestar-notifications`— Concede permissões para permitir que os diretores acessem recursos nas AWS CodeStar Notificações.
- `ec2`— Concede permissões para permitir implantações CodeCatalyst para gerenciar o balanceamento elástico de carga no Amazon EC2.
- `ecr`: concede permissões para o acesso a recursos no Amazon ECR.
- `elasticbeanstalk`: concede permissões para que as entidades principais acessem recursos no Elastic Beanstalk.
- `iam`: concede permissões para que as entidades principais gerenciem perfis e políticas no IAM.
- `lambda`: concede permissões para que as entidades principais gerenciem recursos no Lambda.
- `events`— Concede permissões para permitir que os diretores gerenciem recursos em CloudWatch Eventos.
- `opsworks`— Concede permissões para permitir que os diretores gerenciem recursos em AWS OpsWorks.

- `s3`: concede permissões para que as entidades principais gerenciem recursos no Amazon S3.
- `sns`: concede permissões para que as entidades principais gerenciem recursos de notificação no Amazon SNS.
- `states`— Concede permissões para permitir que os diretores visualizem as máquinas de estado em AWS Step Functions. Uma máquina de estado consiste em uma coleção de estados que gerenciam tarefas e fazem a transição entre estados.

Para a política, consulte AWSCodePipeline_ FullAccess.

AWS política gerenciada: **`AWSCodePipeline_ ReadOnlyAccess`**

Essa é uma política que concede acesso somente para leitura a. CodePipeline Para ver o documento de política JSON no console do IAM, consulte AWSCodePipeline_ ReadOnlyAccess.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `codepipeline`— Concede permissões para ações em CodePipeline.
- `codestar-notifications`— Concede permissões para permitir que os diretores acessem recursos nas AWS CodeStar Notificações.
- `s3`: concede permissões para que as entidades principais gerenciem recursos no Amazon S3.
- `sns`: concede permissões para que as entidades principais gerenciem recursos de notificação no Amazon SNS.

Para a política, consulte AWSCodePipeline_ ReadOnlyAccess.

AWS política gerenciada: **`AWSCodePipelineApproverAccess`**

Esta é uma política que concede permissão para aprovar ou rejeitar uma ação de aprovação manual. Para ver o documento de política JSON no console do IAM, consulte AWSCodePipelineApproverAccess.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `codepipeline`— Concede permissões para ações em CodePipeline.

Para a política, consulte AWSCodePipelineApproverAccess.

AWS política gerenciada: **`AWSCodePipelineCustomActionAccess`**

Essa é uma política que concede permissão para criar ações personalizadas CodePipeline ou integrar recursos do Jenkins para criar ou testar ações. Para ver o documento de política JSON no console do IAM, consulte AWSCodePipelineCustomActionAccess.

Detalhes das permissões

Esta política inclui as seguintes permissões.

- `codepipeline`— Concede permissões para ações em CodePipeline.

Para a política, consulte AWSCodePipelineCustomActionAccess.

CodePipeline políticas e notificações gerenciadas

CodePipeline suporta notificações, que podem notificar os usuários sobre mudanças importantes nos pipelines. Políticas gerenciadas para CodePipeline incluir declarações de política para funcionalidade de notificação. Para obter mais informações, consulte O que são notificações?.

Permissões relacionadas a notificações em políticas gerenciadas de acesso total

Essa política gerenciada concede permissões CodePipeline junto com os serviços relacionados CodeCommit CodeBuild, CodeDeploy, e AWS CodeStar notificações. A política também concede as permissões necessárias para trabalhar com outros serviços que se integram aos seus pipelines, como Amazon S3, Elastic Beanstalk, Amazon EC2 e. CloudTrail CloudFormation Os usuários

com essa política gerenciada aplicada também podem criar e gerenciar tópicos do Amazon SNS para notificações, assinar e cancelar a assinatura de usuários dos tópicos, listar tópicos a serem escolhidos como destinos para regras de notificação e listar clientes de aplicações em chat do Amazon Q Developer configurados para o Slack.

A política gerenciada `AWSCodePipeline_FullAccess` inclui as declarações a seguir para permitir acesso total às notificações.

```
    {
        "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": "*"
    }
```

Permissões relacionadas a notificações em políticas gerenciadas somente leitura

A política gerenciada `AWSCodePipeline_ReadOnlyAccess` inclui as instruções a seguir para permitir acesso somente leitura às notificações. Os usuários com essa política aplicada podem visualizar notificações de recursos, mas não podem criá-las, gerenciá-las ou inscrever-se nelas.

```
  {
        "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 obter mais informações sobre o IAM e as notificações, consulte Identity and Access Management para o AWS CodeStar Notifications.

AWS CodePipeline atualizações para AWS políticas gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas CodePipeline desde que esse serviço começou a rastrear essas alterações. Para receber alertas automáticos sobre alterações realizadas nesta página, assine o feed RSS na CodePipeline página de histórico do documento do https://docs.aws.amazon.com/codepipeline/latest/userguide/history.html.

Alteração	Descrição	Data
AWSCodePipeline_ FullAccess — Atualizações da política existente	CodePipeline adicionou uma permissão a essa política para apoiar `ListStacks` em CloudFormation.	15 de março de 2024
AWSCodePipeline_ FullAccess — Atualizações da política existente	Essa política foi atualizada a fim de adicionar permissões para o Amazon Q Developer em aplicações de chat. Para obter mais informações, consulte CodePipeline políticas e notificações gerenciadas.	21 de junho de 2023
AWSCodePipeline_ FullAccess e AWSCodePipeline_ políticas ReadOnlyAccess gerenciadas — Atualizações da política existente	CodePipeline adicionou uma permissão a essas políticas para oferecer suporte a um tipo de notificação adicional usando o Amazon Q Developer em aplicativos	16 de maio de 2023

Alteração	Descrição	Data
	de bate-papo,`chatbot:ListMicrosoftTeamsChannelConfigurations` .	
AWSCodePipelineFullAccess: descontinuada	Esta política foi substituída por `AWSCodePipeline_FullAccess` . Depois de 17 de novembro de 2022, esta política não poderá ser anexada a novos usuários, grupos ou perfis. Para obter mais informações, consulte AWS políticas gerenciadas para AWS CodePipeline.	17 de novembro de 2022
AWSCodePipelineReadOnlyAccess: descontinuada	Esta política foi substituída por `AWSCodePipeline_ReadOnlyAccess` . Depois de 17 de novembro de 2022, esta política não poderá ser anexada a novos usuários, grupos ou perfis. Para obter mais informações, consulte AWS políticas gerenciadas para AWS CodePipeline.	17 de novembro de 2022
CodePipeline começou a rastrear as alterações	CodePipeline começou a rastrear as mudanças em suas políticas AWS gerenciadas.	12 de março de 2021

Exemplos de política gerenciada pelo cliente

Nesta seção, você encontrará exemplos de políticas de usuário que concedem permissões para diversas ações do . Essas políticas funcionam quando você está usando a API, AWS os SDKs ou o. AWS CLI Ao usar o console, você deve conceder permissões adicionais específicas para o console. Para obter mais informações, consulte Permissões necessárias para usar o console.

> **Note**
>
> Todos os exemplos usam a Região do Oeste dos EUA (Oregon) (`us-west-2`) e contêm IDs de conta fictícios.

Exemplos

- Exemplo 1: conceder permissões para obter o estado de um pipeline
- Exemplo 2: conceder permissões para habilitar e desabilitar transições entre estágios
- Exemplo 3: conceder permissões para obter uma lista de todos os tipos de ação disponíveis
- Exemplo 4: conceder permissões para aprovar ou rejeitar ações de aprovação manual
- Exemplo 5: conceder permissões para pesquisar por trabalhos para uma ação personalizada
- Exemplo 6: anexar ou editar uma política para integração do Jenkins com AWS CodePipeline
- Exemplo 7: configurar o acesso entre contas em um pipeline

Exemplo 1: conceder permissões para obter o estado de um pipeline

O exemplo a seguir concede permissões para obter o estado do pipeline chamado `MyFirstPipeline`:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:GetPipelineState"
```

```
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline"
        }
    ]
}
```

Exemplo 2: conceder permissões para habilitar e desabilitar transições entre estágios

O exemplo a seguir concede permissões para permitir e desativar transições entre todos os estágios no pipeline chamado `MyFirstPipeline`:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:DisableStageTransition",
                "codepipeline:EnableStageTransition"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline/*"
        }
    ]
}
```

Para permitir que o usuário permita ou desative transições de um único estágio em um pipeline, você deve especificar o estágio. Por exemplo, para permitir que o usuário habilite e desabilite transições de um estágio chamado `Staging` em um pipeline chamado `MyFirstPipeline`:

```
"Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/Staging"
```

Exemplo 3: conceder permissões para obter uma lista de todos os tipos de ação disponíveis

O exemplo a seguir concede permissões para obter uma lista de todos os tipos de ação disponíveis para os pipelines na região `us-west-2`:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:ListActionTypes"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:actiontype:*"
        }
    ]
}
```

Exemplo 4: conceder permissões para aprovar ou rejeitar ações de aprovação manual

O exemplo a seguir concede permissões para aprovar ou rejeitar ações de aprovação manual em um estágio chamado `Staging` em um pipeline chamado `MyFirstPipeline`:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PutApprovalResult"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline/Staging/*"
        }
    ]
}
```

Exemplo 5: conceder permissões para pesquisar por trabalhos para uma ação personalizada

O exemplo a seguir concede permissões para pesquisar por trabalhos para a ação personalizada chamada `TestProvider`, que é um tipo de ação de `Test` na primeira versão, em todos os pipelines:

> **Note**
>
> O operador de trabalho de uma ação personalizada pode ser configurado em outra conta da AWS ou exigir um perfil do IAM específica para funcionar.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PollForJobs"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111222333444:actionType:Custom/Test/TestProvider/1"
            ]
        }
    ]
}
```

Exemplo 6: anexar ou editar uma política para integração do Jenkins com AWS CodePipeline

Se você configurar um pipeline para usar o Jenkins para criar ou testar, crie uma identidade separada para essa integração e anexe uma política do IAM que tenha as permissões mínimas necessárias para a integração entre Jenkins e. Essa política é a mesma da política gerenciada `AWSCodePipelineCustomActionAccess`. O exemplo a seguir mostra uma política para integração do Jenkins:

JSON

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

    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:AcknowledgeJob",
                "codepipeline:GetJobDetails",
                "codepipeline:PollForJobs",
                "codepipeline:PutJobFailureResult",
                "codepipeline:PutJobSuccessResult"
            ],
            "Resource": "*"
        }
    ]
}
```

Exemplo 7: configurar o acesso entre contas em um pipeline

Você pode configurar o acesso a pipelines para usuários e grupos em outra conta do AWS . A maneira recomendada é criar uma função na conta onde o pipeline foi criado. A função deve permitir que os usuários da outra AWS conta assumam essa função e acessem o pipeline. Para obter mais informações, consulte Passo a passo: Cross-Account Acesso usando funções.

O exemplo a seguir mostra uma política na conta 80398EXAMPLE que permite que os usuários visualizem, mas não alterem, o pipeline nomeado `MyFirstPipeline` no console. Essa política é baseada na política gerenciada `AWSCodePipeline_ReadOnlyAccess`, mas, como ela é específica ao pipeline `MyFirstPipeline`, a política gerenciada não pode ser usada diretamente. Se você não desejar restringir a política a um pipeline específico, leve em consideração usar uma das políticas gerenciadas criadas e mantidas pelo . Para obter mais informações, consulte Como trabalhar com políticas gerenciadas. Você deve anexar essa política a um perfil do IAM criado para acesso, por exemplo, a um perfil chamado `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"
        }
    ]
}
```

Após criar essa política, crie o perfil do IAM na conta 80398EXAMPLE e anexe a política a esse perfil. Nas relações de confiança da função, você deve adicionar a AWS conta que assume essa função.

O exemplo a seguir mostra uma política criada na *111111111111* AWS conta que permite que os usuários assumam a função nomeada `CrossAccountPipelineViewers` na conta 80398EXAMPLE:

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource":
 "arn:aws:iam::111122223333:role/CrossAccountPipelineViewers"
        }
    ]
}
```

AWS CodePipeline exemplos de políticas baseadas em recursos

Outros serviços, como o Amazon S3, também aceitam políticas de permissões baseadas em recurso. Por exemplo: você pode anexar uma política a um bucket do S3 para gerenciar permissões de acesso a esse bucket. Embora CodePipeline não ofereça suporte a políticas baseadas em recursos, ele armazena artefatos para serem usados em pipelines em buckets S3 versionados.

Example Para criar uma política para um bucket do S3 a ser usado como armazenamento de artefatos para CodePipeline

Você pode usar qualquer bucket S3 versionado como armazenamento de artefatos para. CodePipeline Se você usar o assistente Create Pipeline (Criar pipeline) para criar o primeiro pipeline, esse bucket do S3 será criado para garantir que todos os objetos carregados para o armazenamento de artefatos sejam criptografados e que as conexões com o bucket sejam confiáveis. Como melhor prática, se você criar seu próprio bucket do S3, considere adicionar a política a seguir ou seus elementos ao bucket. Nessa política, o ARN do bucket do S3 é `codepipeline-us-east-2-1234567890`. Substitua esse ARN pelo ARN do seu bucket do 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
                }
            }
        }
    ]
}
```

Solução de problemas AWS CodePipeline identidade e acesso

Use as informações a seguir para ajudá-lo a diagnosticar e corrigir problemas comuns que você pode encontrar ao trabalhar com CodePipeline um IAM.

Tópicos

- Não estou autorizado a realizar uma ação em CodePipeline
- Não estou autorizado a realizar iam: PassRole
- Sou administrador e quero permitir que outras pessoas acessem CodePipeline
- Quero permitir que pessoas fora da minha AWS conta para acessar meus CodePipeline recursos

Não estou autorizado a realizar uma ação em CodePipeline

Se o Console de gerenciamento da AWS indicar que você não está autorizado a realizar uma ação, entre em contato com o administrador para obter ajuda. O administrador é a pessoa que forneceu a você o seu nome de usuário e senha.

O erro de exemplo a seguir ocorre quando o usuário `mateojackson` do IAM tenta usar o console para visualizar detalhes sobre um pipeline, mas não tem permissões `codepipeline:GetPipeline`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 codepipeline:GetPipeline on resource: my-pipeline
```

Neste caso, Mateo pede ao administrador para atualizar suas políticas para permitir a ele o acesso ao recurso `my-pipeline` usando a ação `codepipeline:GetPipeline`.

Não estou autorizado a realizar iam: PassRole

Se você receber uma mensagem de erro informando que você não está autorizado a executar a ação `iam:PassRole`, entre em contato com o administrador para obter assistência. O administrador é a pessoa que forneceu a você o seu nome de usuário e senha. Peça a essa pessoa para atualizar suas políticas para permitir que você passe uma função para o CodePipeline.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço, em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta utilizar o console para executar uma ação no CodePipeline. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar a função para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Neste caso, Mary pede ao administrador para atualizar suas políticas para permitir que ela execute a ação `iam:PassRole`.

Sou administrador e quero permitir que outras pessoas acessem CodePipeline

Para permitir que outras pessoas acessem CodePipeline, você deve conceder permissão às pessoas ou aplicativos que precisam de acesso. Se você estiver usando o Centro de Identidade do AWS IAM para gerenciar pessoas e aplicações, atribua conjuntos de permissões a usuários ou grupos para definir o nível de acesso. Os conjuntos de permissões criam e atribuem automaticamente políticas do IAM aos perfis do IAM associados à pessoa ou aplicação. Para ter mais informações, consulte Conjuntos de permissões no Guia do usuário do Centro de Identidade do AWS IAM .

Se você não estiver usando o Centro de Identidade do IAM, deverá criar entidades do IAM (usuários ou perfis) para as pessoas ou aplicações que precisam de acesso. Você deve anexar uma política à entidade que concede a eles as permissões corretas no CodePipeline. Depois que as permissões forem concedidas, forneça as credenciais ao usuário ou desenvolvedor da aplicação. Eles usarão essas credenciais para acessar AWS. Para saber mais sobre como criar grupos, políticas, permissões e usuários do IAM, consulte Identidades do IAM e Políticas e permissões no IAM no Guia do usuário do IAM.

Quero permitir que pessoas fora da minha AWS conta para acessar meus CodePipeline recursos

É possível criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), é possível usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se é CodePipeline compatível com esses recursos, consulteComo AWS CodePipeline funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

referência de permissões

Use a tabela a seguir como referência ao configurar o controle de acesso e escrever políticas de permissões que você pode anexar a uma identidade do IAM (políticas baseadas em identidade). A tabela lista cada operação da API do e as ações correspondentes para as quais você pode conceder permissões para executar a ação. Para operações que oferecem suporte a permissões em nível

de recurso, a tabela lista o AWS recurso para o qual você pode conceder as permissões. Você especifica as ações no campo `Action` das políticas.

Resource-level permissões são aquelas que permitem especificar em quais recursos os usuários podem realizar ações. AWS CodePipeline fornece suporte parcial para permissões em nível de recurso. Isso significa que, para algumas chamadas de AWS CodePipeline API, você pode controlar quando os usuários podem usar essas ações com base nas condições que devem ser atendidas ou nos recursos que os usuários podem usar. Por exemplo, você pode conceder permissões a usuários para listar as informações de execução de pipelines, mas apenas para um pipeline ou pipelines específicos.

> **Note**
>
> A coluna Resources (Recursos) lista o recurso necessário para chamadas de API que oferecem suporte a permissões no nível do recurso. Para chamadas de API que não oferecem suporte a permissões no nível do recurso, é possível conceder aos usuários permissão para usá-las, mas é necessário especificar um curinga (*) para o elemento de recurso da declaração de política.

Operações de API do e permissões necessárias para ações

AcknowledgeJob

Ação: `codepipeline:AcknowledgeJob`

Necessário para visualizar informações sobre um trabalho especificado e se esse trabalho foi recebido pelo operador do trabalho. Usado somente para ações personalizadas.

Recursos: compatível apenas com um caractere curinga (*) no elemento de política `Resource`.

AcknowledgeThirdPartyJob

Ação: `codepipeline:AcknowledgeThirdPartyJob`

Necessário para confirmar que um operador do trabalho recebeu o trabalho especificado. Usado somente para ações de parceiros.

Recursos: compatível apenas com um caractere curinga (*) no elemento de política `Resource`.

CreateCustomActionType

Ação: `codepipeline:CreateCustomActionType`

Necessário para criar uma nova ação personalizada que possa ser usada em todos os pipelines associados à AWS conta. Usado somente para ações personalizadas.

Recursos:

Tipo de ação

`arn:aws:codepipeline:`*region*`:`*account*`:actiontype:`*owner*`/`*category*`/`*provider*`/`*versio*

CreatePipeline

Ação: `codepipeline:CreatePipeline`

Necessário para criar um pipeline.

Recursos:

Pipeline

`arn:aws:codepipeline:`*region*`:`*account*`:`*pipeline-name*

DeleteCustomActionType

Ação: `codepipeline:DeleteCustomActionType`

Necessário para marcar uma ação personalizada como excluída. `PollForJobs` para a ação personalizada falha após a ação ser marcada para exclusão. Usado somente para ações personalizadas.

Recursos:

Tipo de ação

`arn:aws:codepipeline:`*region*`:`*account*`:actiontype:`*owner*`/`*category*`/`*provider*`/`*versio*

DeletePipeline

Ação: `codepipeline:DeletePipeline`

Necessário para apagar um pipeline.

Recursos:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

DeleteWebhook

Ação:`codepipeline:DeleteWebhook`

Necessário para apagar um webhook.

Recursos:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

DeregisterWebhookWithThirdParty

Ação:`codepipeline:DeregisterWebhookWithThirdParty`

Antes de excluir um webhook, é necessário remover a conexão entre o webhook criado por CodePipeline e a ferramenta externa com eventos a serem detectados. Atualmente suportado apenas para webhooks que têm como alvo um tipo de ação de. GitHub

Recursos:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

DisableStageTransition

Ação: `codepipeline:DisableStageTransition`

Necessário para impedir que os artefatos em um pipeline façam a transição para o próximo estágio no pipeline.

Recursos:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

EnableStageTransition

Ação: `codepipeline:EnableStageTransition`

Necessário para permitir que os artefatos em um pipeline façam a transição para um estágio em um pipeline.

Recursos:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

GetJobDetails

Ação: `codepipeline:GetJobDetails`

Necessário para recuperar informações sobre um trabalho. Usado somente para ações personalizadas.

Recursos: nenhum recurso necessário.

GetPipeline

Ação: `codepipeline:GetPipeline`

Necessário para recuperar a estrutura, os estágios, as ações e os metadados de um pipeline, inclusive o ARN do pipeline.

Recursos:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

GetPipelineExecution

Ação: `codepipeline:GetPipelineExecution`

Necessário para recuperar informações sobre uma execução de um pipeline, incluindo os detalhes sobre os artefatos, o ID de execução do pipeline e o nome, a versão e o status do pipeline.

Recursos:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

GetPipelineState

Ação: `codepipeline:GetPipelineState`

Necessário para recuperar informações sobre o estado de um pipeline, incluindo os estágios e as ações.

Recursos:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

GetThirdPartyJobDetails

Ação: `codepipeline:GetThirdPartyJobDetails`

Necessário para solicitar os detalhes de um trabalho de uma ação de terceiros. Usado somente para ações de parceiros.

Recursos: compatível apenas com um caractere curinga (*) no elemento de política `Resource`.

ListActionTypes

Ação: `codepipeline:ListActionTypes`

Necessário para gerar um resumo de todos os tipos de CodePipeline ação associados à sua conta.

Recursos:

Tipo de ação

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versior`

ListPipelineExecutions

Ação: `codepipeline:ListPipelineExecutions`

Necessário para gerar um resumo das execuções mais recentes de um pipeline.

Recursos:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

ListPipelines

Ação: `codepipeline:ListPipelines`

Necessário para gerar um resumo de todos os pipelines associados à sua conta.

Recursos:

ARN do pipeline com curinga (permissões em nível de recurso no nível do nome do pipeline não são compatíveis)

`arn:aws:codepipeline:region:account:*`

ListTagsForResource

Ação: `codepipeline:ListTagsForResource`

Obrigatório para listar tags de um recurso específico.

Recursos:

Tipo de ação

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

ListWebhooks

Ação:`codepipeline:ListWebhooks`

Necessário para listar todos os webhooks na conta dessa região.

Recursos:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

PollForJobs

Ação/Ações: `codepipeline:PollForJobs`

Necessário para recuperar informações sobre quaisquer trabalhos nos CodePipeline quais atuar.

Recursos:

Tipo de ação

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versio`

PollForThirdPartyJobs

Ação: `codepipeline:PollForThirdPartyJobs`

Necessário para determinar se há trabalhos de terceiros nos quais um operador de trabalho possa atuar. Usado somente para ações de parceiros.

Recursos: compatível apenas com um caractere curinga (*) no elemento de política `Resource`.

PutActionRevision

Ação: `codepipeline:PutActionRevision`

Obrigatório para relatar informações CodePipeline sobre novas revisões a uma fonte.

Recursos:

Ação

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

PutApprovalResult

Ação: `codepipeline:PutApprovalResult`

Necessário para relatar a resposta a uma solicitação de aprovação manual ao CodePipeline As respostas válidas são `Approved` e `Rejected`.

Recursos:

Ação

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

> **Note**
>
> Essa chamada à API oferece suporte a permissões em nível de recurso. No entanto, você pode encontrar um erro ao usar o console do IAM ou o Gerador de políticas para criar políticas com `"codepipeline:PutApprovalResult"` que especifica um ARN de recurso. Se encontrar um erro, você poderá usar a guia JSON no console do IAM ou a CLI para criar uma política.

PutJobFailureResult

Ação: `codepipeline:PutJobFailureResult`

Necessário para relatar a falha de um trabalho como devolvido para o pipeline por um operador de trabalho. Usado somente para ações personalizadas.

Recursos: compatível apenas com um caractere curinga (*) no elemento de política `Resource`.

PutJobSuccessResult

Ação: `codepipeline:PutJobSuccessResult`

Necessário para relatar o sucesso de um trabalho como devolvido ao pipeline por um operador de trabalho. Usado somente para ações personalizadas.

Recursos: compatível apenas com um caractere curinga (*) no elemento de política `Resource`.

PutThirdPartyJobFailureResult

Ação: `codepipeline:PutThirdPartyJobFailureResult`

Necessário para relatar a falha de um trabalho de terceiros como devolvido ao pipeline por um operador de trabalho. Usado somente para ações de parceiros.

Recursos: compatível apenas com um caractere curinga (*) no elemento de política `Resource`.

PutThirdPartyJobSuccessResult

Ação: `codepipeline:PutThirdPartyJobSuccessResult`

Necessário para relatar o sucesso de um trabalho de terceiros como devolvido ao pipeline por um operador de trabalho. Usado somente para ações de parceiros.

Recursos: compatível apenas com um caractere curinga (*) no elemento de política `Resource`.

PutWebhook

Ação:`codepipeline:PutWebhook`

Necessário para criar um webhook.

Recursos:

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RegisterWebhookWithThirdParty

Ação:`codepipeline:RegisterWebhookWithThirdParty`

Recursos:

Depois da criação de um webhook, necessário para configurar os terceiros com suporte para chamar o URL do webhook gerado.

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RetryStageExecution

Ação: `codepipeline:RetryStageExecution`

Necessário para reiniciar a execução do pipeline ao recuperar as últimas ações falhas em um estágio.

Recursos:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

StartPipelineExecution

Ação: `codepipeline:StartPipelineExecution`

Necessário para iniciar o pipeline especificado (especificamente, para iniciar o processamento da confirmação mais recente para o local de origem especificado como parte do pipeline).

Recursos:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

TagResource

Ação: `codepipeline:TagResource`

Obrigatório para marcar o recurso especificado.

Recursos:

Tipo de ação

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versio`

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UntagResource

Ação: `codepipeline:UntagResource`

Necessário para desmarcar o recurso especificado.

Recursos:

Tipo de ação

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versio`

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UpdatePipeline

Ação: `codepipeline:UpdatePipeline`

Necessário para atualizar um pipeline especificado com edições ou alterações em sua estrutura.

Recursos:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Gerenciar a função CodePipeline de serviço

A função de serviço é configurada com uma ou mais políticas que controlam o acesso aos AWS recursos usados pelo pipeline. Talvez você queira anexar mais políticas a essa função, editar a política anexada à função ou configurar políticas para outras funções de serviço em AWS. Pode ser que você também queira anexar uma política a uma função ao configurar o acesso entre contas ao pipeline.

Important

Alterar uma declaração de política ou anexar outra política à função pode impedir o funcionamento dos pipelines. Compreenda as implicações antes de alterar de alguma maneira a função de serviço do . Teste os pipelines após ter feito alterações na função de serviço.

Note

No console, as funções de serviço criadas antes de setembro de 2018 são criadas com o nome `oneClick_AWS-CodePipeline-Service_ID-Number`.

As funções de serviço criadas após setembro de 2018 usam o formato de nome da função de serviço `AWSCodePipelineServiceRole-`*`Region`*`-`*`Pipeline_Name`*. Por exemplo, para um pipeline chamado `MyFirstPipeline` na `eu-west-2`, o console nomeia a função e a política como `AWSCodePipelineServiceRole-eu-west-2-MyFirstPipeline`.

CodePipeline política de função de serviço

A declaração de política da função de CodePipeline serviço contém as permissões mínimas para gerenciar pipelines. Você pode editar a declaração do perfil de serviço para remover ou adicionar o acesso aos recursos que você não usa. Consulte a referência de ação apropriada para ver os CodePipeline usos mínimos de permissões necessárias para cada ação.

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**
>
> Na política, as seguintes permissões são necessárias quando os objetos do S3 no bucket de origem têm tags:
>
> ```
> s3:PutObjectTagging
> s3:GetObjectTagging
> s3:GetObjectVersionTagging
> ```

Remover permissões da função CodePipeline de serviço

Você pode editar a declaração da função de serviço para remover o acesso aos recursos que você não utiliza. Por exemplo, se nenhum dos pipelines incluir o Elastic Beanstalk, você poderá editar a declaração de política para remover a seção que concede acesso aos recursos do Elastic Beanstalk.

Da mesma forma, se nenhum de seus pipelines incluir CodeDeploy, você poderá editar a declaração de política para remover a seção que concede acesso aos CodeDeploy recursos:

```
    {
    "Action": [
        "codedeploy:CreateDeployment",
```

```
        "codedeploy:GetApplicationRevision",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:RegisterApplicationRevision"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
```

Adicionar permissões à função CodePipeline de serviço

Você deve atualizar a declaração de política do perfil de serviço com permissões para um AWS service (Serviço da AWS) ainda não incluído na declaração de política de perfil de serviço padrão para que possa usá-la nos pipelines.

Isso é especialmente importante se a função de serviço que você usa para seus pipelines foi criada antes da adição do suporte para um AWS service (Serviço da AWS).

A tabela a seguir mostra quando o suporte foi adicionado para outros Serviços da AWS.

AWS service (Serviço da AWS)	CodePipeline data de suporte
CodePipeline suporte de ação de invocação adicionado. Consulte Permissões da política de função de serviço para a CodePipeline ação de invocação.	14 de março de 2025
Suporte à ação do `EC2` adicionado. Consulte Permissões de política do perfil de serviço para a ação de implantação do EC2.	21 de fevereiro de 2025
Suporte à ação do `EKS` adicionado. Consulte Permissões de política de perfil de serviço.	20 de fevereiro de 2025
Suporte à ação `ECRBuildAndPublish` ação do Amazon Elastic Container Registry adicionado. Consulte Permissões do perfil de serviço: ação `ECRBuildAndPublish` .	22 de novembro de 2024

AWS service (Serviço da AWS)	CodePipeline data de suporte
Suporte à ação `InspectorScan` do Amazon Inspector adicionado. Consulte Permissões do perfil de serviço: ação `InspectorScan`.	22 de novembro de 2024
Suporte à ação de comandos adicionado. Consulte Permissões do perfil de serviço: ação Comandos.	3 de outubro de 2024
CloudFormation suporte de ação adicionado. Consulte Permissões do perfil de serviço: ação `CloudFormationStackSet` e Permissões do perfil de serviço: ação `CloudFormationStackInstances`.	30 de dezembro de 2020
CodeCommit suporte de ação de formato de artefato de saída de clone completo adicionado. Consulte Permissões da função de serviço: CodeCommit ação.	11 de novembro de 2020
AWS CodeBuild suporte de ação de compilações em lote adicionado. Consulte Permissões da função de serviço: CodeCommit ação.	30 de julho de 2020
AWS AppConfig suporte de ação adicionado. Consulte Permissões do perfil de serviço: ação `AppConfig`.	22 de junho de 2020
AWS Step Functions suporte de ação adicionado. Consulte Permissões do perfil de serviço: ação `StepFunctions`.	27 de maio de 2020
AWS CodeStar Suporte de ação de conexões adicionado. Consulte Permissões da função de serviço: AWS CodeConnections ação.	18 de dezembro de 2019

AWS service (Serviço da AWS)	CodePipeline data de suporte
Suporte à ação de implantação do S3 adicionado. Consulte Permissões do perfil de serviço: ação de implantação do S3.	16 de janeiro de 2019
O suporte à ação `CodeDeployToECS` foi adicionado. Consulte Permissões do perfil de serviço: ação `CodeDeployToECS`.	27 de novembro de 2018
Suporte à ação do Amazon ECR adicionado. Consulte Permissões do perfil de serviço: ação do Amazon ECR.	27 de novembro de 2018
Suporte à ação do Service Catalog adicionado. Consulte Permissões do perfil de serviço: ação do Service Catalog.	16 de outubro de 2018
AWS Device Farm suporte de ação adicionado. Consulte Permissões da função de serviço: AWS Device Farm ação.	19 de julho de 2018
Suporte à ação do Amazon ECS adicionado. Consulte Permissões do perfil de serviço: ação padrão do Amazon ECS.	12 de dezembro de 2017//Atualização para aceitar a autorização de marcação em 21 de julho de 2017
CodeCommit suporte de ação adicionado. Consulte Permissões da função de serviço: CodeCommit ação.	18 de abril de 2016
AWS OpsWorks suporte de ação adicionado. Consulte Permissões da função de serviço: AWS OpsWorks ação.	2 de junho de 2016
CloudFormation suporte de ação adicionado. Consulte Permissões da função de serviço: CloudFormation ação.	3 de novembro de 2016

AWS service (Serviço da AWS)	CodePipeline data de suporte
AWS CodeBuild suporte de ação adicionado. Consulte Permissões da função de serviço: CodeBuild ação.	1º de dezembro de 2016
Suporte à ação do Elastic Beanstalk adicionado. Consulte Permissões do perfil de serviço: ação de implantação do `ElasticBeanstalk`.	Lançamento do serviço inicial
CodeDeploy suporte de ação adicionado. Consulte Permissões da função de serviço: AWS CodeDeploy ação.	Lançamento do serviço inicial
Suporte à ação de origem do S3 adicionado. Consulte Permissões do perfil de serviço: ação de origem do S3.	Lançamento do serviço inicial

Siga estes passos para adicionar permissões para um serviço compatível:

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No console do IAM, no painel de navegação, escolha Perfis e, em seguida, escolha o perfil `AWS-CodePipeline-Service` na lista de perfis.
3. Na guia Permissões, em Políticas em linha, na linha da sua política de perfil de serviço, escolha Editar política.
4. Adicione as permissões necessárias na caixa Documento da política.

> **Note**
>
> Ao criar políticas do IAM, siga as dicas de segurança padrão de concessão de privilégio mínimo, ou seja, conceda apenas as permissões necessárias à execução de uma tarefa. Determinadas chamadas de API são compatíveis com permissões baseadas em recursos e permitem que o acesso seja limitado. Por exemplo, neste caso, para limitar as permissões quando chamar `DescribeTasks` e `ListTasks`, você pode substituir o

caractere curinga (*) por um ARN de recurso ou um ARN de recurso que contenha um caractere curinga (*). Para obter mais informações sobre a criação de uma política que conceda acesso com privilégios mínimos, consulte https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege.

5. Selecione Review policy (Revisar política) para garantir que a política não contenha erros. Quando a política não tiver erros, selecione Aplicar política.

Registro e monitoramento em CodePipeline

Você pode usar os recursos de login AWS para determinar as ações que os usuários realizaram em sua conta e os recursos que foram usados. Os arquivos de log mostram:

- A data e hora das ações.
- O endereço IP de origem de uma ação.
- As ações que falharam devido a permissões inadequadas.

Os atributos de registro em log estão disponíveis nos seguintes Serviços da AWS:

- AWS CloudTrail pode ser usado para registrar chamadas de AWS API e eventos relacionados feitos por ou em nome de um Conta da AWS. Para obter mais informações, consulte Registrando chamadas de CodePipeline API com AWS CloudTrail.
- O Amazon CloudWatch Events pode ser usado para monitorar seus Nuvem AWS recursos e os aplicativos em que você executa AWS. Você pode criar alertas no Amazon CloudWatch Events com base nas métricas que você define. Para obter mais informações, consulte Monitoramento de CodePipeline eventos.

Validação de conformidade AWS CodePipeline

Nossa nova experiência AWS de inscrição não foi projetada para cargas de trabalho regulamentadas. Se você estiver usando nossa nova experiência de AWS inscrição, mas quiser usá-la AWS para cargas de trabalho regulamentadas, você pode se inscrever AWS (avançado) ou ativar recursos avançados para seu AWS ambiente.

Para saber se um AWS service (Serviço da AWS) está dentro do escopo de programas de conformidade específicos, consulte Serviços da AWS Escopo por Programa de Conformidade

Serviços da AWS e escolha o programa de conformidade em que você está interessado. Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando artefatos. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade de conformidade ao usar Serviços da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua empresa e pelas leis e regulamentações aplicáveis. Para obter mais informações sobre sua responsabilidade de conformidade ao usar Serviços da AWS, consulte a documentação AWS de segurança.

Resiliência em AWS CodePipeline

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Segurança da infraestrutura em AWS CodePipeline

Como serviço gerenciado, AWS CodePipeline é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar CodePipeline pela rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Suítes de criptografia com sigilo direto perfeito (PFS), como DHE (efêmero) ou ECDHE (curva elíptica efêmera Diffie-Hellman). Diffie-Hellman A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Práticas recomendadas de segurança

CodePipeline fornece vários recursos de segurança a serem considerados ao desenvolver e implementar suas próprias políticas de segurança. As práticas recomendadas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas práticas recomendadas podem não ser adequadas ou suficientes para o seu ambiente, trate-as como considerações úteis em vez de prescrições.

Você usa criptografia e autenticação para os repositórios de origem que se conectam a seus pipelines. Estas são as CodePipeline melhores práticas de segurança:

- Se você criar uma configuração de pipeline ou ação que precise incluir segredos, como tokens ou senhas, não insira segredos diretamente na configuração da ação ou nos valores padrão das variáveis definidas no nível do pipeline ou na configuração do CloudFormation , pois as informações serão exibidas nos logs. Use o Secrets Manager para configurar e armazenar segredos e, em seguida, use o segredo referenciado na configuração do pipeline e da ação, conforme descrito em Use AWS Secrets Manager para rastrear senhas de bancos de dados ou chaves de API de terceiros.
- Se você criar um pipeline que usa um bucket de origem do S3, configure a criptografia do lado do servidor para artefatos armazenados no Amazon S3 por CodePipeline meio do gerenciamento, conforme descrito em. AWS KMS keysConfigure a criptografia do lado do servidor para artefatos armazenados no Amazon S3 para CodePipeline
- Se estiver usando um provedor de ação do Jenkins, ao usar um provedor de compilação do Jenkins para a ação de teste ou compilação do pipeline, instale o Jenkins em uma instância do EC2 e configure um perfil de instância do EC2 separado. Certifique-se de que o perfil da instância conceda ao Jenkins somente as AWS permissões necessárias para realizar tarefas para seu projeto, como recuperar arquivos do Amazon S3. Para saber como criar a função para o seu perfil de instância do Jenkins, consulte as etapas em Criar um perfil do IAM a ser usado na integração do Jenkins.

CodePipeline referência de estrutura de tubulação

Você pode usar CodePipeline para estruturar um CI/CD pipeline de etapas automatizadas que realizam tarefas que criam, testam e implantam o código-fonte do seu aplicativo. Esta seção de referência dá detalhes sobre a estrutura JSON e os parâmetros no pipeline. Para obter uma lista detalhada dos conceitos que descrevem como os pipelines são usados, consulte CodePipeline conceitos.

- Ao criar um pipeline, você escolhe uma ação e um provedor de origem disponíveis, como um bucket do S3, CodeCommit repositório, repositório Bitbucket ou GitHub repositório que contém seu código-fonte e inicia seu pipeline quando você confirma uma alteração no código-fonte. Esta seção de referência fornece informações de referência sobre as origens disponíveis para o pipeline. Para obter mais informações sobre como trabalhar com ações de origem, consulte Inicie um funil em CodePipeline.
- Você pode escolher as ações e os provedores de teste, criação e implantação que deseja incluir automaticamente nas execuções do pipeline. Esta seção de referência apresenta informações de referência sobre as ações disponíveis e como elas se encaixam no JSON do pipeline.
- O pipeline finalizado consistirá em um estágio de origem com estágios adicionais nos quais você configura ações para implantar e testar a aplicação. Para ver um exemplo conceitual de um DevOps pipeline que implanta seu aplicativo, consulte. DevOps exemplo de pipeline

Por padrão, qualquer pipeline criado com sucesso AWS CodePipeline tem uma estrutura válida. No entanto, se você criar ou editar manualmente um arquivo JSON para criar um pipeline ou atualizar um pipeline a partir do AWS CLI, poderá criar inadvertidamente uma estrutura inválida. A referência a seguir pode ajudar a entender melhor os requisitos da estrutura do pipeline e como solucionar problemas. Consulte as restrições em Cotas em AWS CodePipeline, que se aplicam a todos os pipelines.

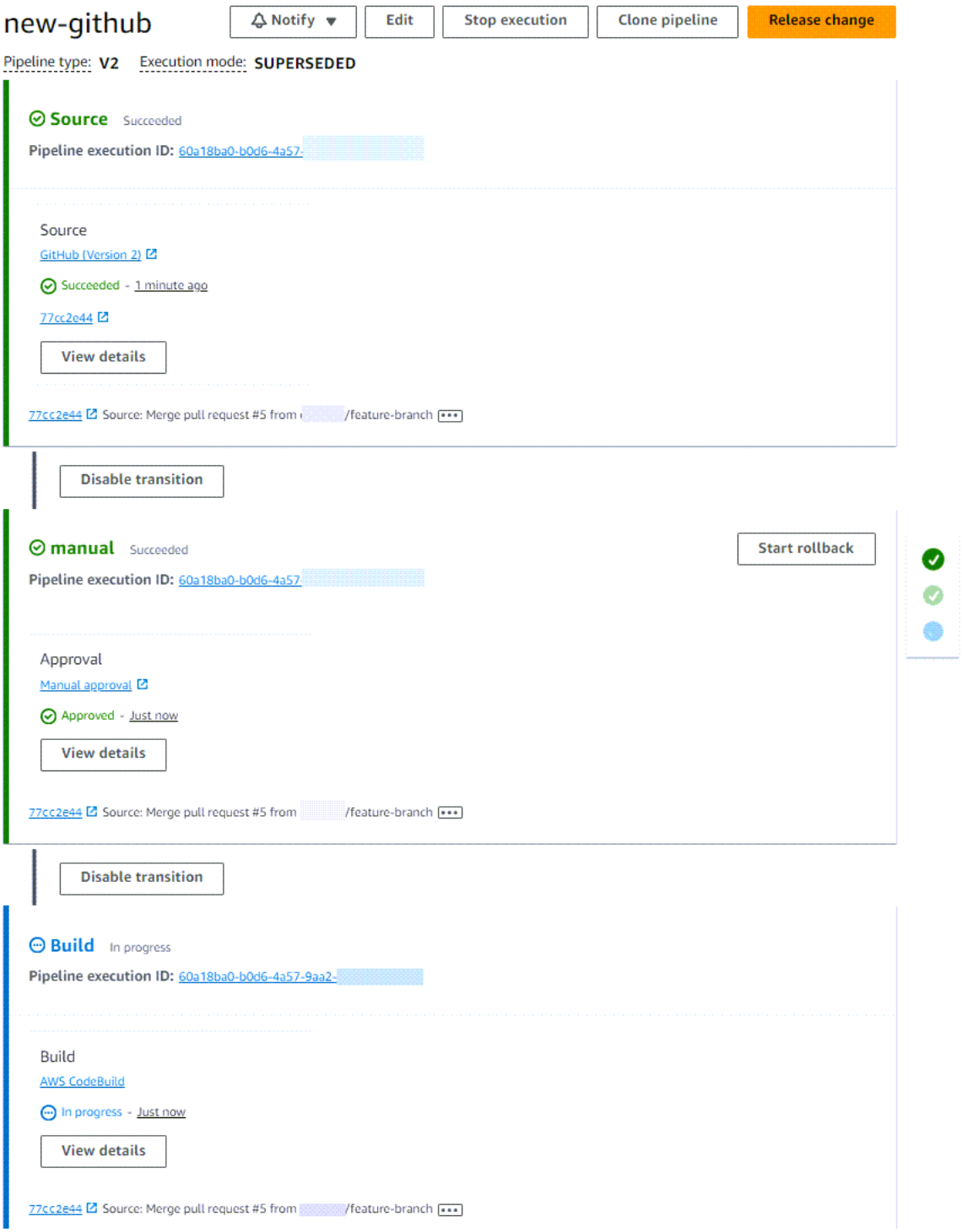
As seções a seguir fornecem parâmetros de alto nível e a posição na estrutura do pipeline. Os requisitos de estrutura de pipeline são detalhados em cada seção para os seguintes tipos de componentes de pipeline:

- Referência de campo para o Declaração de pipeline
- Referência de campo para o Declaração de estágio
- Referência de campo para o Declaração de ação

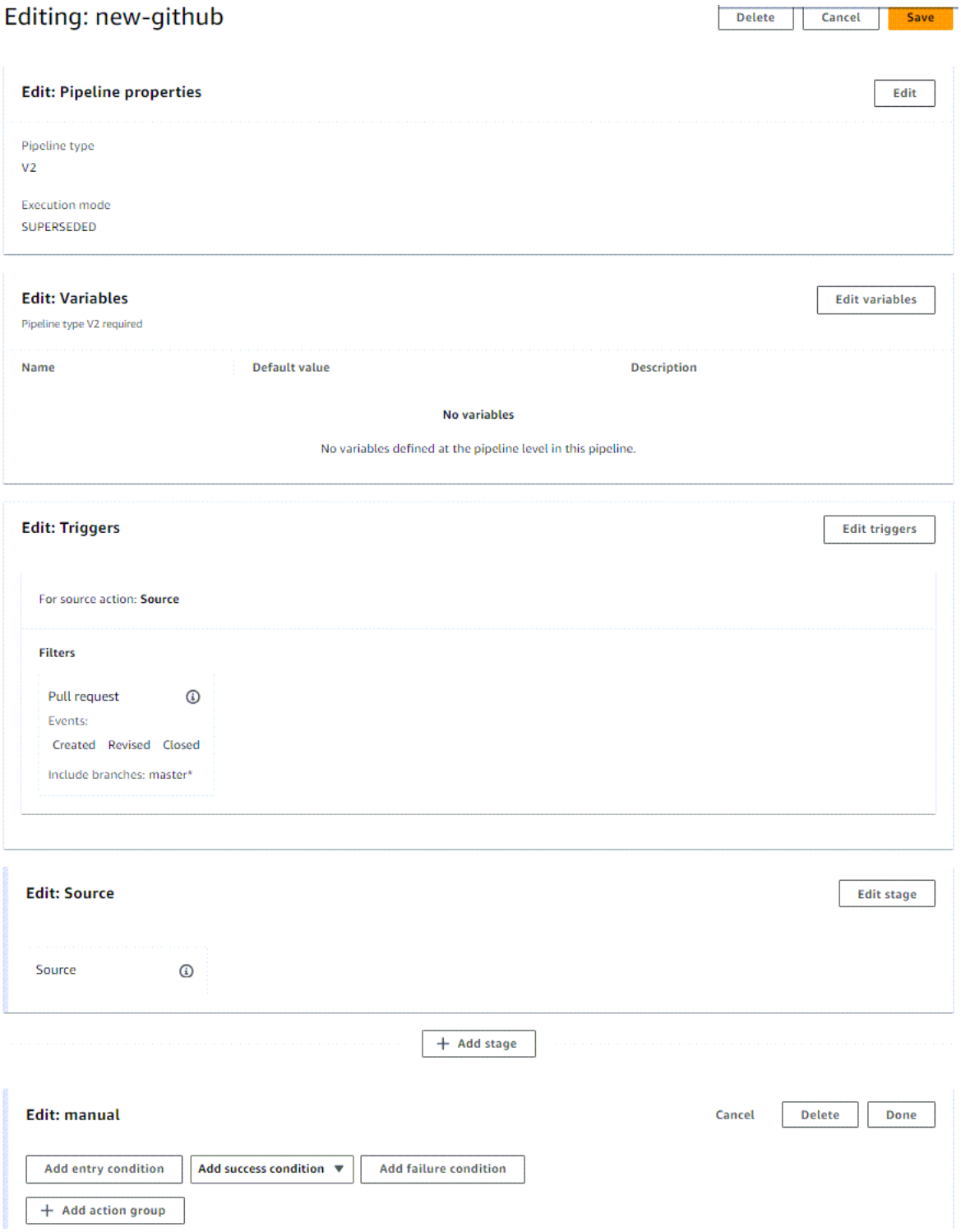
- Lista de Provedores de ação válidos em CodePipeline por tipo de ação
- Referência do Configurações válidas para o parâmetro `PollForSourceChanges`
- Referência do Artefatos de entrada e saída válidos para cada tipo de ação
- Lista de links para Parâmetros de configuração válidos para cada tipo de provedor

Para obter mais informações, consulte o PipelineDeclaration objeto no Guia CodePipeline da API.

O exemplo de visualização do console do pipeline a seguir mostra o pipeline chamado new-github, os estágios denominados `Source` e e e `Build` as ações de GitHub (via GitHub aplicativo), aprovação manual e CodeBuild provedores de ações. `manual`



O modo de edição de pipeline, quando visualizado no diagrama do console, permite que você edite substituições de origem, gatilhos e ações, conforme mostrado no exemplo a seguir.



Tópicos

- Declaração de pipeline
- Declaração de estágio
- Declaração de ação

- Provedores de ação válidos em CodePipeline
- Configurações válidas para o parâmetro PollForSourceChanges
- Artefatos de entrada e saída válidos para cada tipo de ação
- Parâmetros de configuração válidos para cada tipo de provedor

Declaração de pipeline

O nível de pipeline e metadados de um pipeline tem uma estrutura básica que inclui os seguintes parâmetros e sintaxe. O parâmetro pipeline representa a estrutura de ações e estágios a serem executados no pipeline.

Para obter mais informações, consulte o PipelineDeclaration objeto no Guia CodePipeline da API.

O exemplo a seguir mostra o nível de pipeline e metadados da estrutura do pipeline em JSON e YAML para um pipeline do 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

O nome do pipeline. Quando você edita ou atualiza um pipeline, o nome do pipeline não pode ser alterado.

> **Note**
>
> Se você deseja renomear um pipeline existente, pode usar o comando `get-pipeline` da CLI para criar um arquivo JSON que contenha a estrutura do pipeline. Depois, você pode usar o comando `create-pipeline` da CLI para criar um pipeline com essa estrutura e dar a ele um novo nome.

roleArn

O ARN do IAM para a função de CodePipeline serviço, como arn:aws:iam: :80398 _. EXAMPLE:role/ CodePipeline Service_Role

Para usar o console e visualizar o ARN do perfil de serviço do pipeline em vez da estrutura JSON, escolha o pipeline no console e selecione Configurações. Na guia Geral, o campo ARN do perfil de serviço é exibido.

Loja de artefatos OU lojas de artefatos

O `artifactStore` campo contém o tipo de bucket de artefatos e a localização de um pipeline com todas as ações na mesma AWS região. Se você adicionar ações em uma região diferente do seu funil, o `artifactStores` mapeamento será usado para listar o bucket de artefatos de cada AWS região em que as ações são executadas. Ao criar ou editar um pipeline, é necessário ter um bucket de artefato na região do pipeline e ter um bucket de artefato por região em que planeja executar uma ação.

> **Note**
>
> Na estrutura do pipeline, você deve incluir `artifactStore` ou `artifactStores` no pipeline, mas não é possível usar os dois. Se você criar uma ação entre regiões em seu pipeline, use `artifactStores`.

O exemplo a seguir mostra a estrutura básica de um pipeline com ações entre regiões que usa o parâmetro `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`

O tipo de localização do bucket de artefatos, especificado como Amazon S3.

`location`

O Amazon S3 gera automaticamente um bucket para você na primeira criação de um pipeline pelo console, como codepipeline-us-east-2-1234567890, ou qualquer bucket do Amazon S3 configurado para essa finalidade

`estágios`

Esse parâmetro contém o nome de cada estágio no pipeline. Para obter mais informações sobre os parâmetros e a sintaxe no nível de estágio da estrutura do pipeline, consulte o StageDeclaration objeto no Guia da CodePipeline API.

A estrutura do pipeline para estágios tem os seguintes requisitos:

- Um pipeline deve ter pelo menos dois estágios.
- O primeiro estágio de um pipeline deve ter pelo menos uma ação de origem. Só pode conter ações de origem.
- Somente o primeiro estágio de um pipeline pode ter ações de origem.
- Pelo menos um estágio em cada pipeline deve ter uma ação que não seja uma ação de origem.

- Os nomes de todos os estágios de um pipeline devem ser exclusivos.
- Os nomes artísticos não podem ser editados no CodePipeline console. Se você editar o nome de um estágio usando o. AWS CLI e o estágio contiver uma ação com um ou mais parâmetros secretos (como um token OAuth), o valor desses parâmetros secretos não será preservado. É necessário inserir manualmente o valor dos parâmetros (que são mascarados por quatro asteriscos no JSON devolvido pela AWS CLI) e incluí-los na estrutura JSON.

> **Important**
>
> Os pipelines inativos por mais de 30 dias terão a sondagem desabilitada para o pipeline. Para obter mais informações, consulte sondagem DisabledAt na referência da estrutura do pipeline. Consulte as etapas de migração do pipeline da sondagem para a detecção de alterações baseada em eventos em Métodos de detecção da alteração.

`version`

O número de versão de um pipeline é gerado e atualizado automaticamente sempre que o pipeline é atualizado.

`executionMode`

Você pode definir o modo de execução do pipeline para poder especificar o comportamento do pipeline para execuções consecutivas, como enfileiramento, substituição ou execução em modo paralelo. Para obter mais informações, consulte Definir ou alterar o modo de execução do pipeline.

> **Important**
>
> Para pipelines em modo PARALLEL, a reversão do estágio não está disponível. Da mesma forma, as condições de falha com um tipo de resultado de reversão não podem ser adicionadas a um pipeline de modo PARALLEL.

`Tipo de tubulação`

O tipo de pipeline especifica a estrutura e os recursos disponíveis no pipeline, como para um pipeline do tipo V2. Para obter mais informações, consulte Tipos de pipeline.

`variables`

As variáveis em nível de pipeline são definidas quando o pipeline é criado e resolvido no runtime do pipeline. Para obter mais informações, consulte Referência de variáveis. Para assistir a um tutorial com uma variável no nível do pipeline que é passada no momento da execução do pipeline, consulte. Tutorial: Usar variáveis no nível do pipeline

`triggers`

Os gatilhos permitem configurar o pipeline para iniciar em um determinado tipo de evento ou tipo de evento filtrado, como quando uma alteração em uma ramificação específica ou solicitação pull é detectada. Os gatilhos são configuráveis para ações de origem com conexões que usam a `CodeStarSourceConnection` ação em CodePipeline GitHub, como Bitbucket e. GitLab Para ter mais informações sobre ações de origem que usam conexões, consulte Adicione provedores de origem terceirizados aos pipelines usando AWS CodeConnections.

Para obter informações e exemplos mais detalhados, consulte Automatizar a inicialização de pipelines usando gatilhos e filtragem.

Para filtragem, padrões de expressão regular no formato glob são compatíveis conforme detalhado em Trabalhar com padrões glob na sintaxe.

> **Note**
>
> As ações de origem CodeCommit e S3 exigem um recurso de detecção de alterações configurado (uma EventBridge regra) ou usam a opção de pesquisar alterações na fonte no repositório. Para pipelines com uma ação de origem do Bitbucket ou do GitHub Enterprise Server, você não precisa configurar um webhook ou usar como padrão a pesquisa. GitHub A ação do Connections gerencia a detecção de alterações para você.

> **Important**
>
> Os pipelines inativos por mais de 30 dias terão a sondagem desabilitada para o pipeline. Para obter mais informações, consulte sondagem DisabledAt na referência da estrutura do pipeline. Consulte as etapas de migração do pipeline da sondagem para a detecção de alterações baseada em eventos em Métodos de detecção da alteração.

Campos de configuração do Git

A configuração do Git para o acionador, inclusive os tipos de eventos e todos os parâmetros para filtragem por ramificações, caminhos de arquivo, tags ou eventos pull request.

Os campos na estrutura JSON são definidos da seguinte forma:

- `sourceActionName`: o nome da ação de origem do pipeline com a configuração do Git.
- `push`: envie eventos com filtragem. Esses eventos usam o operador OU entre diferentes filtros push e o operador E dentro dos filtros.
  - `branches`: as ramificações a serem filtradas. As ramificações usam o operador E entre inclusões e exclusões.
    - `includes`: padrões para filtrar as ramificações que serão incluídas. Inclui o uso de um operador OU.
    - `excludes`: padrões para filtrar as ramificações que serão excluídas. Exclui o uso de um operador OU.
  - `filePaths`: os nomes dos caminhos do arquivo a serem filtrados.
    - `includes`: padrões para filtrar os caminhos de arquivo que serão incluídos. Inclui o uso de um operador OU.
    - `excludes`: padrões para filtrar os caminhos de arquivo que serão excluídos. Exclui o uso de um operador OU.
  - `tags`: os nomes das etiquetas a serem filtradas.
    - `includes`: padrões para filtrar as etiquetas que serão incluídas. Inclui o uso de um operador OU.
    - `excludes`: padrões para filtrar as etiquetas que serão excluídas. Exclui o uso de um operador OU.
- `pullRequest`: eventos de solicitação pull com filtragem de eventos de solicitação pull e filtros de solicitação pull.
  - `events`: filtra eventos de solicitação pull abertos, atualizados ou fechados, conforme especificado.
  - `branches`: As ramificações a serem filtradas. As ramificações usam o operador E entre inclusões e exclusões.
    - `includes`: padrões para filtrar as ramificações que serão incluídas. Inclui o uso de um operador OU.

- `excludes`: padrões para filtrar as ramificações que serão excluídas. Exclui o uso de um operador OU.

- `filePaths`: os nomes dos caminhos do arquivo a serem filtrados.
  - `includes`: padrões para filtrar os caminhos de arquivo que serão incluídos. Inclui o uso de um operador OU.
  - `excludes`: padrões para filtrar os caminhos de arquivo que serão excluídos. Exclui o uso de um operador OU.

Este é um exemplo da configuração do acionador para tipos de evento push e pull request.

```
"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 do tipo de evento para incluir e excluir

O comportamento de inclusão e exclusão dos níveis dos campos de configuração do Git para tipos de evento push é mostrado na seguinte 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 pull request do tipo de evento para inclusão e exclusão

O comportamento de inclusão e exclusão dos níveis dos campos de configuração do Git para tipos de evento pull request é mostrado na seguinte 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)
```

metadata

Os campos de metadados de pipeline são diferentes da estrutura do pipeline e não podem ser editados. Quando você atualiza um pipeline, a data no campo de metadados `updated` é alterada automaticamente.

`pipelineArn`

O nome do recurso da Amazon (ARN) do pipeline.

Para usar o console e visualizar o ARN do pipeline em vez da estrutura JSON, selecione o pipeline no console e selecione Configurações. Na guia Geral, o campo ARN do pipeline é exibido.

`created`

A data e a hora em que o pipeline foi criado.

`updated`

A data e a hora em que o pipeline foi atualizado pela última vez.

`votação DisabledAt`

A data e a hora quando, para um pipeline configurado para sondagem para detecção de alterações, a sondagem foi desativada.

Os pipelines inativos por mais de 30 dias terão a sondagem desabilitada para o pipeline.

- Os pipelines inativos terão a pesquisa desabilitada após 30 dias sem execuções.
- Os pipelines que usam EventBridge CodeStar conexões ou webhooks não serão afetados.
- Os pipelines ativos não serão afetados.

Para obter mais informações, consulte o `pollingDisabledAt` parâmetro em PipelineMetadata objeto no Guia de CodePipeline API. Para obter as etapas a fim de migrar o pipeline da sondagem para a detecção de alterações baseada em eventos, consulte Métodos de detecção da alteração.

Declaração de estágio

O nível de estágio de um pipeline tem uma estrutura básica que inclui os seguintes parâmetros e sintaxe. Para obter mais informações, consulte o StageDeclaration objeto no Guia CodePipeline da API.

O exemplo a seguir mostra o nível de estágio da estrutura do pipeline em JSON e YAML. O exemplo mostra dois estágios chamados `Source` e `Build`. O exemplo contém duas condições, uma para `onSuccess` e outra 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

O nome do estágio do .

actions

O nível de ação de um pipeline tem uma estrutura básica que inclui os seguintes parâmetros e sintaxe. Para ver parâmetros e exemplos, consulte Declaração de ação.

condições

As condições contêm uma ou mais regras que estão disponíveis em uma lista de regras em CodePipeline. Se todas as regras em uma condição forem bem-sucedidas, a condição será atendida. É possível definir condições para que, caso os critérios não sejam cumpridos, o resultado especificado seja ativado.

Você pode configurar os seguintes tipos de condições:

- `beforeEntry`
- `onFailure`

- onSuccess

Para ter mais informações e exemplos, consulte Configurar condições para um estágio.

regras

Toda condição contém um conjunto estruturado de regras, organizadas em uma ordem específica e avaliadas simultaneamente. Portanto, se uma regra falhar na condição, a condição falhará. Você pode substituir as condições da regra no runtime do pipeline.

As regras disponíveis são fornecidas na referência de Regra. Para ter mais informações, consulte a referência da estrutura de regras em Referência da estrutura de regras.

Declaração de ação

O nível de ação de um pipeline tem uma estrutura básica que inclui os seguintes parâmetros e sintaxe. Para obter mais informações, consulte o ActionDeclaration objeto no Guia CodePipeline da API.

O exemplo a seguir mostra o nível de ação da estrutura do pipeline nos formatos JSON e 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 obter uma lista de exemplos de detalhes de `configuration` adequados para o tipo de provedor, consulte Parâmetros de configuração válidos para cada tipo de provedor.

A estrutura de ação tem estes requisitos:

- Os nomes de todas as ações em um estágio devem ser exclusivos.
- É necessária uma ação de origem para cada pipeline.
- As ações de origem que não usam uma conexão podem ser configuradas para detecção de alterações ou para desativar a detecção de alterações. Consulte Métodos de detecção de alterações.
- Isso ocorre com todas as ações, não importa se estão no mesmo estágio ou em estágios seguintes. No entanto, o artefato de entrada não precisa ser a próxima ação na sequência rígida da ação que forneceu o artefato de saída. Ações em paralelo podem declarar pacotes de artefatos de saída diferentes, que, por sua vez, são usados por ações seguintes diferentes.
- Quando você usa um bucket do Amazon S3 como local de implantação, também especifica uma chave de objeto. Uma chave de objeto pode ser um nome de arquivo (objeto) ou uma combinação de um prefixo (caminho da pasta) e um nome de arquivo. Você pode usar variáveis para especificar o nome do local que deseja que o pipeline use. As ações de implantação do Amazon S3 são compatíveis com o uso das variáveis nas chaves de objeto do Amazon S3 a seguir.

Uso de variáveis no Amazon S3

Variável	Exemplo de entrada do console	Saída
datetime	js-application/{datetime}.zip	Carimbo de data e hora UTC neste formato: <AAAA>-<MM>-DD>_<HH>-<MM>-<SS> Exemplo: js- application/2019 -01-10_07-39-57.zip
uuid	js-application/{uuid}.zip	O UUID é um identificador globalmente exclusivo com garantia de diferenciação de qualquer outro identificador. O UUID está neste formato (todos os dígitos no formato hexadecimal): <8-dígitos>-<4-dígitos>-4-dígitos>-<4-dígitos>-<12-dígitos> Exemplo: js- application/54a60075 -b96a-4bf3-9013-db3a9 EXAMPLE.zip

`name`

O nome da ação.

`region`

Para ações em que o provedor é um AWS service (Serviço da AWS), o Região da AWS do recurso.

Cross-Region as ações usam o `Region` campo para designar Região da AWS onde as ações devem ser criadas. Os AWS recursos criados para essa ação devem ser criados na mesma região fornecida no `region` campo. Não é possível criar ações entre regiões para os seguintes tipos de ação:

- Ações de origem

- Ações por provedores de terceiros
- Ações por provedores personalizados

roleArn

O ARN da função de serviço do IAM que executa a ação declarada. Isso é assumido por meio do roleArn especificado no nível do pipeline.

namespace

As ações podem ser configuradas com variáveis. Use o campo `namespace` para definir as informações de namespace e de variáveis para as variáveis de execução. Para obter informações de referência sobre variáveis de execução e variáveis de saída de ação, consulte Referência de variáveis.

> **Note**
>
> Para Amazon ECR, Amazon S3 ou CodeCommit fontes, você também pode criar uma substituição de origem usando a entrada de transformação de entrada para usar o `revisionValue` in EventBridge para seu evento de pipeline, em que `revisionValue` é derivado da variável de evento de origem para sua chave de objeto, confirmação ou ID de imagem. Para saber mais, consulte a etapa opcional da entrada da transformação de entrada incluída nos procedimentos em Ações e recursos de origem do Amazon ECR EventBridge, Conexão a ações de origem do Amazon S3 com uma fonte habilitada para eventos ou CodeCommit ações de origem e EventBridge.

ação TypeId

O ID do tipo de ação é identificado como uma combinação dos quatro campos a seguir.

categoria

O tipo de etapa ou ação dentro do pipeline, como, por exemplo, uma ação de origem. Cada tipo de ação tem um conjunto específico de provedores de ações válidos. Para conferir uma lista de provedores válidos por tipo de ação, consulte Referência da estrutura da ação.

Estas são as `actionTypeId` categorias válidas (tipos de ação) para CodePipeline:

- `Source`
- `Build`
- `Approval`
- `Deploy`
- `Test`
- `Invoke`
- `Compute`

`proprietário`

Para todos os tipos de ação compatíveis no momento, a única string proprietária válida é `AWS`, `ThirdParty` ou `Custom`. Para encontrar a string de proprietário válida para uma ação específica, consulte Referência da estrutura da ação.

Para obter mais informações, consulte a Referência da API do CodePipeline .

`version`

A versão da ação.

`provider`

O provedor da ação, como CodeBuild.

- Os tipos válidos de provedor de uma categoria de ação dependem da categoria. Por exemplo, para um tipo de ação de origem, um tipo de provedor válido é `S3`, `CodeStarSourceConnection`, `CodeCommit` ou `Amazon ECR`. Esse exemplo mostra a estrutura para uma ação de origem com um provedor `S3`:

```
"actionTypeId": {
  "category": "Source",
  "owner": "AWS",
  "version": "1",
  "provider": "S3"},
```

InputArtifacts

Esse campo contém a estrutura do artefato de entrada, se for compatível com a categoria de ação. O artefato de entrada de uma ação deve ser exatamente igual ao artefato de saída declarado em uma ação precedente. Por exemplo, se uma ação anterior inclui a seguinte declaração:

```
"outputArtifacts": [
    {
    "MyApp"
    }
],
```

e não há outros artefatos de saída, o artefato de entrada de uma ação seguinte deve ser:

```
"inputArtifacts": [
    {
    "MyApp"
    }
],
```

Por exemplo, uma ação de origem não pode ter artefatos de entrada porque é a primeira ação no pipeline. No entanto, uma ação de origem sempre terá artefatos de saída que são processados pela ação a seguir. Os artefatos de saída para uma ação de origem são os arquivos do aplicativo do repositório de origem, compactados e fornecidos por meio do bucket de artefatos, que são processados pela ação a seguir, como uma CodeBuild ação que atua nos arquivos do aplicativo com comandos de construção.

Como exemplo de ações que não podem ter artefatos de saída, as ações de implantação não têm artefatos de saída porque essas ações geralmente são a última ação em um pipeline.

name

O nome do artefato para os artefatos de entrada da ação.

Artefatos de saída

Os nomes dos artefatos de saída devem ser exclusivos em um pipeline. Por exemplo, um pipeline pode incluir uma ação que tenha um artefato de saída com o nome `"MyApp"` e outra ação que tenha um artefato de saída com o nome `"MyBuiltApp"`. Mas um pipeline não pode incluir duas ações que tenham um artefato de saída com o nome `"MyApp"`.

Esse campo contém a estrutura do artefato de saída, se for compatível com a categoria de ação. O artefato de saída de uma ação deve ser exatamente igual ao artefato de saída declarado em uma ação anterior. Por exemplo, se uma ação anterior inclui a seguinte declaração:

```
"outputArtifacts": [
    {
    "MyApp"
    }
],
```

e não há outros artefatos de saída, o artefato de entrada de uma ação seguinte deve ser:

```
"inputArtifacts": [
    {
    "MyApp"
    }
],
```

Por exemplo, uma ação de origem não pode ter artefatos de entrada porque é a primeira ação no pipeline. No entanto, uma ação de origem sempre terá artefatos de saída que são processados pela ação a seguir. Os artefatos de saída para uma ação de origem são os arquivos do aplicativo do repositório de origem, compactados e fornecidos por meio do bucket de artefatos, que são processados pela ação a seguir, como uma CodeBuild ação que atua nos arquivos do aplicativo com comandos de construção.

Como exemplo de ações que não podem ter artefatos de saída, as ações de implantação não têm artefatos de saída porque essas ações geralmente são a última ação em um pipeline.

`name`

O nome do artefato para os artefatos de saída da ação.

`configuração` (por provedor de ação)

A configuração da ação contém detalhes e parâmetros apropriados ao tipo de provedor. Na seção abaixo, os parâmetros de configuração da ação de exemplo são específicos da ação de origem do S3.

A configuração da ação e os limites do input/output artefato podem variar de acordo com o provedor da ação. Para conferir uma lista de exemplos de configuração de ação por provedor de ações,

consulte Referência da estrutura da ação e a tabela em Parâmetros de configuração válidos para cada tipo de provedor. A tabela fornece um link para a referência de ação de cada tipo de provedor, que mostra os parâmetros de configuração para cada ação em detalhes. Para conferir uma tabela com os limites de artefatos de entrada e saída para cada provedor de ações, consulte Artefatos de entrada e saída válidos para cada tipo de ação.

As considerações a seguir se aplicam ao trabalho com ações:

- As ações de origem não têm artefatos de entrada, e as ações de implantação não têm artefatos de saída.
- Para provedores de ações de origem que não usam uma conexão, como o S3, você deve usar o parâmetro `PollForSourceChanges` para especificar se deseja que o pipeline seja iniciado automaticamente quando uma alteração for detectada. Consulte Configurações válidas para o parâmetro `PollForSourceChanges`.
- Para configurar a detecção automática de alterações para iniciar o pipeline ou para desativar a detecção de alterações, consulte Ações de origem e métodos de detecção de alterações.
- Para configurar gatilhos com filtragem, use a ação de origem para conexões e, em seguida, consulte Automatizar a inicialização de pipelines usando gatilhos e filtragem.
- Para conhecer as variáveis de saída de cada ação, consulte Referência de variáveis.

> **Note**
>
> Para Amazon ECR, Amazon S3 ou CodeCommit fontes, você também pode criar uma substituição de origem usando a entrada de transformação de entrada para usar o `revisionValue` in EventBridge para seu evento de pipeline, em que `revisionValue` é derivado da variável de evento de origem para sua chave de objeto, confirmação ou ID de imagem. Para obter mais informações, consulte a etapa opcional da entrada da transformação de entrada incluída nos procedimentos em Ações e recursos de origem do Amazon ECR EventBridge, Conexão a ações de origem do Amazon S3 com uma fonte habilitada para eventos ou CodeCommit ações de origem e EventBridge.

> **Important**
>
> Os pipelines inativos por mais de 30 dias terão a sondagem desabilitada para o pipeline. Para obter mais informações, consulte sondagem DisabledAt na referência da estrutura

do pipeline. Consulte as etapas de migração do pipeline da sondagem para a detecção de alterações baseada em eventos em Métodos de detecção da alteração.

> **Note**
>
> As ações de origem CodeCommit e S3 exigem um recurso de detecção de alterações configurado (uma EventBridge regra) ou usam a opção de pesquisar alterações na fonte no repositório. Para pipelines com uma ação de origem do Bitbucket ou do GitHub Enterprise Server, você não precisa configurar um webhook ou usar como padrão a pesquisa. GitHub A ação do Connections gerencia a detecção de alterações para você.

`Ordem de execução`

Um número inteiro positivo que indica a ordem de execução da ação dentro do estágio. As ações paralelas no estágio são mostradas como tendo o mesmo número inteiro. Por exemplo, duas ações com uma ordem de execução de duas serão executadas paralelamente após a execução da primeira ação na etapa.

O valor `runOrder` padrão de uma ação é 1. O valor deve ser um inteiro positivo (número natural). Não é permitido usar frações, números decimais ou negativos ou o número zero. Para especificar uma sequência de ações em série, use o menor número para a primeira ação e os maiores números para cada uma das ações em sequência restantes. Para especificar ações paralelas, use o mesmo número inteiro para cada ação que deseja executar em paralelo. No console, você pode especificar uma sequência serial para uma ação escolhendo Adicionar grupo de ações no nível do estágio em que deseja que ela seja executada ou você pode especificar uma sequência paralela escolhendo Adicionar ação. Grupo de ações se refere a uma ordem de execução de uma ou mais ações no mesmo nível.

Por exemplo, se você deseja que três ações sejam executadas em sequência em um estágio, você daria o valor `runOrder` de 1 para a primeira ação, o valor `runOrder` de 2 para a segunda ação e o valor `runOrder` de 3 para a terceira ação. No entanto, se você deseja que a segunda ação e a terceira ação sejam executadas em paralelo, você daria o valor `runOrder` de 1 para a primeira ação e o valor `runOrder` de 2 para a segunda e a terceira.

> **Note**
>
> A numeração das ações seriais não precisa estar em sequência estrita. Por exemplo, se você tem três ações em uma sequência e decide remover a segunda ação, você não precisa numerar novamente o valor `runOrder` da terceira ação. Como o valor `runOrder` dessa ação (3) é superior ao valor `runOrder` da primeira ação (1), ele será executado em série após a primeira ação no estágio.

Provedores de ação válidos em CodePipeline

O formato da estrutura de pipeline é usado para compilar ações e estágios em um pipeline. Um tipo de ação é composto por uma categoria de ação e um tipo de provedor.

Cada categoria de ação tem uma lista válida de provedores de ações. Para fazer referência aos provedores de ações válidos para cada categoria de ação, consulte Referência da estrutura da ação.

Cada categoria de ação tem um conjunto designado de provedores. Cada provedor de ação, como o Amazon S3, tem um nome de provedor, como `S3`, que deve ser usado no campo `Provider` na categoria de ação da estrutura do pipeline.

Há três valores válidos para o campo `Owner` na seção da categoria de ação na estrutura do pipeline: `AWS`, `ThirdParty` e `Custom`.

Para localizar o nome do provedor e as informações do proprietário do provedor de ações, consulte Referência da estrutura da ação ou Artefatos de entrada e saída válidos para cada tipo de ação.

Essa tabela lista os provedores válidos por tipo de ação.

> **Note**
>
> Para ações do Bitbucket ou GitHub Enterprise Server, consulte o tópico de referência da CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com ação. GitHub

Provedores de ação válidos por tipo de ação

Categoria de ação	Provedores de ação válidos	Tipo de pipeline compatível	Referência da ação
Fonte	Amazon S3	V1, V2	Referência da ação de origem do Amazon S3
	Amazon ECR	V1, V2	Referência da ação de origem do Amazon ECR
	CodeCommit	V1, V2	CodeCommit referência de ação de origem
	CodeStarSourceConnection (para ações do Bitbucket GitHub, GitHub Enterprise Server)	V1, V2	CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com
Criar	Ação `ECRBuildAndPublish` do Amazon ECR	Somente V2	Referência da ação de compilação `ECRBuildAndPublish`
	CodeBuild	V1, V2	AWS CodeBuild referência de ação de construção e teste
	Ação de comandos (consulte Computação)	Somente V2	
	Personalizado CloudBees	V1, V2	Artefatos de entrada e saída válidos para cada tipo de ação
	Jenkins personalizado	V1, V2	Artefatos de entrada e saída válidos para cada tipo de ação

Categoria de ação	Provedores de ação válidos	Tipo de pipeline compatível	Referência da ação
	Personalizado TeamCity	V1, V2	Artefatos de entrada e saída válidos para cada tipo de ação
Testar	CodeBuild	V1, V2	AWS CodeBuild referência de ação de construção e teste
	AWS Device Farm	V1, V2	Artefatos de entrada e saída válidos para cada tipo de ação
	Personalizado BlazeMeter	V1, V2	Artefatos de entrada e saída válidos para cada tipo de ação
	ThirdParty GhostInspector		Artefatos de entrada e saída válidos para cada tipo de ação
	Jenkins personalizado		Artefatos de entrada e saída válidos para cada tipo de ação
	ThirdParty StormRunner Carga Micro Focus		Artefatos de entrada e saída válidos para cada tipo de ação
	ThirdParty Nouvola		Artefatos de entrada e saída válidos para cada tipo de ação
	ThirdParty Runscope		Artefatos de entrada e saída válidos para cada tipo de ação

Categoria de ação	Provedores de ação válidos	Tipo de pipeline compatível	Referência da ação
Implantar	Amazon S3		Referência da ação de implantação do Amazon S3
	CloudFormation		Referência da ação de implantação do CloudFormation
	CodeDeploy		Artefatos de entrada e saída válidos para cada tipo de ação
	Ação invocar do EC2	Somente V2	Referência de ação do Amazon EC2
	Amazon ECS		Artefatos de entrada e saída válidos para cada tipo de ação
	Amazon ECS (Blue/Green) (esta é a `CodeDeployToECS` ação)		Artefatos de entrada e saída válidos para cada tipo de ação
	Ação do Amazon EKS	Somente V2	Referência de ação de implantação do Amazon Elastic Kubernetes Service EKS
	Pé de feijão elástico		Artefatos de entrada e saída válidos para cada tipo de ação
	AWS AppConfig		Referência da ação de implantação do AWS AppConfig

Categoria de ação	Provedores de ação válidos	Tipo de pipeline compatível	Referência da ação
	OpsWorks		Artefatos de entrada e saída válidos para cada tipo de ação
	Service Catalog		Artefatos de entrada e saída válidos para cada tipo de ação
	Amazon Alexa		Artefatos de entrada e saída válidos para cada tipo de ação
	Personalizado XebiaLabs		Artefatos de entrada e saída válidos para cada tipo de ação
Aprovação	Manual		Artefatos de entrada e saída válidos para cada tipo de ação
Invocar	CodePipelineInvocar ação		AWS CodePipeline invocar referência de ação
	AWS Lambda		AWS Lambda invocar referência de ação
	AWS Step Functions		AWS Step Functions invocar referência de ação
	InspectorScan		Referência da ação invocar `InspectorScan` do Amazon Inspector

Categoria de ação	Provedores de ação válidos	Tipo de pipeline compatível	Referência da ação
Computação	Ação Commands		Referência da ação Commands

Alguns tipos de ação CodePipeline estão disponíveis somente em AWS regiões selecionadas. É possível que um tipo de ação esteja disponível em uma AWS região, mas um AWS provedor para esse tipo de ação não esteja disponível.

Para obter mais informações sobre cada provedor de ações, consulte Integrações com tipos de CodePipeline ação.

Configurações válidas para o parâmetro `PollForSourceChanges`

O padrão do parâmetro `PollForSourceChanges` é determinado pelo método usado para criar o pipeline, conforme descrito na tabela a seguir. Em muitos casos, o parâmetro `PollForSourceChanges` é padronizado como verdadeiro e deve ser desativado.

Quando o parâmetro `PollForSourceChanges` for padronizado como verdadeiro, faça o seguinte:

- Adicione o parâmetro `PollForSourceChanges` ao arquivo JSON ou ao modelo do CloudFormation .
- Crie recursos de detecção de alterações (regra de CloudWatch eventos, conforme aplicável).
- Defina o parâmetro `PollForSourceChanges` para false.

> **Note**
>
> Se você criar uma regra de CloudWatch eventos ou webhook, deverá definir o parâmetro como false para evitar acionar o pipeline mais de uma vez.

O parâmetro `PollForSourceChanges` não é usado em ações de origem do Amazon ECR.

- Padrões de parâmetros **`PollForSourceChanges`**

Fonte	Método de criação	Exemplo de saída da estrutura JSON de "configuração"
CodeCommit	O pipeline é criado com o console (e recursos de detecção de alterações são criados pelo console). O parâmetro é exibido na saída da estrutura de pipeline e assume `false` como padrão.	`BranchName": "main", "PollForSourceChanges": "false", "RepositoryName": "my-repo"`
	O pipeline é criado com a CLI ou CloudFormation, e o `PollForSourceChanges` parâmetro não é exibido na saída JSON, mas é definido como .² `true`	`BranchName": "main", "RepositoryName": "my-repo"`
Amazon S3	O pipeline é criado com o console (e recursos de detecção de alterações são criados pelo console). O parâmetro é exibido na saída da estrutura de pipeline e assume `false` como padrão.	`"S3Bucket": "my-bucket", "S3ObjectKey": "object.zip", "PollForSourceChanges": "false"`
	O pipeline é criado com a CLI ou CloudFormation, e o `PollForSourceChanges` parâmetro não é exibido na saída JSON, mas é definido como .² `true`	`"S3Bucket": "my-bucket", "S3ObjectKey": "object.zip"`
GitHub	O pipeline é criado com o console (e recursos de detecção de alterações são criados pelo console). O parâmetro é exibido na saída da estrutura de pipeline e assume `false` como padrão.	`"Owner": "MyGitHubAccountName ", "Repo": " MyGitHubRepositoryName " "PollForSourceChanges": "false", "Branch": " main" "OAuthToken": " ****"`

Fonte	Método de criação	Exemplo de saída da estrutura JSON de "configur ação"
	O pipeline é criado com a CLI ou CloudForm ation, e o `PollForSourceChanges` parâmetro não é exibido na saída JSON, mas é definido como .[2] `true`	`"Owner": "MyGitHubA ccountName ", "Repo": "MyGitHubR epositoryName ", "Branch": " main", "OAuthToken": " ****"`

[2] Se `PollForSourceChanges` tiver sido adicionado em qualquer ponto à estrutura JSON ou ao CloudFormation modelo, ele será exibido conforme mostrado:

```
"PollForSourceChanges": "true",
```

[3] Para obter informações sobre os recursos de detecção de alterações que se aplicam a cada provedor de origem, consulte Métodos de detecção de alterações.

Artefatos de entrada e saída válidos para cada tipo de ação

Dependendo do tipo de ação e do provedor, você pode ter o seguinte número de artefatos de entrada e saída.

Limitações dos tipos de ação dos artefatos

Proprietário	Tipo de ação	Fornecedor	Número válido de artefatos de entrada	Número válido de artefatos de saída
AWS	Fonte	S3	0	1
AWS	Fonte	CodeCommit	0	1
AWS	Fonte	ECR	0	1

Proprietário	Tipo de ação	Fornecedor	Número válido de artefatos de entrada	Número válido de artefatos de saída
ThirdParty	Fonte	CodeStarSourceConnection	0	1
AWS	Criar	CodeBuild	1 – 5	0 – 5
AWS	Testar	CodeBuild	1 – 5	0 – 5
AWS	Testar	DeviceFarm	1	0
AWS	Aprovação	ThirdParty	0	0
AWS	Implantar	S3	1	0
AWS	Implantar	CloudFormation	0 – 10	0 – 1
AWS	Implantar	CodeDeploy	1	0
AWS	Implantar	ElasticBeanstalk	1	0
AWS	Implantar	OpsWorks	1	0
AWS	Implantar	ECS	1	0
AWS	Implantar	ServiceCatalog	1	0
AWS	Invocar	Lambda	0 – 5	0 – 5
ThirdParty	Implantar	AlexaSkillsKit	1 – 2	0
Custom	Criar	Jenkins	0 – 5	0 – 5

Proprietário	Tipo de ação	Fornecedor	Número válido de artefatos de entrada	Número válido de artefatos de saída
`Custom`	Testar	`Jenkins`	0 – 5	0 – 5
`Custom`	Qualquer categoria compatível	Conforme especificado na ação personalizada	0 – 5	0 – 5

Parâmetros de configuração válidos para cada tipo de provedor

Essa seção lista parâmetros de `configuration` válidos para cada provedor de ações.

Toda ação deve ter uma configuração válida de ação, que depende do tipo de provedor para a ação em questão. A tabela a seguir mostra os elementos de configuração de ação necessários para cada tipo de provedor válido:

Propriedades de configuração de ação para tipos de provedor

Nome do provedor	Nome do provedor no tipo de ação	Propriedades de configuração	Required/Optional
Amazon S3 (provedor de ação de implantação)	Para obter mais informações, inclusive exemplos relacionados aos parâmetros de ação de implantação do Amazon S3, consulte Referência da ação de implantação do Amazon S3.		
Amazon S3 (provedor de ação de origem)	Para obter mais informações, incluindo exemplos relacionados a parâmetros de ação de origem do Amazon S3, consulte Referência da ação de origem do Amazon S3.		
Amazon ECR	Para obter mais informações, incluindo exemplos relacionados a parâmetros do Amazon ECR, consulte Referência da ação de origem do Amazon ECR.		

Nome do provedor	Nome do provedor no tipo de ação	Propriedades de configuração	Required/Optional
CodeCommit	Para obter mais informações, incluindo exemplos relacionados aos CodeCommit parâmetros, consulteCodeCommit referência de ação de origem.		
CodeStarSourceConnection ação para Bitbucket, GitHub (via GitHub aplicativo), GHES e GitLab	Para obter mais informações, inclusive exemplos da configuração de ação, consulte Parâmetros de configuração.		
GitHub (por meio do aplicativo OAuth)	Para obter mais informações, incluindo exemplos relacionados aos GitHub parâmetros, consulteGitHub (via aplicativo OAuth) referência de ação de origem. Essa é a GitHub ação da Versão 1.		
AWS CloudFormation	Para obter mais informações, incluindo exemplos relacionados aos AWS CloudFormation parâmetros, consulteReferência da ação de implantação do CloudFormation.		
CodeBuild	Para obter mais descrições e exemplos relacionados aos CodeBuild parâmetros, consulteAWS CodeBuild referência de ação de construção e teste.		
CodeDeploy	Para obter mais descrições e exemplos relacionados aos CodeDeploy parâmetros, consulteReferência da ação de implantação do AWS CodeDeploy.		
AWS Device Farm	Para obter mais descrições e exemplos relacionados aos AWS Device Farm parâmetros, consulteAWS Device Farm referência de ação de teste.		
AWS Elastic Beanstalk	`ElasticBeanstalk`	`ApplicationName`	Obrigatório
		`EnvironmentName`	Obrigatório

Nome do provedor	Nome do provedor no tipo de ação	Propriedades de configuração	Required/Optional
AWS Lambda	Para obter mais informações, incluindo exemplos relacionados aos AWS Lambda parâmetros, consulteAWS Lambda invocar referência de ação.		
AWS OpsWorks Stacks	`OpsWorks`	`Stack`	Obrigatório
		`Layer`	Opcional
		`App`	Obrigatório
Amazon ECS	Para obter mais descrição e exemplos relacionados a parâmetros do Amazon ECS, consulte Referência da ação de implantação do Amazon Elastic Container Service.		
Amazon ECS e CodeDeploy () Blue/Green	Para obter mais descrições e exemplos relacionados ao Amazon ECS e aos CodeDeploy blue/green parâmetros, consulteAmazon Elastic Container Service e referência de ação de implantação CodeDeploy azul-esverdeada.		
Service Catalog	`ServiceCatalog`	`TemplateFilePath`	Obrigatório
		`ProductVersionName`	Obrigatório
		`ProductType`	Obrigatório
		`ProductVersionDescription`	Opcional
		`ProductId`	Obrigatório
Alexa Skills Kit	`AlexaSkillsKit`	`ClientId`	Obrigatório
		`ClientSecret`	Obrigatório
		`RefreshToken`	Obrigatório
		`SkillId`	Obrigatório

Nome do provedor	Nome do provedor no tipo de ação	Propriedades de configuração	Required/Optional
Jenkins	O nome da ação que você forneceu no CodePipeline Plugin para Jenkins (por exemplo, *MyJenkinsProviderName*)	`ProjectName`	Obrigatório
Aprovação manual	`Manual`	`CustomData`	Opcional
		`ExternalEntityLink`	Opcional
		`NotificationArn`	Opcional

O exemplo a seguir mostra uma configuração válida para uma ação de implantação que usa o Alexa Skills Kit:

```
"configuration": {
  "ClientId": "amzn1.application-oa2-client.aadEXAMPLE",
  "ClientSecret": "****",
  "RefreshToken": "****",
  "SkillId": "amzn1.ask.skill.22649d8f-0451-4b4b-9ed9-bfb6cEXAMPLE"
}
```

O exemplo a seguir mostra uma configuração válida para uma aprovação manual:

```
"configuration": {
  "CustomData": "Comments on the manual approval",
  "ExternalEntityLink": "http://my-url.com",
  "NotificationArn": "arn:aws:sns:us-west-2:12345EXAMPLE:Notification"
}
```

Referência da estrutura da ação

Esta seção é uma referência somente para a configuração de ações. Para obter uma visão geral conceitual da estrutura do pipeline, consulte CodePipeline referência de estrutura de tubulação.

Cada provedor de ação CodePipeline usa um conjunto de campos de configuração obrigatórios e opcionais na estrutura do pipeline. Esta seção fornece as seguintes informações de referência por provedor de ações:

- Valores válidos para os campos `ActionType` incluídos no bloco de ação da estrutura do pipeline, como `Owner` e `Provider`.
- Descrições e outras informações de referência para os parâmetros de `Configuration` (obrigatórios e opcionais) incluídos na seção de ação da estrutura do pipeline.
- Exemplo de campos de ação JSON e YAML válidos.

Esta seção é atualizada periodicamente com mais provedores de ações. No momento, as informações de referência estão disponíveis para os seguintes provedores de ações:

Tópicos

- Referência de ação do Amazon EC2
- Referência da ação de origem do Amazon ECR
- Referência da ação de compilação ECRBuildAndPublish
- Amazon Elastic Container Service e referência de ação de implantação CodeDeploy azul-esverdeada
- Referência da ação de implantação do Amazon Elastic Container Service
- Referência de ação de implantação do Amazon Elastic Kubernetes Service EKS
- AWS Lambda Referência da ação de implantação do
- Referência da ação de implantação do Amazon S3
- Referência da ação de origem do Amazon S3
- Referência da ação de implantação do AWS AppConfig
- Referência da ação de implantação do CloudFormation
- CloudFormation Referência da ação de implantação do StackSets
- AWS CodeBuild referência de ação de construção e teste

- AWS CodePipeline invocar referência de ação
- CodeCommit referência de ação de origem
- Referência da ação de implantação do AWS CodeDeploy
- CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com
- Referência da ação Commands
- AWS Device Farm referência de ação de teste
- Referência da ação de implantação do Elastic Beanstalk
- Referência da ação invocar InspectorScan do Amazon Inspector
- AWS Lambda invocar referência de ação
- AWS OpsWorks Referência da ação de implantação do
- AWS Service Catalog Referência da ação de implantação do
- AWS Step Functions invocar referência de ação

Referência de ação do Amazon EC2

Você usa uma ação EC2 do Amazon EC2 para implantar o código do aplicativo à frota de implantação. Sua frota de implantação pode consistir em instâncias Linux ou SSM-managed nós Linux do Amazon EC2. As instâncias devem ter o atendente do SSM instalado.

> **Note**
>
> Esta ação oferece suporte somente para tipos de instância do Linux. O tamanho máximo da frota compatível é de 500 instâncias.

A ação escolherá várias instâncias com base em um máximo especificado. As instâncias com falha das instâncias anteriores serão escolhidas primeiro. A ação vai ignorar a implantação em determinadas instâncias se a instância já tiver recebido a implantação do mesmo artefato de entrada, como um caso no qual a ação falhou anteriormente.

> **Note**
>
> Essa ação só é compatível para pipelines tipo V2.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Permissões de política do perfil de serviço para a ação de implantação do EC2
- Implantar referência do arquivo de especificação
- Declaração de ação
- Declaração de ação com exemplo de especificação Implantar
- Consulte também

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `EC2`
- Versão: `1`

Parâmetros de configuração

InstanceTagKey

Obrigatório: Sim

A chave de tag das instâncias que você criou no Amazon EC2, como `Name`.

InstanceTagValue

Obrigatório: não

O valor de tag das instâncias criadas por você no Amazon EC2, como `my-instances`.

Quando esse valor não for especificado, todas as instâncias com InstanceTagKey serão correspondidas.

InstanceType

Obrigatório: Sim

O tipo de instâncias ou nós SSM criados no Amazon EC2. Os valores válidos são `EC2` e `SSM_MANAGED_NODE`.

Você já deve ter criado, marcado e instalado o atendente SSM em todas as instâncias.

> **Note**
>
> Ao criar a instância, você cria ou usa uma função de instância do EC2 existente. Para evitar `Access Denied` erros, você deve adicionar permissões de bucket do S3 à função de instância para dar à instância permissões para o bucket de CodePipeline artefatos. Crie uma função padrão ou atualize a função existente com a permissão `s3:GetObject` limitada ao bucket de artefato da região do pipeline.

TargetDirectory

Obrigatório: sim (se o script for especificado)

O diretório a ser usado na instância do Amazon EC2 para executar scripts.

DeploySpec

Obrigatório: sim (se a especificação da implantação for especificada)

O arquivo a ser usado para configurar eventos de instalação e ciclo de vida da implantação. Para ver descrições do campo de especificação e informações, consulte Implantar referência do arquivo de especificação. Para exibir uma configuração de ação com o arquivo de especificação de implantação especificado, consulte o exemplo em Declaração de ação com exemplo de especificação Implantar.

MaxBatch

Obrigatório: não

O número máximo de instâncias permitidas para implantação em paralelo.

MaxError

Obrigatório: não

O número máximo de erros de instância permitidos durante a implantação.

TargetGroupNameList

Obrigatório: não

A lista de nomes do grupo de destino para implantação. Você já deve ter criado os grupos de destino.

Os grupos de destino oferecem um conjunto de instâncias para processar solicitações específicas. Se o grupo de destino for especificado, as instâncias serão removidas do grupo de destino antes da implantação e readicionadas ao grupo de destino depois da implantação.

PreScript

Obrigatório: não

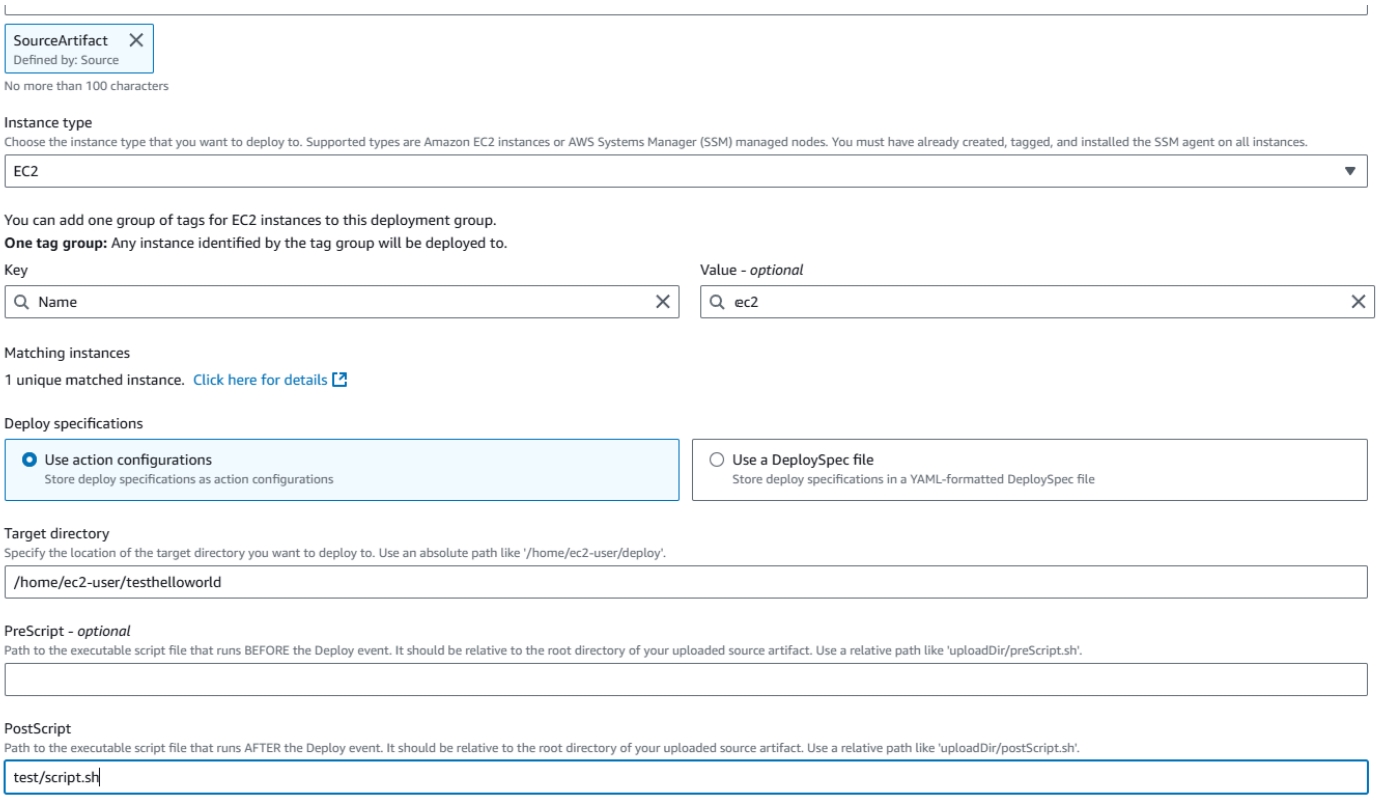
O script a ser executado antes da fase Implantar da ação.

PostScript

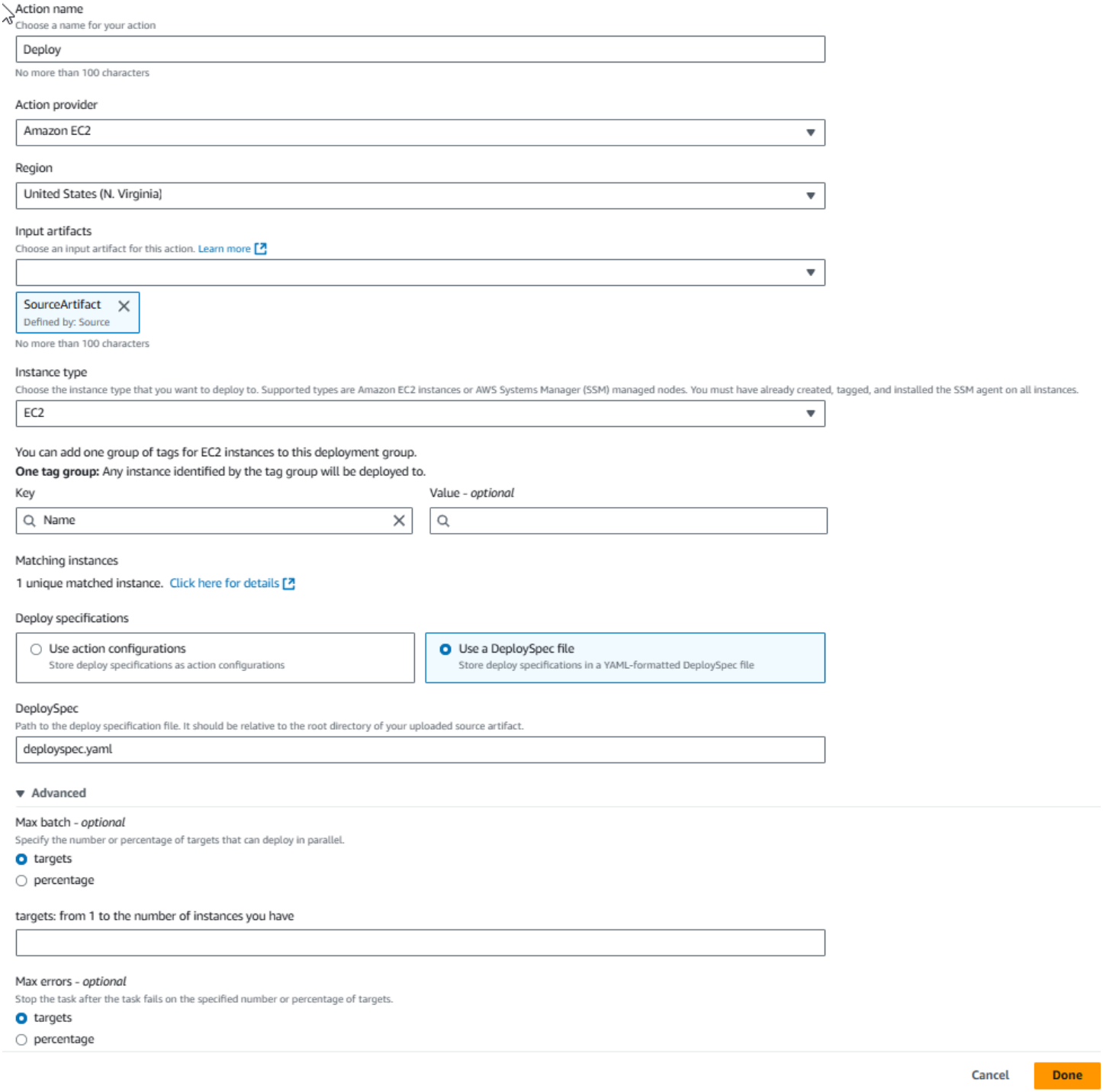
Obrigatório: Sim

O script a ser executado depois da fase Implantar da ação.

A imagem a seguir mostra um exemplo da página Editar da ação na qual Usar configurações de ação é escolhida.



A imagem a seguir mostra um exemplo da página Editar para a ação em que Usar um DeploySpec arquivo é escolhido.



Input artifacts (Artefatos de entrada)

- Número de artefatos: 1

- Descrição: os arquivos fornecidos, se houver, para oferecer suporte às ações de script durante a implantação.

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Permissões de política do perfil de serviço para a ação de implantação do EC2

Quando CodePipeline executa a ação, a função CodePipeline de serviço exige as seguintes permissões, com escopo adequado para acesso com o mínimo de privilégios.

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"
            ]
        }
    ]
}
```

Grupos de registros para seu funil em CloudWatch registros

Quando CodePipeline executa a ação, CodePipeline cria um grupo de registros usando o nome do pipeline da seguinte forma. Isso permite reduzir as permissões para os recursos de log usando o nome do pipeline.

```
/aws/codepipeline/MyPipelineName
```

As seguintes permissões do registro em log estão incluídas nas atualizações acima para o perfil de serviço.

- registros: CreateLogGroup
- registros: CreateLogStream
- registros: PutLogEvents

Para visualizar logs no console usando a página de diálogo de detalhes da ação, a permissão para visualizar logs deve ser adicionada ao perfil do console. Para ter mais informações, consulte o exemplo de política de permissões do console em Permissões necessárias para visualizar registros computacionais no console.

Permissões da política de função de serviço para CloudWatch registros

Quando CodePipeline executa a ação, CodePipeline cria um grupo de registros usando o nome do pipeline da seguinte forma. Isso permite reduzir as permissões para os recursos de log usando o nome do pipeline.

```
/aws/codepipeline/MyPipelineName
```

Para visualizar logs no console usando a página de diálogo de detalhes da ação, a permissão para visualizar logs deve ser adicionada ao perfil do console. Para ter mais informações, consulte o exemplo de política de permissões do console em Permissões necessárias para visualizar registros computacionais no console.

Implantar referência do arquivo de especificação

Ao CodePipeline executar a ação, você pode especificar um arquivo de especificação para configurar a implantação em suas instâncias. O arquivo de especificação da implantação especifica o que instalar e quais hooks de evento do ciclo de vida executar em resposta a eventos do ciclo de vida da implantação. O arquivo de especificação de implantação é sempre YAML-formatted. O arquivo de especificação da implantação é usado para:

- Mapear os arquivos de origem na sua revisão de aplicativo para os respectivos destinos na instância.
- Especificar permissões personalizadas para arquivos implantados.
- Especificar scripts a serem executados em cada instância, em vários estágios do processo de implantação.

O arquivo de especificações de implantação oferece suporte a parâmetros específicos de configuração de implantação suportados pelo CodeDeploy AppSpec arquivo. Você pode usar seu AppSpec arquivo existente diretamente, e quaisquer parâmetros não suportados serão ignorados. Para obter mais informações sobre o AppSpec arquivo em CodeDeploy, consulte a referência do arquivo de especificação do aplicativo no Guia CodeDeploy do usuário.

Os parâmetros de implantação do arquivo são especificados da seguinte forma:

- `files`: o arquivo de especificação da implantação designa `source:` e `destination:` para os arquivos de implantação.
- `scripts`: os eventos com script para a implantação. Dois eventos são compatíveis: `BeforeDeploy` e `AfterDeploy`.
- `hooks`: os ganchos do ciclo de vida do evento. Os seguintes hooks são compatíveis: `ApplicationStop`, `BeforeInstall`, `AfterInstall`, `ApplicationStart` e `ValidateService`.

> **Note**
>
> O parâmetro hooks está disponível para AppSpec compatibilidade com CodeDeploy e só está disponível na versão 0.0 (AppSpec formato). Para este formato, CodePipeline realizará um mapeamento da melhor forma possível dos eventos.

O espaçamento YAML correto deve ser usado no arquivo de especificação. Do contrário, um erro não será lançado se os locais e o número de espaços em um arquivo de especificação da implantação não estiverem corretos. Para obter mais informações sobre espaçamento, consulte a especificação YAML.

Um arquivo de especificação da implantação de exemplo está abaixo.

```
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 exibir uma configuração de ação com o arquivo de especificação de implantação especificado, consulte o exemplo em Declaração de ação com exemplo de especificação Implantar.

Declaração de ação

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"
        }
    ]
},
```

Declaração de ação com exemplo de especificação Implantar

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"
        }
    ]
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Tutorial: Implante em instâncias do Amazon EC2 com CodePipeline— Este tutorial orienta você na criação de instâncias do EC2 nas quais você implantará um arquivo de script, junto com a criação do pipeline usando a ação do EC2.
- A ação de implantação do EC2 falha com uma mensagem de erro `Não existe tal arquivo`: este tópico descreve a solução de problemas dos erros de arquivo não encontrado com a ação do EC2.

Referência da ação de origem do Amazon ECR

Aciona o pipeline quando uma nova imagem é enviada por push para o repositório do Amazon ECR. Essa ação fornece um arquivo de definições de imagem que faz referência ao URI da imagem que foi enviada por push ao Amazon ECR. Essa ação de origem geralmente é usada em conjunto com outra ação de origem, como CodeCommit, para permitir um local de origem para todos os outros artefatos de origem. Para obter mais informações, consulte Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR.

Quando você usa o console para criar ou editar seu pipeline, CodePipeline cria uma EventBridge regra que inicia seu pipeline quando ocorre uma alteração no repositório.

> **Note**
>
> Para Amazon ECR, Amazon S3 ou CodeCommit fontes, você também pode criar uma substituição de origem usando a entrada de transformação de entrada para usar o

`revisionValue` in EventBridge para seu evento de pipeline, em que `revisionValue` é derivado da variável de evento de origem para sua chave de objeto, confirmação ou ID de imagem. Para saber mais, consulte a etapa opcional da entrada da transformação de entrada incluída nos procedimentos em Ações e recursos de origem do Amazon ECR EventBridge, Conexão a ações de origem do Amazon S3 com uma fonte habilitada para eventos ou CodeCommit ações de origem e EventBridge.

Você já deve ter criado um repositório do Amazon ECR e enviado uma imagem por push para que possa conectar o pipeline por meio de uma ação do Amazon ECR.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Permissões do perfil de serviço: ação do Amazon ECR
- Declaração de ação (exemplo do Amazon ECR)
- Consulte também

Tipo de ação

- Categoria: `Source`
- Proprietário: `AWS`
- Fornecedor: `ECR`
- Versão: 1

Parâmetros de configuração

RepositoryName

Obrigatório: Sim

O nome do repositório do Amazon ECR ao qual a imagem foi enviada por push.

ImageTag

Obrigatório: não

A tag usada para a imagem.

> **Note**
>
> Se não for especificado um valor para `ImageTag`, o valor assumirá `latest` como padrão.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `0`
- Descrição: os artefatos de entrada não se aplicam a esse tipo de ação.

Artefatos de saída

- Número de artefatos: `1`
- Descrição: esta ação produz um artefato que contém um arquivo `imageDetail.json` que contém o URI da imagem que acionou a execução do pipeline. Para mais informações sobre o arquivo `imageDetail.json`, consulte Detail.json arquivo de imagem para ações de blue/green implantação do Amazon ECS.

Variáveis de saída

Quando configurada, essa ação produz variáveis que podem ser referenciadas pela configuração de ação de uma ação downstream no pipeline. Esta ação produz variáveis que podem ser visualizadas como variáveis de saída, mesmo que a ação não tenha um namespace. Configure uma ação com um namespace a fim de disponibilizar as variáveis para a configuração de ações downstream.

Para obter mais informações, consulte Referência de variáveis.

RegistryId

O ID da AWS conta associado ao registro que contém o repositório.

RepositoryName

O nome do repositório do Amazon ECR ao qual a imagem foi enviada por push.

ImageTag

A tag usada para a imagem.

> **Note**
>
> A variável `ImageTag` de saída não é gerada quando a revisão da fonte é substituída.

ImageDigest

O resumo `sha256` do manifesto da imagem.

ImageURI

O URI da imagem.

Permissões do perfil de serviço: ação do Amazon ECR

Para compatibilidade com o Amazon ECR, adicione o seguinte à declaração da política:

```
{
    "Effect": "Allow",
    "Action": [
        "ecr:DescribeImages"
    ],
    "Resource": "resource_ARN"
},
```

Para obter mais informações sobre a ação, consulte Referência da ação de origem do Amazon ECR.

Declaração de ação (exemplo do 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"
        }
    ]
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR— Este tutorial fornece um AppSpec arquivo de amostra e um exemplo de CodeDeploy aplicativo e grupo de implantação para criar um pipeline com uma CodeCommit fonte do Amazon ECR que é implantada em instâncias do Amazon ECS.

Referência da ação de compilação **ECRBuildAndPublish**

Esta ação de compilação permite automatizar a compilação e o envio de uma nova imagem quando ocorre uma alteração na origem. Essa ação é criada com base em um local específico do Dockerfile e envia a imagem. Essa ação de criação não é igual à ação de origem do Amazon ECR em CodePipeline, que aciona o pipeline quando ocorre uma alteração no seu repositório de origem do Amazon ECR. Para obter informações sobre essa ação, consulte Referência da ação de origem do Amazon ECR.

Esta não é uma ação de origem que vai disparar o pipeline. Esta ação compila uma imagem e a envia para o repositório de imagens do Amazon ECR.

Você já deve ter criado um repositório Amazon ECR e ter adicionado um Dockerfile ao seu repositório de código-fonte, por exemplo GitHub, antes de adicionar a ação ao seu pipeline.

> **Important**
>
> Essa ação usa CodeBuild computação CodePipeline gerenciada para executar comandos em um ambiente de compilação. A execução da ação Comandos incorrerá em cobranças separadas no AWS CodeBuild.

> **Note**
>
> Esta só está disponível para pipelines do tipo V2.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Permissões do perfil de serviço: ação ECRBuildAndPublish
- Declaração de ação
- Consulte também

Tipo de ação

- Categoria: `Build`
- Proprietário: `AWS`
- Fornecedor: `ECRBuildAndPublish`
- Versão: `1`

Parâmetros de configuração

ECRRepositoryName

Obrigatório: Sim

O nome do repositório do Amazon ECR para o qual a imagem é enviada por push.

DockerFilePath

Obrigatório: não

A localização do Dockerfile usado para criar a imagem. Opcionalmente, você pode fornecer um local alternativo do Dockerfile se não estiver no nível raiz.

> **Note**
>
> Se um valor para `DockerFilePath` não for especificado, o valor assumirá o nível raiz do repositório de origem como padrão.

ImageTags

Obrigatório: não

As tags usadas para a imagem. Você pode inserir várias tags como uma lista de strings delimitada por vírgulas.

> **Note**
>
> Se não for especificado um valor para `ImageTags`, o valor assumirá `latest` como padrão.

RegistryType

Obrigatório: não

Especifica se o repositório é público ou privado. O valor válido é `private | public`.

> **Note**
>
> Se não for especificado um valor para `RegistryType`, o valor assumirá `private` como padrão.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `1`
- Descrição: o artefato produzido pela ação de origem que contém o Dockerfile necessário para compilar a imagem.

Artefatos de saída

- Número de artefatos: `0`

Variáveis de saída

Quando configurada, essa ação produz variáveis que podem ser referenciadas pela configuração de ação de uma ação downstream no pipeline. Esta ação produz variáveis que podem ser visualizadas

como variáveis de saída, mesmo que a ação não tenha um namespace. Configure uma ação com um namespace a fim de disponibilizar as variáveis para a configuração de ações downstream.

Para obter mais informações, consulte Referência de variáveis.

ECRImageDigestId

O resumo `sha256` do manifesto da imagem.

ECRRepositoryName

O nome do repositório do Amazon ECR ao qual a imagem foi enviada por push.

Permissões do perfil de serviço: ação **`ECRBuildAndPublish`**

Para suporte à ação `ECRBuildAndPublish`, adicione o seguinte à declaração da 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": "*"
        }
    ]
}
```

Além disso, se ainda não tiver sido adicionado à `Commands` ação, adicione as seguintes permissões à sua função de serviço para visualizar CloudWatch os registros.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Restrinja as permissões ao nível de recurso do pipeline aplicando permissões baseadas em recurso na declaração de política do perfil de serviço.

Para obter mais informações sobre a ação, consulte Referência da ação de compilação `ECRBuildAndPublish`.

Declaração de ação

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"
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Tutorial: crie e envie uma imagem do Docker para o Amazon ECR com CodePipeline (tipo V2): este tutorial apresenta um Dockerfile de amostra e instruções para criar um pipeline que envia por push a imagem para o ECR em uma alteração feita no repositório de origem e, em seguida, implanta-a no Amazon ECS.

Amazon Elastic Container Service e referência de ação de implantação CodeDeploy azul-esverdeada

Você pode configurar um pipeline AWS CodePipeline que implanta aplicativos de contêiner usando uma blue/green implantação. Em uma blue/green implantação, você pode iniciar uma nova versão do seu aplicativo junto com a versão antiga e testar a nova versão antes de redirecionar o tráfego para ela. Você também poderá monitorar o processo de implantação e realizar uma reversão rapidamente se houver algum problema.

O pipeline concluído detecta alterações em suas imagens ou arquivo de definição de tarefas e usa CodeDeploy para rotear e implantar tráfego em um cluster e balanceador de carga do Amazon ECS. CodeDeploy cria um novo ouvinte em seu balanceador de carga que pode direcionar sua nova tarefa por meio de uma porta especial. Você também pode configurar o pipeline para usar um local de origem, como um CodeCommit repositório, onde sua definição de tarefa do Amazon ECS está armazenada.

Antes de criar seu pipeline, você já deve ter criado os recursos do Amazon ECS, os recursos, o CodeDeploy balanceador de carga e o grupo-alvo. Você já deve ter marcado e armazenado a imagem em seu repositório de imagens e carregado a definição da tarefa e o AppSpec arquivo em seu repositório de arquivos.

> **Note**
>
> Este tópico descreve a ação de CodeDeploy blue/green implantação do Amazon ECS para CodePipeline. Para obter informações de referência sobre as ações de implantação padrão

do Amazon ECS em CodePipeline, consulteReferência da ação de implantação do Amazon Elastic Container Service.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Permissões do perfil de serviço: ação CodeDeployToECS
- Declaração de ação
- Consulte também

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `CodeDeployToECS`
- Versão: `1`

Parâmetros de configuração

ApplicationName

Obrigatório: Sim

O nome do aplicativo em CodeDeploy. Antes de criar seu funil, você já deve ter criado o aplicativo em CodeDeploy.

DeploymentGroupName

Obrigatório: Sim

O grupo de implantação especificado para os conjuntos de tarefas do Amazon ECS que você criou para seu CodeDeploy aplicativo. Antes de criar seu funil, você já deve ter criado o grupo de implantação em CodeDeploy.

TaskDefinitionTemplateArtifact

Obrigatório: Sim

O nome do artefato de entrada que fornece o arquivo de definição de tarefa para a ação de implantação. Geralmente, esse é o nome do artefato de saída da ação de origem. Quando você usa o console, o nome padrão do artefato de saída da ação de origem é `SourceArtifact`.

AppSpecTemplateArtifact

Obrigatório: Sim

O nome do artefato de entrada que fornece o AppSpec arquivo para a ação de implantação. Esse valor é atualizado quando o pipeline é executado. Geralmente, esse é o nome do artefato de saída da ação de origem. Quando você usa o console, o nome padrão do artefato de saída da ação de origem é `SourceArtifact`. Para `TaskDefinition` no AppSpec arquivo, você pode manter o texto do `<TASK_DEFINITION>` espaço reservado conforme mostrado aqui.

AppSpecTemplatePath

Obrigatório: não

O nome do AppSpec arquivo armazenado no local do arquivo de origem do pipeline, como o CodeCommit repositório do pipeline. O nome de arquivo padrão é `appspec.yaml`. Se o AppSpec arquivo tiver o mesmo nome e estiver armazenado no nível raiz do repositório de arquivos, você não precisará fornecer o nome do arquivo. Se o caminho não for o padrão, insira o caminho e o nome do arquivo.

TaskDefinitionTemplatePath

Obrigatório: não

O nome do arquivo da definição da tarefa armazenada no local de origem do arquivo do pipeline, como o CodeCommit repositório do pipeline. O nome de arquivo padrão é `taskdef.json`. Se o arquivo de definição tiver o mesmo nome e estiver armazenado no nível raiz do repositório de arquivos, você não precisará fornecer o nome do arquivo. Se o caminho não for o padrão, insira o caminho e o nome do arquivo.

Imagem <Number>ArtifactName

Obrigatório: não

O nome do artefato de entrada que fornece a imagem para a ação de implantação. Geralmente, esse é o artefato de saída do repositório de imagens, como a saída da ação de origem do Amazon ECR.

Os valores disponíveis para `<Number>` vão de 1 a 4.

Imagem <Number>ContainerName

Obrigatório: não

O nome da imagem disponível no repositório de imagens; por exemplo, o repositório de origem do Amazon ECR.

Os valores disponíveis para `<Number>` vão de 1 a 4.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `1 to 5`
- Descrição: a `CodeDeployToECS` ação primeiro procura o arquivo de definição da tarefa e o AppSpec arquivo no repositório do arquivo de origem, depois procura a imagem no repositório de imagens, depois gera dinamicamente uma nova revisão da definição da tarefa e, finalmente, executa AppSpec os comandos para implantar o conjunto de tarefas e o contêiner no cluster.

  A ação `CodeDeployToECS` procura um arquivo `imageDetail.json` que mapeie o URI de imagem para a imagem. Quando você confirmar uma alteração no repositório de imagens do Amazon ECR, a ação de origem do ECR do pipeline criará um arquivo `imageDetail.json` para essa confirmação. Você também pode adicionar manualmente um arquivo `imageDetail.json` para um pipeline em que a ação não seja automatizada. Para mais informações sobre o arquivo `imageDetail.json`, consulte Detail.json arquivo de imagem para ações de blue/green implantação do Amazon ECS.

  A ação `CodeDeployToECS` gera dinamicamente uma nova revisão da definição de tarefa. Nessa fase, essa ação substitui os espaços reservados no arquivo de definição de tarefas pelo URI de imagem recuperado dos arquivos de imagem. Detail.json Por exemplo, se você definir IMAGE1_NAME como Image1ContainerName parâmetro, deverá especificar o espaço reservado <IMAGE1_NAME>como o valor do campo de imagem em seu arquivo de definição de tarefa. Nesse caso, a CodeDeployToECS ação substitui o espaço reservado pelo <IMAGE1_NAME>URI de imagem real recuperado da imagem Detail.json no artefato que você especifica como. Image1ArtifactName

Para atualizações de definição de tarefas, o CodeDeploy `AppSpec.yaml` arquivo contém a `TaskDefinition` propriedade.

```
TaskDefinition: <TASK_DEFINITION>
```

Essa propriedade será atualizada pela ação `CodeDeployToECS` depois que a nova definição de tarefa for criada.

Para o valor do campo `TaskDefinition`, o texto do espaço reservado deve ser <TASK_DEFINITION>. A ação `CodeDeployToECS` substitui esse espaço reservado pelo ARN real da definição de tarefa gerada dinamicamente.

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Permissões do perfil de serviço: ação **`CodeDeployToECS`**

Para a `CodeDeployToECS` ação (blue/green implantações), a seguir estão as permissões mínimas necessárias para criar pipelines com uma CodeDeploy ação de implantação no Amazon ECS. blue/green

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"
                    ]
                }
            }
        }
    ]
}
```

Você pode aceitar o uso da autorização para atribuição de tags no Amazon ECS. Ao aceitar, você deve conceder as seguintes permissões: `ecs:TagResource`. Para obter mais informações sobre como aceitar e determinar se a permissão será necessária e a autorização de tag será aplicada, consulte o Cronograma de autorização para atribuição de tags no Guia do desenvolvedor do Amazon Elastic Container Service.

Você também deve adicionar as permissões `iam:PassRole` para usar os perfis do IAM para tarefas. Para obter mais informações, consulte Perfil do IAM para execução de tarefas do Amazon ECS e Perfis do IAM para tarefas.

Você também pode adicionar `ecs-tasks.amazonaws.com` à lista de serviços na condição `iam:PassedToService`, conforme mostrado no exemplo acima.

Declaração de ação

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"
        }
    ]
}
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR— Este tutorial orienta você na criação dos recursos CodeDeploy e do Amazon ECS necessários para uma blue/green implantação. O tutorial mostra como enviar por push uma imagem do Docker para o Amazon ECR e criar uma definição de tarefa do Amazon ECS que lista o nome da imagem do Docker, o nome do contêiner, o nome do serviço do Amazon ECS e a configuração do balanceador de carga. Em seguida, o tutorial orienta você na criação do AppSpec arquivo e do pipeline para sua implantação.

  > **Note**
  >
  > Este tópico e este tutorial descrevem a CodeDeploy/ECS blue/green ação do CodePipeline. Para obter informações sobre as ações padrão do ECS em CodePipeline, consulte Tutorial: implantação contínua com CodePipeline.

- AWS CodeDeploy Guia do usuário — Para obter informações sobre como usar o balanceador de carga, o ouvinte de produção, os grupos-alvo e seu aplicativo Amazon ECS em uma blue/green implantação, consulte o Tutorial: Implantar um serviço do Amazon ECS. Essas informações de referência no Guia do AWS CodeDeploy usuário fornecem uma visão geral das blue/green implantações com o Amazon ECS e. AWS CodeDeploy
- Guia do desenvolvedor do Amazon Elastic Container Service: para obter informações sobre como trabalhar com imagens e contêineres do Docker, serviços e clusters do ECS e conjuntos de tarefas do ECS, consulte O que é o Amazon ECS?

Referência da ação de implantação do Amazon Elastic Container Service

Você pode usar uma ação do Amazon ECS para implantar um serviço e um conjunto de tarefas do Amazon ECS. Um serviço do Amazon ECS é uma aplicação de contêiner implantada em um cluster do Amazon ECS. Um cluster do Amazon ECS é um conjunto de instâncias que hospedam sua aplicação de contêiner na nuvem. A implantação exige uma definição de tarefa que você cria no Amazon ECS e um arquivo de definições de imagem CodePipeline usado para implantar a imagem.

Important

A ação de implantação padrão do Amazon ECS para CodePipeline cria sua própria revisão da definição da tarefa com base na revisão usada pelo serviço Amazon ECS. Se você criar novas revisões para a definição de tarefa sem atualizar o serviço Amazon ECS, a ação de implantação ignorará essas revisões.

Antes de criar seu pipeline, você já deve ter criado os recursos do Amazon ECS, marcado e armazenado a imagem em seu repositório de imagens e carregado o arquivo buildspec em seu repositório de arquivos.

Note

Este tópico de referência descreve a ação de implantação padrão do Amazon ECS para CodePipeline. Para obter informações de referência sobre o Amazon ECS para ações CodeDeploy blue/green de implantação em CodePipeline, consulte Amazon Elastic Container Service e referência de ação de implantação CodeDeploy azul-esverdeada.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Permissões do perfil de serviço: ação padrão do Amazon ECS
- Declaração de ação

- Consulte também

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `ECS`
- Versão: `1`

Parâmetros de configuração

ClusterName

Obrigatório: Sim

O cluster do Amazon ECS no Amazon ECS.

ServiceName

Obrigatório: Sim

O serviço Amazon ECS que você criou no Amazon ECS.

FileName

Obrigatório: não

O nome do arquivo de definições de imagem, o arquivo JSON que descreve o nome de contêiner, a imagem e a tag do serviço. Você usa esse arquivo para implantações padrão do ECS. Para obter mais informações, consulte Input artifacts (Artefatos de entrada) e Arquivo imagedefinitions.json para ações de implantação padrão do Amazon ECS.

DeploymentTimeout

Obrigatório: não

O tempo limite da ação de implantação do Amazon ECS em minutos. O tempo limite é configurável até o tempo limite padrão máximo para essa ação. Por exemplo:

```
"DeploymentTimeout": "15"
```

Input artifacts (Artefatos de entrada)

- Número de artefatos: 1
- Descrição: a ação procura um arquivo `imagedefinitions.json` no repositório de arquivos de origem do pipeline. Um documento de definições de imagem é um arquivo JSON que descreve o nome do contêiner do Amazon ECS, a imagem e a tag. CodePipeline usa o arquivo para recuperar a imagem do seu repositório de imagens, como o Amazon ECR. Você também pode adicionar manualmente um arquivo `imagedefinitions.json` para um pipeline em que a ação não seja automatizada. Para mais informações sobre o arquivo `imagedefinitions.json`, consulte Arquivo imagedefinitions.json para ações de implantação padrão do Amazon ECS.

  A ação requer uma imagem existente que já tenha sido enviada por push para o repositório de imagens. Como o mapeamento da imagem é fornecido pelo arquivo `imagedefinitions.json`, a ação não exige que a fonte do Amazon ECR seja incluída como uma ação de origem no pipeline.

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Permissões do perfil de serviço: ação padrão do Amazon ECS

Para o Amazon ECR, estas são as permissões mínimas necessárias para criar pipelines com uma ação de implantação do 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"
                ]
            }
        }
    }
  ]
}
```

Você pode aceitar o uso da autorização para atribuição de tags no Amazon ECS. Ao aceitar, você deve conceder as seguintes permissões: `ecs:TagResource`. Para obter mais informações sobre como aceitar e determinar se a permissão será necessária e a autorização de tag será aplicada, consulte o Cronograma de autorização para atribuição de tags no Guia do desenvolvedor do Amazon Elastic Container Service.

Você deve adicionar as permissões `iam:PassRole` para usar os perfis do IAM para tarefas. Para obter mais informações, consulte Perfil do IAM para execução de tarefas do Amazon ECS e Perfis do IAM para tarefas. Use o seguinte texto de política:

Declaração de ação

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"
        }
    ]
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Veja Tutorial: crie e envie uma imagem do Docker para o Amazon ECR com CodePipeline (tipo V2) um tutorial que mostra como usar a ECRBuildandPublish ação para enviar uma imagem e, em seguida, usar a ação padrão do ECS para implantá-la no Amazon ECS.
- Tutorial: Implantação contínua com CodePipeline — Este tutorial mostra como criar um Dockerfile que você armazena em um repositório de arquivos de origem, como. CodeCommit A seguir, o tutorial mostra como incorporar um arquivo CodeBuild buildspec que cria e envia sua imagem do Docker para o Amazon ECR e cria seu arquivo imagedefinitions.json. Por fim, você cria uma definição de serviço e tarefa do Amazon ECS e, em seguida, cria seu pipeline com uma ação de implantação do Amazon ECS.

> **Note**
>
> Este tópico e tutorial descrevem a ação de implantação padrão do Amazon ECS para CodePipeline. Para obter informações sobre o Amazon ECS para ações CodeDeploy blue/

green de implantação em CodePipeline, consulteTutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR.

- Guia do desenvolvedor do Amazon Elastic Container Service: para obter informações sobre como trabalhar com imagens e contêineres do Docker, serviços e clusters do Amazon ECS e conjuntos de tarefas do ECS, consulte O que é o Amazon ECS?

Referência de ação de implantação do Amazon Elastic Kubernetes Service **EKS**

Você pode usar a ação `EKSDeploy` para implantar um serviço do Amazon EKS. A implantação exige um arquivo de manifesto do Kubernetes CodePipeline usado para implantar a imagem.

Antes de criar o pipeline, você também já deve ter criado os recursos do Amazon EKS e armazenado a imagem no repositório de imagens. Você também pode fornecer as informações do VPC para o cluster.

Important

Essa ação usa CodeBuild computação CodePipeline gerenciada para executar comandos em um ambiente de compilação. A execução da ação Comandos incorrerá em cobranças separadas no AWS CodeBuild.

Note

A ação de implantação `EKS` está disponível somente para pipelines do tipo V2.

A ação do EKS oferece suporte a clusters do EKS públicos e privados. Os clusters privados são do tipo recomendado pelo EKS. No entanto, ambos os tipos são compatíveis.

A ação do EKS é compatível com ações entre contas. Para adicionar uma ação do EKS entre contas, adicione `actionRoleArn` da conta de destino na declaração de ação.

Tópicos

- Tipo de ação

- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de ambiente
- Variáveis de saída
- Permissões de política de perfil de serviço
- Declaração de ação
- Consulte também

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `EKS`
- Versão: `1`

Parâmetros de configuração

ClusterName

Obrigatório: Sim

O cluster do Amazon EKS no Amazon EKS.

Opções no Helm

Estas serão as opções disponíveis quando o Helm for a ferramenta de implantação selecionada.

HelmReleaseName

Obrigatório: sim (obrigatório apenas para o tipo Helm)

O nome da versão da implantação.

HelmChartLocation

Obrigatório: sim (obrigatório apenas para o tipo Helm)

A localização do gráfico da implantação.

HelmValuesFiles

Obrigatório: não (opcional apenas para o tipo Helm)

Para substituir os arquivos de valores do Helm, insira os arquivos de valores do Helm separados por vírgula no local do gráfico do Helm.

Opções em Kubectl

Estas serão as opções disponíveis quando o Kubectl for a ferramenta de implantação selecionada.

ManifestFiles

Obrigatório: sim (obrigatório apenas para o tipo Kubectl)

O nome do arquivo de manifesto, o arquivo de texto que descreve o nome de contêiner, a imagem e a tag do serviço. Você usa esse arquivo para parametrizar o URI da imagem e outras informações. Você pode usar uma variável de ambiente para essa finalidade.

Você armazena esse arquivo no repositório de origem do pipeline.

Namespace

Obrigatório: não

O namespace Kubernetes a ser usado em nossos comandos. `kubectl` `helm`

Sub-redes

Obrigatório: não

As sub-redes da VPC do cluster. Elas fazem parte da mesma VPC conectada ao cluster. Você também pode fornecer sub-redes ainda não conectadas ao cluster e especificá-las aqui.

SecurityGroupIds

Obrigatório: não

Os grupos de segurança da VPC do cluster. Elas fazem parte da mesma VPC conectada ao cluster. Você também pode fornecer grupos de segurança ainda não conectados ao cluster e especificá-los aqui.

Input artifacts (Artefatos de entrada)

- Número de artefatos: 1

- Descrição: a ação procura o arquivo de manifesto do Kubernetes ou o chart do Helm no repositório de arquivos de origem do pipeline. Se quiser usar gráficos do Helm no formato.tgz armazenados em um bucket do S3, você pode fazer isso configurando o Bucket/Key S3 como sua ação de origem. Por exemplo, a chave do objeto fornecida seria `my-chart-0.1.0.tgz`.

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Variáveis de ambiente

Usado para substituir variáveis como repositórios de imagens ou tags de imagem em arquivos de manifesto ou arquivos de valores do gráfico Helm.

Chave

A chave em um par de variáveis do ambiente chave-valor, como `$IMAGE_TAG`.

Valor

O valor do par chave-valor, como `v1.0`. O valor pode ser parametrizado com variáveis de saída a partir de ações ou variáveis do pipeline. Por exemplo, o pipeline pode ter uma ECRBuildAndPublish ação que cria uma imagem ECR com`${codepipeline.PipelineExecutionId}`, e a ação EKS pode usar essa imagem usando `${codepipeline.PipelineExecutionId}` como valor da variável de ambiente.

Variáveis de saída

EKSClusterName

O cluster do Amazon EKS no Amazon EKS.

Permissões de política de perfil de serviço

Para executar essa ação, as permissões a seguir devem estar disponíveis na política do perfil de serviço do pipeline.

- Ações do EC2: Quando a ação CodePipeline é executada, as permissões da instância do EC2 são necessárias. Não se trata da mesma função da instância do EC2 necessária quando você cria o cluster do EKS.

  Se estiver usando um perfil de serviço existente, para usar essa ação, você precisará adicionar as permissões a seguir para o perfil de serviço.

  - ec2: CreateNetworkInterface
  - ec2: DescribeDhcpOptions
  - ec2: DescribeNetworkInterfaces
  - ec2: DeleteNetworkInterface
  - ec2: DescribeSubnets
  - ec2: DescribeSecurityGroups
  - ec2: DescribeVpcs
- Ações do EKS: quando a ação CodePipeline é executada, as permissões do cluster EKS são necessárias. Não se trata da mesma função do cluster do EKS do IAM necessária quando você cria o cluster do EKS.

  Se estiver usando um perfil de serviço existente, para usar essa ação, você precisará adicionar a permissão a seguir para o perfil de serviço.

  - ex: DescribeCluster
- Ações de fluxo de log: quando CodePipeline executa a ação, CodePipeline cria um grupo de logs usando o nome do pipeline da seguinte forma. Isso permite reduzir as permissões para os recursos de log usando o nome do pipeline.

  ```
  /aws/codepipeline/MyPipelineName
  ```

  Se estiver usando um perfil de serviço existente, para usar essa ação, você precisará adicionar as permissões a seguir para o perfil de serviço.

  - registros: CreateLogGroup
  - registros: CreateLogStream
  - registros: PutLogEvents

Na declaração da política do perfil de serviço, reduza o escopo das permissões até o nível de recurso conforme mostrado no exemplo a seguir.

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 visualizar logs no console usando a página de diálogo de detalhes da ação, a permissão para visualizar logs deve ser adicionada ao perfil do console. Para ter mais informações, consulte o exemplo de política de permissões do console em Permissões necessárias para visualizar registros computacionais no console.

Adição do perfil de serviço como uma entrada de acesso para o cluster

Depois que as permissões estiverem disponíveis na política de função de serviço do seu pipeline, você configura suas permissões de cluster adicionando a função de CodePipeline serviço como uma entrada de acesso para seu cluster.

Você também pode usar uma função de ação que tenha as permissões atualizadas. Para obter mais informações, consulte o exemplo do tutorial em Etapa 4: criar uma entrada de acesso para a função CodePipeline de serviço.

Declaração de ação

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"
        }
    ]
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Consulte Tutorial: Implante no Amazon EKS com CodePipeline acessar para um tutorial que demonstre como criar um cluster do EKS e um arquivo de manifesto do Kubernetes para adicionar a ação ao pipeline.

AWS Lambda Referência da ação de implantação do

Você usa uma ação de AWS Lambda implantação para gerenciar a implantação do código do aplicativo para sua implantação sem servidor. Você pode implantar uma função e usar estratégias de implantação para implantação de tráfego assim:

- Implantações canário e linear para mudança de tráfego
- Implantações todas de uma vez

> **Note**
>
> Essa ação é compatível somente para pipelines tipo V2.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Permissões de política do perfil de serviço para a ação de implantação do Lambda
- Declaração de ação
- Consulte também

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `Lambda`
- Versão: `1`

Parâmetros de configuração

FunctionName

Obrigatório: Sim

O nome da função criado por você no Lambda, como `MyLambdaFunction`.

Você já deve ter criado uma versão.

FunctionAlias

Obrigatório: não

O alias da função que você criou no Lambda e é a função a ser implantada, como `live`. O alias deve existir e ter uma versão por trás dele quando a execução da ação começar. (Será a versão de destino da reversão.)

Se não for fornecida, a ação vai implantar o artefato de origem em `$LATEST` e criar uma nova versão. Nesse caso de uso, as opções da estratégia de implantação e da versão de destino não estão disponíveis.

PublishedTargetVersion

Obrigatório: não

A versão desejada da função Lambda a ser implantada. FunctionAlias Podem ser variáveis do pipeline ou do nível de ação, como `#{variables.lambdaTargetVersion}`. A versão deverá ser publicada quando a execução da ação for iniciada.

Obrigatório se nenhum artefato de entrada for fornecido.

DeployStrategy

Obrigatório: não (o padrão é `AllAtOnce`)

Determina a taxa que a ação de implantação do Lambda adota para transferir o tráfego da versão original da função do Lambda para a nova versão do. FunctionAlias As estratégias de implantação disponíveis são canário ou linear. Formatos aceitos:

- `AllAtOnce` -

  Desloca todo o tráfego para as funções do Lambda atualizadas de uma única vez.

  Se não especificado, o padrão será `AllAtOnce`.
- `Canary10Percent5Minutes`: muda 10 por cento do tráfego no primeiro incremento. Os 90 por cento restantes serão implantados cinco minutos depois.

  Os valores de porcentagem e minutos podem ser alterados.
- `Linear10PercentEvery1Minute`: muda 10 por cento do tráfego a cada minuto até que todo o tráfego seja deslocado.

  Os valores de porcentagem e minutos podem ser alterados.

As seguintes considerações se aplicam para este campo:

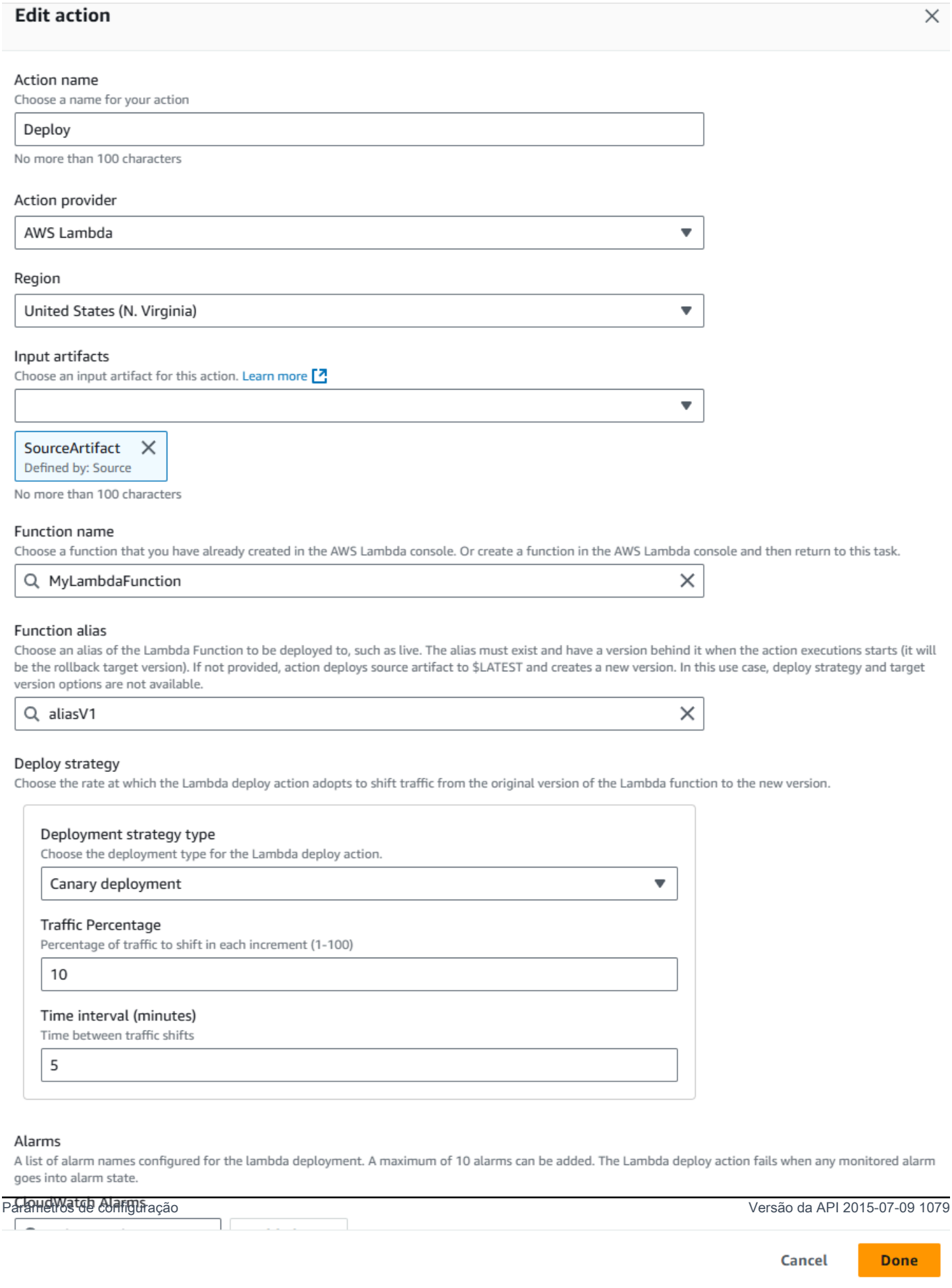
- O tempo de espera total máximo é de 2 dias.
- Disponível somente quando FunctionAlias fornecido.

Alarmes

Obrigatório: não

Uma lista separada por vírgulas dos nomes de alarme configurados para a implantação do Lambda. No máximo, dez alarmes podem ser adicionados. A ação falha quando alarmes monitorados vão para o estado ALARME.

A imagem a seguir mostra um exemplo da página Editar da ação.


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

Input artifacts (Artefatos de entrada)

- Número de artefatos: 1
- Descrição: os arquivos fornecidos, se houver, para oferecer suporte às ações de script durante a implantação.

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Variáveis de saída

Quando configurada, essa ação produz variáveis que podem ser referenciadas pela configuração de ação de uma ação downstream no pipeline. Esta ação produz variáveis que podem ser visualizadas como variáveis de saída, mesmo que a ação não tenha um namespace. Configure uma ação com um namespace a fim de disponibilizar as variáveis para a configuração de ações downstream.

Para obter mais informações, consulte Referência de variáveis.

FunctionVersion

A nova versão da função do Lambda que foi implantada.

Permissões de política do perfil de serviço para a ação de implantação do Lambda

Quando CodePipeline executa a ação, a função CodePipeline de serviço exige as seguintes permissões, com escopo adequado para acesso com o mínimo de privilégios.

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:*"
            ]
        }
    ]
}
```

Declaração de ação

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"
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Tutorial: implantações da função Lambda com CodePipeline: este tutorial apresenta a criação de uma função do Lambda de amostra em que você criará um alias e uma versão, adicionar a função do Lambda compactada ao local de origem e executar a ação do Lambda no pipeline.

Referência da ação de implantação do Amazon S3

Você usa uma ação de implantação do Amazon S3 para implantar arquivos em um bucket do Amazon S3 para hospedagem ou arquivamento estático de sites. Você pode especificar se deseja extrair os arquivos de implantação antes de fazer upload para seu bucket.

> **Note**
>
> Este tópico de referência descreve a ação de implantação do Amazon S3 em CodePipeline que a plataforma de implantação é um bucket do Amazon S3 configurado para hospedagem. Para obter informações de referência sobre a ação de origem do Amazon S3 em CodePipeline, consulteReferência da ação de origem do Amazon S3.

Tópicos

- Tipo de ação

- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Permissões do perfil de serviço: ação de implantação do S3
- Exemplo de configuração da ação
- Consulte também

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `S3`
- Versão: `1`

Parâmetros de configuração

BucketName

Obrigatório: Sim

O nome do bucket do Amazon S3 em que os arquivos serão implantados.

Extract

Obrigatório: Sim

Se for true, especifica que os arquivos serão extraídos antes do upload. Caso contrário, os arquivos do aplicativo permanecerão compactados para upload, como no caso de um site estático hospedado. Se for false, `ObjectKey` será obrigatório.

ObjectKey

Condicional. Obrigatório se `Extract` = falso

O nome da chave de objeto do Amazon S3 que identifica exclusivamente o objeto no bucket do S3.

KMSEncryptionKeyARN

Obrigatório: não

O ARN da chave de AWS KMS criptografia para o bucket do host. O parâmetro `KMSEncryptionKeyARN` criptografa os artefatos carregados com a AWS KMS key fornecida. Para uma chave do KMS, você pode usar o ID da chave, o ARN da chave ou o ARN do alias.

> **Note**
>
> Os aliases são reconhecidos apenas na conta que criou a chave do KMS. Para ações entre contas, você só pode usar o ID da chave ou o ARN da chave para identificar a chave. Cross-account as ações envolvem o uso da função da outra conta (contaB), portanto, especificar o ID da chave usará a chave da outra conta (contaB).

> **Important**
>
> CodePipeline suporta apenas chaves KMS simétricas. Não use uma chave assimétrica do KMS para criptografar os dados no bucket do S3.

CannedACL

Obrigatório: não

O parâmetro `CannedACL` aplica a ACL pré-configurada especificada aos objetos implantados no Amazon S3. Isso substitui todas as ACLs existentes que foram aplicadas ao objeto.

CacheControl

Obrigatório: não

O `CacheControl` parâmetro controla o comportamento de armazenamento em cache requests/responses de objetos no bucket. Para obter uma lista de valores válidos, consulte o campo de cabeçalho `Cache-Control` para operações HTTP. Para inserir vários valores em `CacheControl`, use uma vírgula entre cada valor. É possível adicionar um espaço após cada vírgula (opcional), conforme mostrado neste exemplo para a CLI:

```
"CacheControl": "public, max-age=0, no-transform"
```

Input artifacts (Artefatos de entrada)

- Número de artefatos: 1
- Descrição: Os arquivos para implantação ou arquivamento são obtidos do repositório de origem, compactados e enviados por. CodePipeline

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Permissões do perfil de serviço: ação de implantação do S3

Para suporte à ação de implantação do S3, adicione o seguinte à declaração da 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 respaldar a ação de implantação do S3, se os objetos do S3 tiverem tags, você também deverá adicionar as seguintes permissões à declaração da política:

```
"s3:GetObjectTagging",
"s3:GetObjectVersionTagging",
"s3:PutObjectTagging"
```

Exemplo de configuração da ação

O exemplo a seguir mostra exemplos da configuração da ação.

Exemplo de configuração quando a **`Extração`** é definida **`como falsa`**

O exemplo a seguir mostra a configuração de ação padrão quando a ação é criada com o campo `Extract` definido como `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"
        }
    ]
},
```

Exemplo de configuração quando **`Extract`** é definido como **`true`**

O exemplo a seguir mostra a configuração de ação padrão quando a ação é criada com o campo `Extract` definido como `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"
        }
    ]
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Tutorial: Criar um pipeline que usa o Amazon S3 como um provedor de implantação: este tutorial mostra dois exemplos de criação de um pipeline com uma ação de implantação do S3. Você baixa arquivos de amostra, carrega os arquivos no seu CodeCommit repositório, cria seu bucket do S3 e configura seu bucket para hospedagem. Em seguida, você usa o CodePipeline console para criar seu pipeline e especificar uma configuração de implantação do Amazon S3.
- Referência da ação de origem do Amazon S3— Essa referência de ação fornece informações de referência e exemplos de ações de origem do Amazon S3 em CodePipeline.

Referência da ação de origem do Amazon S3

Uma ação de origem aciona o pipeline quando um novo objeto é carregado no bucket e na chave do objeto configurados.

Este tópico de referência descreve a ação de origem do Amazon S3 em que o CodePipeline local de origem é um bucket do Amazon S3 configurado para controle de versão. Para obter informações de referência sobre a ação de implantação do Amazon S3 em CodePipeline, consulteReferência da ação de implantação do Amazon S3.

Antes de conectar o pipeline, crie um bucket de origem do Amazon S3, habilite o controle de versão e faça o upload dos arquivos de origem como um único arquivo ZIP.

Você pode criar um novo bucket no Amazon S3 ou selecionar um bucket existente do Amazon S3 para usar como local de origem dos arquivos do seu aplicativo. Para obter mais informações sobre como ativar o controle de versão, consulte Usando o controle de versão no Guia do usuário do Amazon S3.

Quando o Amazon S3 é o provedor de origem do seu pipeline, você pode compactar seu arquivo ou arquivos de origem em um único .zip e fazer o upload do .zip no seu bucket de origem. Você também pode fazer upload de um único arquivo descompactado; no entanto, as ações posteriores que esperam um arquivo.zip falharão.

Se você usar o console para criar ou editar seu pipeline, CodePipeline crie uma EventBridge regra que inicia seu pipeline quando ocorre uma alteração no bucket de origem do Amazon S3.

Você também pode criar uma substituição de origem usando a entrada de transformação de entrada para usar o `revisionValue` in EventBridge para seu evento de pipeline, em que `revisionValue` é derivado da variável de evento de origem para sua chave de objeto, confirmação ou ID de imagem. Para obter mais informações, consulte a etapa opcional para a entrada de transformação de entrada incluída nos procedimentos abaixoConexão a ações de origem do Amazon S3 com uma fonte habilitada para eventos.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Permissões do perfil de serviço: ação de origem do S3
- Declaração de ação
- Consulte também

Tipo de ação

- Categoria: `Source`
- Proprietário: `AWS`
- Fornecedor: `S3`
- Versão: `1`

Parâmetros de configuração

S3 Bucket

Obrigatório: Sim

O nome do bucket do Amazon S3 em que as alterações na origem devem ser detectadas.

S3ObjectKey

Obrigatório: Sim

O nome da chave de objeto do Amazon S3 em que as alterações na origem devem ser detectadas.

AllowOverrideForS3ObjectKey

Obrigatório: não

`AllowOverrideForS3ObjectKey` controla se as substituições de origem de `StartPipelineExecution` podem substituir a `S3ObjectKey` que já está configurada na ação de origem. Para ter mais informações sobre substituições de origem com a chave de objeto do S3, consulte Iniciar um pipeline com uma substituição da revisão de origem.

> **Important**
>
> Se você omitir`AllowOverrideForS3ObjectKey`, CodePipeline definirá como padrão a capacidade de substituir o S3 ObjectKey na ação de origem definindo esse parâmetro como. `false`

Os valores válidos para esse parâmetro:

- `true`: se definido, a chave de objeto do S3 pré-configurada poderá ser substituída por revisões de origem durante a execução de um pipeline.

  > **Note**
  >
  > Se você pretende permitir que todos os CodePipeline usuários substituam a chave de objeto pré-configurada do S3 ao iniciar uma nova execução de pipeline, você deve definir como. `AllowOverrideForS3ObjectKey` `true`

- `false`:

  Se definido, não CodePipeline permitirá que a chave de objeto do S3 seja substituída usando substituições de revisão de origem. Esse também é o valor padrão para este parâmetro.

PollForSourceChanges

Obrigatório: não

`PollForSourceChanges`controla se CodePipeline pesquisa o bucket de origem do Amazon S3 em busca de alterações na fonte. Recomendamos que você use CloudWatch Eventos e CloudTrail, em vez disso, detecte alterações na fonte. Para obter mais informações sobre como

configurar CloudWatch eventos, consulte Migre canais de pesquisa com uma origem e CloudTrail trilha (CLI) do S3 ouMigre canais de votação com uma origem e uma trilha do S3 (CloudTrail CloudFormation modelo).

> **Important**
>
> Se você pretende configurar CloudWatch eventos, `PollForSourceChanges` defina `false` para evitar execuções duplicadas de pipeline.

Os valores válidos para esse parâmetro:

- `true`: Se definido, CodePipeline pesquisa seu local de origem para ver se há alterações na fonte.

  > **Note**
  >
  > Se você omitir`PollForSourceChanges`, o CodePipeline padrão é pesquisar o local de origem em busca de alterações na fonte. Esse comportamento será o mesmo quando o `PollForSourceChanges` estiver incluído e definido como `true`.

- `false`: Se definido, CodePipeline não pesquisa sua localização de origem em busca de alterações na fonte. Use essa configuração se você pretende configurar uma regra de CloudWatch eventos para detectar alterações na fonte.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `0`
- Descrição: os artefatos de entrada não se aplicam a esse tipo de ação.

Artefatos de saída

- Número de artefatos: `1`
- Descrição: fornece os artefatos disponíveis no bucket de origem configurado para conectar-se ao pipeline. Os artefatos gerados no bucket são os artefatos de saída para a ação do Amazon S3. Os metadados do objeto Amazon S3 (ETag e ID da versão) são exibidos CodePipeline como a revisão de origem para a execução do pipeline acionado.

Variáveis de saída

Quando configurada, essa ação produz variáveis que podem ser referenciadas pela configuração de ação de uma ação downstream no pipeline. Esta ação produz variáveis que podem ser visualizadas como variáveis de saída, mesmo que a ação não tenha um namespace. Configure uma ação com um namespace a fim de disponibilizar as variáveis para a configuração de ações downstream.

Para obter mais informações sobre variáveis em CodePipeline, consulteReferência de variáveis.

BucketName

O nome do bucket do Amazon S3 relacionado à alteração de origem que acionou o pipeline.

ETag

A tag de entidade do objeto relacionado à alteração na origem que acionou o pipeline. A ETag é um hash MD5 do objeto. A ETag reflete apenas as alterações no conteúdo de um objeto, não em seus metadados.

ObjectKey

O nome da chave de objeto do Amazon S3 relacionada à alteração de origem que acionou o pipeline.

VersionId

O ID da versão do objeto relacionado à alteração na origem que acionou o pipeline.

Permissões do perfil de serviço: ação de origem do S3

Para suporte à ação de origem do S3, adicione o seguinte à declaração da 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"
        }
      }
    }
  ]
}
```

Declaração de ação

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": []
        }
    ]
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Tutorial: Criar um pipeline simples (bucket do S3)— Este tutorial fornece um AppSpec arquivo de amostra e um exemplo de CodeDeploy aplicativo e grupo de implantação. Use este tutorial para criar um pipeline com uma origem do Amazon S3 implantada em instâncias do Amazon EC2.

Referência da ação de implantação do AWS AppConfig

AWS AppConfig é uma capacidade de AWS Systems Manager. AppConfig oferece suporte a implantações controladas em aplicativos de qualquer tamanho e inclui verificações e monitoramento de validação integrados. Você pode usar AppConfig com aplicativos hospedados em instâncias do Amazon EC2, contêineres AWS Lambda, aplicativos móveis ou dispositivos de IoT.

A `AppConfig` ação de implantação é uma AWS CodePipeline ação que implanta configurações armazenadas no local de origem do pipeline em um AppConfig aplicativo, ambiente e perfil de configuração especificados. Ele usa as preferências definidas em uma estratégia AppConfig de implantação.

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `AppConfig`
- Versão: `1`

Parâmetros de configuração

Aplicação

Obrigatório: Sim

O ID do AWS AppConfig aplicativo com os detalhes de sua configuração e implantação.

Environment

Obrigatório: Sim

O ID do AWS AppConfig ambiente em que a configuração é implantada.

ConfigurationProfile

Obrigatório: Sim

O ID do perfil de AWS AppConfig configuração a ser implantado.

InputArtifactConfigurationPath

Obrigatório: Sim

O caminho do arquivo dos dados de configuração no artefato de entrada a ser implantado.

DeploymentStrategy

Obrigatório: não

A estratégia AWS AppConfig de implantação a ser usada para implantação.

Input artifacts (Artefatos de entrada)

- Número de artefatos: 1
- Descrição: o artefato de entrada da ação de implantação.

Artefatos de saída

Não aplicável.

Permissões do perfil de serviço: ação **AppConfig**

Ao CodePipeline executar a ação, a política de função de CodePipeline serviço exige as seguintes permissões, com escopo adequado até o nível do recurso, a fim de manter o acesso com menos privilégios.

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"
```

```
        }
    ]
}
```

Exemplo de configuração da ação

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"
        }
    ]
}
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- AWS AppConfig— Para obter informações sobre AWS AppConfig implantações, consulte o Guia do AWS Systems Manager usuário.
- Tutorial: Crie um pipeline que use AWS AppConfig como provedor de implantação— Este tutorial ajuda você a começar a configurar arquivos e AppConfig recursos simples de configuração de implantação e mostra como usar o console para criar um pipeline com uma ação de AWS AppConfig implantação.

Referência da ação de implantação do CloudFormation

Executa uma operação em uma CloudFormation pilha. Uma pilha é uma coleção de AWS recursos que você pode gerenciar como uma única unidade. Os recursos em uma pilha são definidos pelo modelo do CloudFormation da pilha. Um conjunto de alterações cria uma comparação que pode ser visualizada sem alterar a pilha original. Para obter informações sobre os tipos de CloudFormation ações que podem ser executadas em pilhas e conjuntos de alterações, consulte o `ActionMode` parâmetro.

Para criar uma mensagem de erro para uma CloudFormation ação em que uma operação de pilha falhou, CodePipeline chame a CloudFormation `DescribeStackEvents` API. Se uma função do IAM de ação tiver permissão para acessar essa API, os detalhes sobre o primeiro recurso com falha serão incluídos na mensagem CodePipeline de erro. Caso contrário, se a política de função não tiver a permissão apropriada, CodePipeline ignorará o acesso à API e, em vez disso, mostrará uma mensagem de erro genérica. Para fazer isso, a permissão `cloudformation:DescribeStackEvents` deve ser adicionada ao perfil de serviço ou a outros perfis do IAM para o pipeline.

Se você não quiser que os detalhes do recurso apareçam nas mensagens de erro do pipeline, poderá revogar essa permissão para o perfil do IAM de ação removendo a permissão `cloudformation:DescribeStackEvents`.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Permissões da função de serviço: CloudFormation ação
- Declaração de ação
- Consulte também

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `CloudFormation`
- Versão: `1`

Parâmetros de configuração

ActionMode

Obrigatório: Sim

`ActionMode`é o nome da ação CloudFormation executada em uma pilha ou conjunto de alterações. Os seguintes modos de ação estão disponíveis:

- `CHANGE_SET_EXECUTE` executa um conjunto de alterações para a pilha de recursos com base em um conjunto de atualizações de recurso especificadas. Com essa ação, CloudFormation começa a alterar a pilha.
- `CHANGE_SET_REPLACE` cria um conjunto de alterações, se ainda não existir, com base no nome da pilha e no modelo enviado. Se o conjunto de alterações existir, ele será CloudFormation excluído e, em seguida, criará um novo.
- `CREATE_UPDATE` cria a pilha, caso não exista. Se a pilha existir, CloudFormation atualiza a pilha. Use esta ação para atualizar pilhas existentes. Ao contrárioREPLACE_ON_FAILURE, se a pilha existir e estiver em um estado de falha, CodePipeline não excluirá nem substituirá a pilha.
- `DELETE_ONLY` exclui uma pilha. Se você especificar uma pilha que não existe, a ação será concluída com êxito sem excluir uma pilha.
- `REPLACE_ON_FAILURE` cria uma pilha, caso não exista. Se a pilha existir e estiver em um estado de falha, CloudFormation excluirá a pilha e criará uma nova pilha. Se a pilha não estiver em um estado de falha, CloudFormation atualize-a.

  A pilha está em no estado de falha quando qualquer um dos seguintes tipos de status estiver exibido no CloudFormation:

  - `ROLLBACK_FAILED`
  - `CREATE_FAILED`
  - `DELETE_FAILED`
  - `UPDATE_ROLLBACK_FAILED`

  Use esta ação para substituir automaticamente as pilhas com falha sem recuperá-las nem solucionar o problema delas.

  > **Important**
  >
  > Recomendamos usar `REPLACE_ON_FAILURE` apenas para fins de teste, pois ele pode excluir sua pilha.

StackName

Obrigatório: Sim

`StackName` é o nome de uma pilha existente ou de uma pilha que você deseja criar.

Capacidades

Obrigatório: condicional

O uso de `Capabilities` reconhece que o modelo pode ter os recursos para criar e atualizar alguns recursos por conta própria e que esses recursos são determinados com base nos tipos de recursos do modelo.

Essa propriedade será necessária se você tiver recursos do IAM em seu modelo de pilha ou criar uma pilha diretamente de um modelo que contém macros. Para que a CloudFormation ação opere com sucesso dessa forma, você deve reconhecer explicitamente que gostaria que ela funcionasse com um dos seguintes recursos:

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`
- `CAPABILITY_AUTO_EXPAND`

Você pode especificar mais de um recurso usando uma vírgula (sem espaço) entre os recursos. O exemplo em Declaração de ação mostra uma entrada com as propriedades CAPABILITY_IAM e CAPABILITY_AUTO_EXPAND.

Para obter mais informações sobre`Capabilities`, consulte as propriedades abaixo UpdateStackna Referência da AWS CloudFormation API.

ChangeSetName

Obrigatório: condicional

`ChangeSetName` é o nome de um conjunto de alterações existente ou um novo conjunto de alterações que você deseja criar para a pilha especificada.

Essa propriedade é necessária para os seguintes modos de ação: CHANGE_SET_REPLACE e CHANGE_SET_EXECUTE. Para todos os outros modos de ação, essa propriedade será ignorado.

RoleArn

Obrigatório: condicional

O `RoleArn` é o ARN da função de serviço do IAM que o CloudFormation assume ao operar em recursos na pilha especificada. `RoleArn` não é aplicado ao executar um conjunto de alterações. Se você não usar CodePipeline para criar o conjunto de alterações, verifique se o conjunto de alterações ou a pilha tem uma função associada.

> **Note**
>
> Esse perfil deve estar na mesma conta do perfil da ação em execução, conforme configurado no `RoleArn` da declaração de ação.

Essa propriedade é necessária para os seguintes modos de ação:

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- DELETE_ONLY
- CHANGE_SET_REPLACE

> **Note**
>
> CloudFormation recebe uma URL assinada em S3 para o modelo; portanto, isso `RoleArn` não precisa de permissão para acessar o repositório de artefatos. No entanto, o `RoleArn` da ação precisa de permissão para acessar o bucket de artefatos, a fim de gerar o URL assinado.

TemplatePath

Obrigatório: condicional

`TemplatePath`representa o arquivo CloudFormation de modelo. Inclua o arquivo em um artefato de entrada para essa ação. O nome do arquivo segue este formato:

`Artifactname``::`*`TemplateFileName`*

`Artifactname` é o nome do artefato de entrada, da forma como aparece no CodePipeline. Por exemplo, um estágio de origem com o nome de artefato de `SourceArtifact` e um nome de arquivo de `template-export.json` cria um nome `TemplatePath`, conforme mostrado neste exemplo:

```
"TemplatePath": "SourceArtifact::template-export.json"
```

Essa propriedade é necessária para os seguintes modos de ação:

- CREATE_UPDATE

- REPLACE_ON_FAILURE
- CHANGE_SET_REPLACE

Para todos os outros modos de ação, essa propriedade será ignorado.

> **Note**
>
> O arquivo CloudFormation de modelo contendo o corpo do modelo tem um comprimento mínimo de 1 byte e um comprimento máximo de 1 MB. Para ações CloudFormation de implantação em CodePipeline, o tamanho máximo do artefato de entrada é sempre 256 MB. Para mais informações, consulte Cotas em AWS CodePipeline e Limites do CloudFormation.

OutputFileName

Obrigatório: não

Use `OutputFileName` para especificar um nome de arquivo de saída, como`CreateStackOutput.json`, que é CodePipeline adicionado ao artefato de saída do pipeline para essa ação. O arquivo JSON contém o conteúdo da `Outputs` seção da CloudFormation pilha.

Se você não especificar um nome, CodePipeline não gera um arquivo ou artefato de saída.

ParameterOverrides

Obrigatório: não

Os parâmetros são definidos no modelo de pilha e permitem que você forneça valores para eles no momento da criação ou atualização da pilha. Você pode usar um objeto JSON para definir valores de parâmetro em seu modelo. (Esses valores substituem os definidos no arquivo de configuração de modelo.) Para obter mais informações sobre como usar substituições de parâmetros, consulte Propriedades de configuração (objeto JSON).

Recomendamos usar o arquivo de configuração de modelo para a maioria dos valores de parâmetro. Use substituições de parâmetros somente para valores que não são conhecidos até que o pipeline esteja em execução. Para obter mais informações, consulte Usando funções de substituição de parâmetros com CodePipeline pipelines no Guia do AWS CloudFormation usuário.

> **Note**
>
> Todos os nomes de parâmetros devem estar presentes no modelo de pilha.

TemplateConfiguration

Obrigatório: não

`TemplateConfiguration` é o arquivo de configuração do modelo. Inclua o arquivo em um artefato de entrada para essa ação. Ele pode conter valores de parâmetro de modelo e uma política de pilha. Para obter mais informações sobre o formato do arquivo de configuração de modelo, consulte Artefatos do AWS CloudFormation.

O nome do arquivo de configuração de modelo segue este formato:

`Artifactname``::`*`TemplateConfigurationFileName`*

`Artifactname` é o nome do artefato de entrada, da forma como aparece no CodePipeline. Por exemplo, um estágio de origem com o nome de artefato de `SourceArtifact` e um nome de arquivo de `test-configuration.json` cria um nome `TemplateConfiguration`, conforme mostrado neste exemplo:

```
"TemplateConfiguration": "SourceArtifact::test-configuration.json"
```

Input artifacts (Artefatos de entrada)

- Número de artefatos: `0 to 10`
- Descrição: Como entrada, a CloudFormation ação aceita opcionalmente artefatos para estas finalidades:
  - Fornecer o arquivo de modelo de pilha a ser executado. (Consulte o parâmetro `TemplatePath`.)
  - Fornecer o arquivo de configuração de modelo a ser usado. (Consulte o parâmetro `TemplateConfiguration`.) Para obter mais informações sobre o formato do arquivo de configuração de modelo, consulte Artefatos do AWS CloudFormation.
  - Fornecer o artefato para que uma função Lambda seja implantada como parte da pilha. CloudFormation

Artefatos de saída

- Número de artefatos: `0 to 1`
- Descrição: se o parâmetro `OutputFileName` for especificado, haverá um artefato de saída produzido por essa ação que contém um arquivo JSON com o nome especificado. O arquivo JSON contém o conteúdo da seção Saídas da pilha do CloudFormation .

  Para obter mais informações sobre a seção de saídas que pode ser criada para a ação do CloudFormation , consulte Saídas.

Variáveis de saída

Quando configurada, essa ação produz variáveis que podem ser referenciadas pela configuração de ação de uma ação downstream no pipeline. Configure uma ação com um namespace a fim de disponibilizar as variáveis para a configuração de ações downstream.

Para CloudFormation ações, as variáveis são produzidas a partir de qualquer valor designado na `Outputs` seção de um modelo de pilha. Observe que os únicos modos de CloudFormation ação que geram saídas são aqueles que resultam na criação ou atualização de uma pilha, como criação de pilha, atualizações de pilha e execução de conjuntos de alterações. Os modos de ação correspondentes que geram variáveis são:

- `CHANGE_SET_EXECUTE`
- `CHANGE_SET_REPLACE`
- `CREATE_UPDATE`
- `REPLACE_ON_FAILURE`

Para obter mais informações, consulte Referência de variáveis. Para ver um tutorial que mostra como criar um pipeline com uma ação de CloudFormation implantação em um pipeline que usa variáveis CloudFormation de saída, consulteTutorial: criar um pipeline que usa variáveis de AWS CloudFormation ações de implantação.

Permissões da função de serviço: CloudFormation ação

Ao CodePipeline executar a ação, a política de função de CodePipeline serviço exige as seguintes permissões, com escopo adequado até o ARN do recurso do pipeline, a fim de manter o acesso com menos privilégios. Por exemplo, adicione o seguinte à declaração da 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"
                    ]
                }
            }
        }
    ]
}
```

Observe que a permissão `cloudformation:DescribeStackEvents` é opcional. Isso permite que a CloudFormation ação mostre uma mensagem de erro mais detalhada. Essa permissão poderá ser revogada do perfil do IAM se você não quiser que os detalhes do recurso apareçam nas mensagens de erro do pipeline.

Declaração de ação

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"
        }
    ]
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Referência de propriedades de configuração — Este capítulo de referência no Guia AWS CloudFormation do usuário fornece mais descrições e exemplos desses CodePipeline parâmetros.
- AWS CloudFormation Referência da API — O CreateStackparâmetro na Referência da AWS CloudFormation API descreve os parâmetros da pilha para CloudFormation modelos.

CloudFormation Referência da ação de implantação do StackSets

CodePipeline oferece a capacidade de realizar CloudFormation StackSets operações como parte de seu CI/CD processo. Você usa um conjunto de pilhas para criar pilhas em AWS contas em todas AWS as regiões usando um único CloudFormation modelo. Todos os recursos incluídos em cada pilha são definidos pelo modelo do conjunto de CloudFormation pilhas. Ao criar o conjunto de pilhas, você especifica o modelo a ser usado, bem como os parâmetros e recursos exigidos pelo modelo.

Para obter mais informações sobre conceitos para CloudFormation StackSets, consulte StackSets conceitos no Guia AWS CloudFormation do usuário.

Você integra seu funil por CloudFormation StackSets meio de dois tipos de ação distintos que você usa em conjunto:

- A ação `CloudFormationStackSet` cria ou atualiza um conjunto de pilhas ou instâncias de pilha a partir do modelo armazenado no local de origem do pipeline. Sempre que um conjunto de pilhas é criado ou atualizado, ele inicia uma implantação dessas alterações em instâncias especificadas. No console, você pode escolher o provedor de ações do CloudFormation Stack Set ao criar ou editar seu funil.
- A ação `CloudFormationStackInstances` implanta alterações da ação `CloudFormationStackSet` para instâncias especificadas, cria novas instâncias de pilha e define substituições de parâmetros para instâncias especificadas. No console, você pode escolher o provedor de ação CloudFormation Stack Instances ao editar um pipeline existente.

Você pode usar essas ações para implantar em AWS contas de destino ou IDs de unidades organizacionais de AWS organizações de destino.

> **Note**
>
> Para implantar contas ou IDs de unidades organizacionais de destino e usar o modelo de permissões gerenciadas pelo serviço, você deve habilitar o acesso confiável entre AWS CloudFormation StackSets as AWS AWS organizações. Para obter mais informações, consulte Como ativar o acesso confiável com AWS CloudFormation Stacksets.

Tópicos

- Como CloudFormation StackSets ações funcionam

- Como estruturar StackSets ações em um pipeline
- A ação CloudFormationStackSet
- A CloudFormationStackInstances ação
- Permissões do perfil de serviço: ação CloudFormationStackSet
- Permissões do perfil de serviço: ação CloudFormationStackInstances
- Modelos de permissões para operações de conjuntos de pilhas
- Tipos de dados de parâmetro do modelo
- Consulte também

Como CloudFormation StackSets ações funcionam

Uma ação `CloudFormationStackSet` cria ou atualiza recursos; o que determinará isso é se a ação está sendo executada pela primeira vez ou não.

A ação `CloudFormationStackSet` cria ou atualiza o conjunto de pilhas e implanta essas alterações em instâncias especificadas.

> **Note**
>
> Se você usar essa ação para fazer uma atualização que inclua a adição de instâncias de pilha, as novas instâncias serão implantadas primeiro e a atualização será concluída por último. As novas instâncias recebem primeiro a versão antiga. Em seguida, a atualização é aplicada a todas as instâncias.

- Criar: quando nenhuma instância é especificada e o conjunto de pilhas não existe, a CloudFormationStackSet ação cria o conjunto de pilhas sem criar nenhuma instância.
- Atualização: quando a CloudFormationStackSet ação é executada para um conjunto de pilhas que já foi criado, a ação atualiza o conjunto de pilhas. Se nenhuma instância for especificada e o conjunto de pilhas já existir, todas as instâncias serão atualizadas. Se essa ação for usada para atualizar instâncias específicas, todas as instâncias restantes passarão para o status OUTDATED.

  Você pode usar a CloudFormationStackSet ação para atualizar o conjunto de pilhas das seguintes maneiras.

  - Atualize o modelo em algumas ou em todas as instâncias.

- Atualize os parâmetros em algumas ou em todas as instâncias.
- Atualize o perfil de execução do conjunto de pilhas (isso deve corresponder ao perfil de execução especificado no perfil Administrador).
- Altere o modelo de permissões (somente se nenhuma instância tiver sido criada).
- Enable/Disable `AutoDeployment`se o modelo de permissões do conjunto de pilhas for`Service Managed`.
- Atue como um administrador delegado em uma conta de membro se o modelo de permissões do conjunto de pilhas for `Service Managed`.
- Atualize o perfil Administrador.
- Atualize a descrição no conjunto de pilhas.
- Adicione destinos de implantação à atualização do conjunto de pilhas para criar novas instâncias de pilha.

A ação `CloudFormationStackInstances` cria novas instâncias de pilha ou atualiza instâncias de pilha desatualizadas. Uma instância fica desatualizada quando um conjunto de pilhas é atualizado, mas nem todas as instâncias dentro dela são atualizadas.

- Criar: se a pilha já existir, a ação `CloudFormationStackInstances` atualizará somente as instâncias e não criará instâncias da pilha.
- Atualizar: depois que a ação `CloudFormationStackSet` for executada, se o modelo ou os parâmetros tiverem sido atualizados somente em algumas instâncias, o restante será marcado como `OUTDATED`. Nos estágios posteriores do pipeline, `CloudFormationStackInstances` atualiza o restante das instâncias no conjunto de pilhas em ondas para que todas as instâncias sejam marcadas como `CURRENT`. Essa ação também pode ser usada para adicionar instâncias extras ou substituir parâmetros em instâncias novas ou existentes.

Como parte de uma atualização, as ações `CloudFormationStackSet` e `CloudFormationStackInstances` podem especificar novos destinos de implantação, o que cria novas instâncias de pilha.

Como parte de uma atualização, as ações `CloudFormationStackSet` e `CloudFormationStackInstances` não excluem conjuntos de pilhas, instâncias ou recursos. Quando a ação atualiza uma pilha, mas não especifica todas as instâncias a serem atualizadas, as instâncias que não foram especificadas para atualização são removidas da atualização e definidas para o status `OUTDATED`.

Durante uma implantação, as instâncias de pilha também poderão mostrar o status `OUTDATED` se a implantação nas instâncias falhar.

Como estruturar StackSets ações em um pipeline

Como prática recomendada, você deve criar o pipeline para que o conjunto de pilhas seja criado e implantado inicialmente em um subconjunto ou em uma única instância. Após testar a implantação e visualizar o conjunto de pilhas gerado, adicione a ação `CloudFormationStackInstances` para que as instâncias restantes sejam criadas e atualizadas.

Use o console ou a CLI para criar a estrutura de pipeline recomendada da seguinte maneira:

1. Crie um pipeline com uma ação de origem (obrigatória) e a ação `CloudFormationStackSet` como ação de implantação. Execute seu pipeline.
2. Quando seu pipeline é executado pela primeira vez, a ação `CloudFormationStackSet` cria seu conjunto de pilhas e pelo menos uma instância inicial. Verifique a criação do conjunto de pilhas e analise a implantação na sua instância inicial. Por exemplo, para a criação inicial do conjunto de pilhas para a conta Account-A onde `us-east-1` está a região especificada, a instância da pilha é criada com o conjunto de pilhas:

Instância da pilha	Região	Status
StackInstanceID-1	us-east-1	CURRENT

3. Edite seu pipeline para adicioná-lo `CloudFormationStackInstances` como segunda ação de implantação para create/update empilhar instâncias para os destinos que você designar. Por exemplo, para a criação de instâncias de pilha para a conta `Account-A` em que as regiões `us-east-2` e `eu-central-1` são especificadas, as instâncias de pilha restantes são criadas e a instância inicial permanece atualizada da seguinte maneira:

Instância da pilha	Região	Status
StackInstanceID-1	us-east-1	CURRENT
StackInstanceID-2	us-east-2	CURRENT
StackInstanceID-3	eu-central-1	CURRENT

4. Execute seu pipeline conforme necessário para atualizar seu conjunto de pilhas e atualize ou crie instâncias de pilha.

Quando você inicia uma atualização da pilha em que removeu os destinos de implantação da configuração da ação, as instâncias da pilha que não foram designadas para atualização são removidas da implantação e passam para o status OUTDATED. Por exemplo, na atualização da instância de pilha para a conta `Account-A` em que a região `us-east-2` é removida da configuração da ação, as instâncias de pilha restantes são criadas e a instância removida é definida como OUTDATED da seguinte maneira:

Instância da pilha	Região	Status
StackInstanceID-1	us-east-1	CURRENT
StackInstanceID-2	us-east-2	OUTDATED
StackInstanceID-3	eu-central-1	CURRENT

Para obter mais informações sobre as melhores práticas para implantar conjuntos de pilhas, consulte Práticas recomendadas para StackSets no Guia do AWS CloudFormation usuário.

A ação **`CloudFormationStackSet`**

Esta ação cria ou atualiza um conjunto de pilhas a partir do modelo armazenado no local de origem do pipeline.

Após definir um conjunto de pilhas, você pode criar, atualizar ou excluir pilhas nas contas e regiões de destino especificadas nos parâmetros de configuração. Ao criar, atualizar ou excluir pilhas, você também pode especificar preferências, como a ordem das regiões para que as operações sejam realizadas, o percentual de tolerância a falhas para interrupção das operações de pilhas e a quantidade de contas nas quais as operações serão executadas nas pilhas simultaneamente.

Um conjunto de pilhas é um recurso regional. Se você criar um conjunto de pilhas em uma AWS região, não poderá acessá-lo de outras regiões.

Quando essa ação é usada como uma ação de atualização para o conjunto de pilhas, as atualizações na pilha não são permitidas sem uma implantação em pelo menos uma instância da pilha.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Exemplo CloudFormationStackSet configuração de ação

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `CloudFormationStackSet`
- Versão: `1`

Parâmetros de configuração

StackSetName

Obrigatório: Sim

O nome a ser associado ao conjunto de pilhas. Esse nome deve ser exclusivo na região em que ele foi criado.

O nome pode conter apenas caracteres alfanuméricos e hifens. Ele deve começar com um caractere alfabético e ter 128 caracteres ou menos.

Description

Obrigatório: não

Uma descrição do conjunto de pilhas. Você pode usar isso para descrever a finalidade do conjunto de pilhas ou outras informações relevantes.

TemplatePath

Obrigatório: Sim

O local do modelo que define os recursos no conjunto de pilhas. Isso deve apontar para um modelo com um tamanho máximo de 460.800 bytes.

Insira o caminho para o nome do artefato de origem e o arquivo de modelo no formato `"InputArtifactName::TemplateFileName"`, conforme mostrado no exemplo a seguir.

```
SourceArtifact::template.txt
```

Parâmetros

Obrigatório: não

Uma lista de parâmetros de modelo para seu conjunto de pilhas que são atualizados durante uma implantação.

Você pode fornecer parâmetros como uma lista literal ou um caminho de arquivo:

- Você pode inserir parâmetros no seguinte formato de sintaxe abreviada: `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV` Para obter mais informações sobre tipos de dados, consulte Tipos de dados de parâmetro do modelo.

  O exemplo a seguir mostra um parâmetro chamado `BucketName` com o valor `amzn-s3-demo-source-bucket`.

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ```

  O exemplo a seguir mostra uma entrada com vários parâmetros:

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ParameterKey=Asset1,ParameterValue=true
  ParameterKey=Asset2,ParameterValue=true
  ```

- Você pode inserir o local do arquivo que contém uma lista de substituições de parâmetros de modelo inseridas no formato `"InputArtifactName::ParametersFileName"`, conforme mostrado no exemplo a seguir.

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

O exemplo a seguir mostra o conteúdo do arquivo `parameters.txt`.

```
[
    {
        "ParameterKey": "KeyName",
        "ParameterValue": "true"
    },
    {
        "ParameterKey": "KeyName",
        "ParameterValue": "true"
    }
]
```

Capacidades

Obrigatório: não

Indica que o modelo pode criar e atualizar recursos, dependendo dos tipos de recursos no modelo.

Você deve usar essa propriedade se tiver recursos do IAM em seu modelo de pilha ou criar uma pilha diretamente a partir de um modelo que contenha macros. Para que a CloudFormation ação opere com êxito dessa maneira, você deve usar um dos seguintes recursos:

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`

Você pode especificar mais de um recurso usando uma vírgula sem espaço entre os recursos. O exemplo em Exemplo CloudFormationStackSet configuração de ação mostra uma entrada com vários recursos.

PermissionModel

Obrigatório: não

Determina como os perfis do IAM são criados e gerenciados. Se o campo não for especificado, o padrão será usado. Para mais informações, consulte Modelos de permissões para operações de conjuntos de pilhas.

Os valores válidos são:

- `SELF_MANAGED` (padrão): você deve criar perfis de administrador e de execução para implantação em contas de destino.

- `SERVICE_MANAGED`: cria CloudFormation StackSets automaticamente as funções do IAM necessárias para implantação em contas gerenciadas por AWS organizações. Isso requer que uma conta seja membro de uma organização.

> **Note**
>
> Este parâmetro só pode ser alterado quando não há instâncias de pilha no conjunto de pilhas.

AdministrationRoleArn

> **Note**
>
> Como CloudFormation StackSets executa operações em várias contas, você deve definir as permissões necessárias nessas contas antes de criar o conjunto de pilhas.

Obrigatório: não

> **Note**
>
> Este parâmetro é opcional para o modelo de permissões SELF_MANAGED e não é usado para o modelo de permissões SERVICE_MANAGED.

O ARN do perfil do IAM na conta do administrador usada para realizar operações de conjunto de pilhas.

O nome pode conter caracteres alfanuméricos, qualquer um dos seguintes caracteres: _+=,.@- e sem espaços. O nome não faz distinção entre maiúsculas e minúsculas. Este nome de perfil deve ter um tamanho mínimo de 20 caracteres e um tamanho máximo de 2.048 caracteres. Os nomes de perfil deve ser exclusivo na conta. O nome do perfil especificado aqui deve ser um nome de perfil existente. Se você não especificar o nome do perfil, ele será definido como AWSCloudFormationStackSetAdministrationRole. Se você especificar ServiceManaged, não deverá definir um nome de função.

ExecutionRoleName

> **Note**
>
> Como CloudFormation StackSets executa operações em várias contas, você deve definir as permissões necessárias nessas contas antes de criar o conjunto de pilhas.

Obrigatório: não

> **Note**
>
> Este parâmetro é opcional para o modelo de permissões SELF_MANAGED e não é usado para o modelo de permissões SERVICE_MANAGED.

O nome do perfil do IAM nas contas de destino usadas para realizar operações de conjunto de pilhas. O nome pode conter caracteres alfanuméricos, qualquer um dos seguintes caracteres: _+=,.@- e sem espaços. O nome não faz distinção entre maiúsculas e minúsculas. Este nome de perfil deve ter um tamanho mínimo de 1 caractere e um tamanho máximo de 64 caracteres. Os nomes de perfil deve ser exclusivo na conta. O nome do perfil especificado aqui deve ser um nome de perfil existente. Não especifique esse perfil se você estiver usando perfis de execução personalizados. Se você não especificar o nome do perfil, ele será definido como `AWSCloudFormationStackSetExecutionRole`. Se você Service_Managed definir como verdadeiro, não deverá definir um nome de função.

OrganizationsAutoDeployment

Obrigatório: não

> **Note**
>
> Este parâmetro é opcional para o modelo de permissões SERVICE_MANAGED e não é usado para o modelo de permissões SELF_MANAGED.

Descreve se é implantado CloudFormation StackSets automaticamente AWS nas contas de organizações que são adicionadas a uma organização ou unidade organizacional (OU) de destino. Se `OrganizationsAutoDeployment` for especificado, não especifique `DeploymentTargets` e `Regions`.

> **Note**
>
> Se nenhuma entrada for fornecida para `OrganizationsAutoDeployment`, o valor padrão será `Disabled`.

Os valores válidos são:

- `Enabled`. Obrigatório: Não.

  StackSets implanta automaticamente instâncias de pilha adicionais em contas de AWS organizações que são adicionadas a uma organização ou unidade organizacional (OU) de destino nas regiões especificadas. Se uma conta for removida de uma organização ou OU de destino, CloudFormation StackSets excluirá as instâncias de pilha da conta nas regiões especificadas.
- `Disabled`. Obrigatório: Não.

  StackSets não implanta automaticamente instâncias de pilha adicionais em contas de AWS organizações que são adicionadas a uma organização ou unidade organizacional (OU) de destino nas regiões especificadas.
- `EnabledWithStackRetention`. Obrigatório: Não.

  Os recursos de pilha são retidos quando uma conta é removida de uma organização ou OU de destino.

DeploymentTargets

Obrigatório: não

> **Note**
>
> Para o modelo de permissões SERVICE_MANAGED, você pode fornecer o ID raiz da organização ou os IDs da unidade organizacional como destinos de implantação. Para o modelo de permissões SELF_MANAGED, você só pode fornecer contas.

> **Note**
>
> Quando este parâmetro é selecionado, você também deve selecionar Regions.

Uma lista de AWS contas ou IDs de unidades organizacionais onde deveriam estar created/updated as instâncias do conjunto de pilhas.

- Accounts:

  Você pode fornecer contas como uma lista literal ou um caminho de arquivo:

  - Literal: insira parâmetros no formato de sintaxe abreviada `account_ID,account_ID`, conforme mostrado no exemplo a seguir.

    ```
    111111222222,333333444444
    ```

  - Caminho do arquivo: o local do arquivo contendo uma lista de AWS contas em que as instâncias do conjunto de pilhas devem estar created/updated, inserido no formato`InputArtifactName::AccountsFileName`. Se você usar o caminho do arquivo para especificar contas ou OrganizationalUnitIds, o formato do arquivo deverá estar em JSON, conforme mostrado no exemplo a seguir.

    ```
    SourceArtifact::accounts.txt
    ```

    O exemplo a seguir mostra o conteúdo do arquivo `accounts.txt`.

    ```
    [
        "111111222222"
    ]
    ```

    O exemplo a seguir mostra o conteúdo do arquivo para `accounts.txt` ao listar mais de uma conta:

    ```
    [
        "111111222222","333333444444"
    ]
    ```

- OrganizationalUnitIds:

  > **Note**
  >
  > Este parâmetro é opcional para o modelo de permissões SERVICE_MANAGED e não é usado para o modelo de permissões SELF_MANAGED. Não use isso se você selecionar OrganizationsAutoDeployment.

As unidades AWS organizacionais nas quais atualizar as instâncias de pilha associadas.

Você pode fornecer IDs de unidade organizacional como uma lista literal ou um caminho de arquivo:

- Literal: insira uma matriz de strings separadas por vírgulas, conforme mostrado no exemplo a seguir.

```
ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
```

- Caminho do arquivo: a localização do arquivo que contém uma lista OrganizationalUnitIds na qual criar ou atualizar instâncias do conjunto de pilhas. Se você usar o caminho do arquivo para especificar contas ou OrganizationalUnitIds, o formato do arquivo deverá estar em JSON, conforme mostrado no exemplo a seguir.

  Insira um caminho para o arquivo no formato `InputArtifactName::OrganizationalUnitIdsFileName`.

```
SourceArtifact::OU-IDs.txt
```

  O exemplo a seguir mostra o conteúdo do arquivo `OU-IDs.txt`:

```
[
    "ou-examplerootid111-exampleouid111","ou-examplerootid222-exampleouid222"
]
```

Regiões

Obrigatório: não

> **Note**
>
> Quando esse parâmetro é selecionado, você também deve selecionar DeploymentTargets.

Uma lista de AWS regiões nas quais as instâncias do conjunto de pilhas são criadas ou atualizadas. As regiões são atualizadas na ordem em que são inseridas.

Insira uma lista de AWS regiões válidas no formato`Region1,Region2`, conforme mostrado no exemplo a seguir.

```
us-west-2,us-east-1
```

FailureTolerancePercentage

Obrigatório: não

A porcentagem de contas por região para as quais essa operação de empilhamento pode falhar antes de CloudFormation interromper a operação nessa região. Se a operação for interrompida em uma região, CloudFormation não tentará a operação nas regiões subsequentes. Ao calcular o número de contas com base na porcentagem especificada, CloudFormation arredonda para o próximo número inteiro.

MaxConcurrentPercentage

Obrigatório: não

A porcentagem máxima de contas em que essa operação pode ser executada ao mesmo tempo. Ao calcular o número de contas com base na porcentagem especificada, CloudFormation arredonda para o próximo número inteiro. Se o arredondamento para baixo resultar em zero, CloudFormation defina o número como um. Embora você use essa configuração para especificar o máximo, nas grandes implementações, o número real de contas utilizadas simultaneamente pode ser menor devido ao controle de utilização do serviço.

RegionConcurrencyType

Obrigatório: não

Você pode especificar se o conjunto de pilhas deve ser implantado entre Regiões da AWS sequencial ou paralelamente, configurando o parâmetro de implantação simultânea da região. Quando a simultaneidade de região é especificada para implantar pilhas em várias Regiões da AWS em paralelo, isso pode resultar em tempos gerais de implantação mais rápidos.

- Em paralelo: as implantações de conjuntos de pilhas serão conduzidas simultaneamente, desde que as falhas de implantação de uma região não excedam uma tolerância a falhas especificada.
- Sequencial: as implantações de conjuntos de pilhas serão conduzidas uma de cada vez, desde que as falhas de implantação de uma região não excedam uma tolerância a falhas especificada. A implantação sequencial é a seleção padrão.

ConcurrencyMode

Obrigatório: não

O modo de simultaneidade permite escolher como o nível de simultaneidade se comporta durante as operações de conjunto de pilhas, seja com tolerância da falhas estrita ou moderada. A Alta tolerância a falhas reduz a velocidade de implantação à medida que ocorrem falhas na operação do conjunto de pilhas porque a simultaneidade diminui para cada falha. A tolerância a falhas leves prioriza a velocidade de implantação e, ao mesmo tempo, aproveita CloudFormation os recursos de segurança.

- `STRICT_FAILURE_TOLERANCE`: essa opção reduz dinamicamente o nível de simultaneidade para garantir que o número de contas com falha nunca exceda uma tolerância a falhas específica. Esse é o comportamento padrão.
- `SOFT_FAILURE_TOLERANCE`: essa opção desacopla a tolerância a falhas da simultaneidade real. Isso permite que as operações de conjunto de pilhas em um nível de simultaneidade definido, independentemente do número de falhas.

CallAs

Obrigatório: não

> **Note**
>
> Este parâmetro é opcional para o modelo de permissões `SERVICE_MANAGED` e não é usado para o modelo de permissões `SELF_MANAGED`.

Especifica se você está atuando na conta de gerenciamento da organização ou como administrador delegado em uma conta de membro.

> **Note**
>
> Se este parâmetro estiver definido como `DELEGATED_ADMIN`, garanta que o perfil do IAM do pipeline tenha a permissão `organizations:ListDelegatedAdministrators`. Caso contrário, a ação falhará durante a execução com um erro semelhante a este: `Account used is not a delegated administrator`.

- `SELF`: a implantação do conjunto de pilhas usará permissões gerenciadas pelo serviço enquanto estiver conectado à conta de gerenciamento.
- `DELEGATED_ADMIN`: a implantação do conjunto de pilhas usará permissões gerenciadas pelo serviço enquanto estiver conectado a uma conta de administrador delegado.

Input artifacts (Artefatos de entrada)

Você deve incluir pelo menos um artefato de entrada que contenha o modelo para o conjunto de pilhas em uma ação `CloudFormationStackSet`. Você pode incluir mais artefatos de entrada para listas de destinos, contas e parâmetros de implantação.

- Número de artefatos: `1 to 3`
- Descrição: você pode incluir artefatos para fornecer:
  - O arquivo de modelo de pilha. (Consulte o parâmetro `TemplatePath`.)
  - O arquivo de parâmetros. (Consulte o parâmetro `Parameters`.)
  - O arquivo de contas. (Consulte o parâmetro `DeploymentTargets`.)

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Variáveis de saída

Se você configurar essa ação, ela produzirá variáveis que podem ser referenciadas pela configuração de uma ação downstream no pipeline. Configure uma ação com um namespace a fim de disponibilizar as variáveis para a configuração de ações downstream.

- StackSetId: O ID do conjunto de pilhas.
- OperationId: O ID da operação de conjunto de pilhas.

Para obter mais informações, consulte Referência de variáveis.

Exemplo CloudFormationStackSet configuração de ação

Os exemplos a seguir mostram a configuração da CloudFormationStackSet ação.

Exemplo do modelo de permissões autogerenciadas

O exemplo a seguir mostra uma CloudFormationStackSet ação em que o alvo de implantação inserido é um ID de AWS conta.

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"
}
```

Exemplo do modelo de permissões autogerenciadas

O exemplo a seguir mostra uma CloudFormationStackSet ação para o modelo de permissões gerenciadas por serviços em que a opção de implantação automática em AWS organizações é ativada com retenção de pilha.

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"
}
```

A CloudFormationStackInstances ação

Essa ação cria novas instâncias e implanta conjuntos de pilhas em instâncias especificadas. Uma instância de pilha é uma referência a uma pilha em uma conta de destino dentro de uma região. É possível haver uma instância de pilha sem uma pilha. Por exemplo, se a criação da pilha não for bem-sucedida, a instância de pilha mostrará o motivo da falha da criação da pilha. Uma instância de pilha está associada a apenas um conjunto de pilhas.

Após a criação inicial de um conjunto de pilhas, você pode adicionar novas instâncias de pilha usando `CloudFormationStackInstances`. Os valores dos parâmetros do modelo podem ser substituídos no nível da instância da pilha durante as operações de criação ou atualização da instância do conjunto de pilhas.

Cada conjunto de pilhas tem um modelo e um conjunto de parâmetros de modelo. Ao atualizar o modelo ou os parâmetros do modelo, você os atualiza para todo o conjunto. Em seguida, todos os status da instância são definidos como `OUTDATED` até que as alterações sejam implantadas nessa instância.

Para substituir valores de parâmetros em instâncias específicas, por exemplo, se o modelo contiver um parâmetro para `stage` com um valor `prod`, você poderá substituir o valor desse parâmetro para que seja `beta` ou `gamma`.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Exemplo de configuração da ação

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `CloudFormationStackInstances`
- Versão: `1`

Parâmetros de configuração

StackSetName

Obrigatório: Sim

O nome a ser associado ao conjunto de pilhas. Esse nome deve ser exclusivo na região em que ele foi criado.

O nome pode conter apenas caracteres alfanuméricos e hifens. Ele deve começar com um caractere alfabético e ter 128 caracteres ou menos.

DeploymentTargets

Obrigatório: não

> **Note**
>
> Para o modelo de permissões SERVICE_MANAGED, você pode fornecer o ID raiz da organização ou os IDs da unidade organizacional como destinos de implantação. Para o modelo de permissões SELF_MANAGED, você só pode fornecer contas.

> **Note**
>
> Quando este parâmetro é selecionado, você também deve selecionar Regions.

Uma lista de AWS contas ou IDs de unidades organizacionais onde deveriam estar created/updated as instâncias do conjunto de pilhas.

- Accounts:

  Você pode fornecer contas como uma lista literal ou um caminho de arquivo:

  - Literal: insira parâmetros no formato de sintaxe abreviada `account_ID,account_ID`, conforme mostrado no exemplo a seguir.

    ```
    111111222222,333333444444
    ```

  - Caminho do arquivo: o local do arquivo contendo uma lista de AWS contas em que as instâncias do conjunto de pilhas devem estar created/updated, inserido no formato`InputArtifactName::AccountsFileName`. Se você usar o caminho do arquivo para especificar contas ou OrganizationalUnitIds, o formato do arquivo deverá estar em JSON, conforme mostrado no exemplo a seguir.

    ```
    SourceArtifact::accounts.txt
    ```

    O exemplo a seguir mostra o conteúdo do arquivo `accounts.txt`:

    ```
    [
        "111111222222"
    ```

```
]
```

O exemplo a seguir mostra o conteúdo do arquivo para `accounts.txt` ao listar mais de uma conta:

```
[
    "111111222222","333333444444"
]
```

- OrganizationalUnitIds:

> **Note**
>
> Este parâmetro é opcional para o modelo de permissões SERVICE_MANAGED e não é usado para o modelo de permissões SELF_MANAGED. Não use isso se você selecionar OrganizationsAutoDeployment.

As unidades AWS organizacionais nas quais atualizar as instâncias de pilha associadas.

Você pode fornecer IDs de unidade organizacional como uma lista literal ou um caminho de arquivo.

- Literal: insira uma matriz de strings separadas por vírgulas, conforme mostrado no exemplo a seguir.

```
ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
```

- Caminho do arquivo: a localização do arquivo que contém uma lista OrganizationalUnitIds na qual criar ou atualizar instâncias do conjunto de pilhas. Se você usar o caminho do arquivo para especificar contas ou OrganizationalUnitIds, o formato do arquivo deverá estar em JSON, conforme mostrado no exemplo a seguir.

  Insira um caminho para o arquivo no formato `InputArtifactName::OrganizationalUnitIdsFileName`.

```
SourceArtifact::OU-IDs.txt
```

O exemplo a seguir mostra o conteúdo do arquivo `OU-IDs.txt`:

```
[
    "ou-examplerootid111-exampleouid111","ou-examplerootid222-exampleouid222"
]
```

Regiões

Obrigatório: Sim

> **Note**
>
> Quando esse parâmetro é selecionado, você também deve selecionar DeploymentTargets.

Uma lista de AWS regiões nas quais as instâncias do conjunto de pilhas são criadas ou atualizadas. As regiões são atualizadas na ordem em que são inseridas.

Insira uma lista de AWS regiões válidas no formato:`Region1,Region2`, conforme mostrado no exemplo a seguir.

```
us-west-2,us-east-1
```

ParameterOverrides

Obrigatório: não

Uma lista de parâmetros de conjunto de pilhas que você deseja substituir nas instâncias de pilha selecionadas. Os valores de parâmetros substituídos são aplicados a todas as instâncias da pilha nas contas e regiões especificadas.

Você pode fornecer parâmetros como uma lista literal ou um caminho de arquivo:

- Você pode inserir parâmetros no seguinte formato de sintaxe abreviada:
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  Para obter mais informações sobre tipos de dados, consulte Tipos de dados de parâmetro do modelo.

  O exemplo a seguir mostra um parâmetro chamado `BucketName` com o valor `amzn-s3-demo-source-bucket`.

```
ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
```

O exemplo a seguir mostra uma entrada com vários parâmetros.

```
ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
ParameterKey=Asset1,ParameterValue=true
ParameterKey=Asset2,ParameterValue=true
```

- Você pode inserir o local do arquivo que contém uma lista de substituições de parâmetros de modelo inseridas no formato `InputArtifactName::ParameterOverridesFileName`, conforme mostrado no exemplo a seguir.

```
SourceArtifact::parameter-overrides.txt
```

O exemplo a seguir mostra o conteúdo do arquivo `parameter-overrides.txt`.

```
[
    {
        "ParameterKey": "KeyName",
        "ParameterValue": "true"
    },
    {
        "ParameterKey": "KeyName",
        "ParameterValue": "true"
    }
]
```

FailureTolerancePercentage

Obrigatório: não

A porcentagem de contas por região para as quais essa operação de empilhamento pode falhar antes de CloudFormation interromper a operação nessa região. Se a operação for interrompida em uma região, CloudFormation não tentará a operação nas regiões subsequentes. Ao calcular o número de contas com base na porcentagem especificada, CloudFormation arredonda para o próximo número inteiro.

MaxConcurrentPercentage

Obrigatório: não

A porcentagem máxima de contas em que essa operação será executada em cada momento. Ao calcular o número de contas com base na porcentagem especificada, CloudFormation arredonda para o próximo número inteiro. Se o arredondamento para baixo resultar em zero, CloudFormation defina o número como um. Embora você especifique o máximo, nas grandes implantações, o número real de contas utilizadas simultaneamente pode ser menor devido ao controle de utilização do serviço.

RegionConcurrencyType

Obrigatório: não

Você pode especificar se o conjunto de pilhas deve ser implantado entre Regiões da AWS sequencial ou paralelamente, configurando o parâmetro de implantação simultânea da região. Quando a simultaneidade de região é especificada para implantar pilhas em várias Regiões da AWS em paralelo, isso pode resultar em tempos gerais de implantação mais rápidos.

- Em paralelo: as implantações de conjuntos de pilhas serão conduzidas simultaneamente, desde que as falhas de implantação de uma região não excedam uma tolerância a falhas especificada.
- Sequencial: as implantações de conjuntos de pilhas serão conduzidas uma de cada vez, desde que as falhas de implantação de uma região não excedam uma tolerância a falhas especificada. A implantação sequencial é a seleção padrão.

ConcurrencyMode

Obrigatório: não

O modo de simultaneidade permite escolher como o nível de simultaneidade se comporta durante as operações de conjunto de pilhas, seja com tolerância da falhas estrita ou moderada. A Alta tolerância a falhas reduz a velocidade de implantação à medida que ocorrem falhas na operação do conjunto de pilhas porque a simultaneidade diminui para cada falha. A tolerância a falhas leves prioriza a velocidade de implantação e, ao mesmo tempo, aproveita CloudFormation os recursos de segurança.

- `STRICT_FAILURE_TOLERANCE`: essa opção reduz dinamicamente o nível de simultaneidade para garantir que o número de contas com falha nunca exceda uma tolerância a falhas específica. Esse é o comportamento padrão.

- `SOFT_FAILURE_TOLERANCE`: essa opção desacopla a tolerância a falhas da simultaneidade real. Isso permite que as operações de conjunto de pilhas em um nível de simultaneidade definido, independentemente do número de falhas.

CallAs

Obrigatório: não

> **Note**
>
> Este parâmetro é opcional para o modelo de permissões `SERVICE_MANAGED` e não é usado para o modelo de permissões `SELF_MANAGED`.

Especifica se você está atuando na conta de gerenciamento da organização ou como administrador delegado em uma conta de membro.

> **Note**
>
> Se este parâmetro estiver definido como `DELEGATED_ADMIN`, garanta que o perfil do IAM do pipeline tenha a permissão `organizations:ListDelegatedAdministrators`. Caso contrário, a ação falhará durante a execução com um erro semelhante a este: `Account used is not a delegated administrator`.

- `SELF`: a implantação do conjunto de pilhas usará permissões gerenciadas pelo serviço enquanto estiver conectado à conta de gerenciamento.
- `DELEGATED_ADMIN`: a implantação do conjunto de pilhas usará permissões gerenciadas pelo serviço enquanto estiver conectado a uma conta de administrador delegado.

Input artifacts (Artefatos de entrada)

`CloudFormationStackInstances` pode conter artefatos que listam destinos e parâmetros de implantação.

- Número de artefatos: `0 to 2`
- Descrição: como entrada, a ação do conjunto de pilhas aceita opcionalmente artefatos para as seguintes finalidades:

- Para fornecer o arquivo de parâmetros a ser usado. (Consulte o parâmetro `ParameterOverrides`.)
- Para fornecer o arquivo de contas de destino a ser usado. (Consulte o parâmetro `DeploymentTargets`.)

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Variáveis de saída

Quando configurada, essa ação produz variáveis que podem ser referenciadas pela configuração de ação de uma ação downstream no pipeline. Configure uma ação com um namespace a fim de disponibilizar as variáveis para a configuração de ações downstream.

- StackSetId: O ID do conjunto de pilhas.
- OperationId: O ID da operação de conjunto de pilhas.

Para obter mais informações, consulte Referência de variáveis.

Exemplo de configuração da ação

Os exemplos a seguir mostram a configuração da CloudFormationStackInstances ação.

Exemplo do modelo de permissões autogerenciadas

O exemplo a seguir mostra uma CloudFormationStackInstances ação em que o alvo de implantação inserido é um Conta da AWS ID111111222222.

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"
}
```

Exemplo do modelo de permissões autogerenciadas

O exemplo a seguir mostra uma CloudFormationStackInstances ação para o modelo de permissões gerenciadas por serviços em que o alvo da implantação é uma ID de unidade organizacional da AWS organização. `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"
}
```

Permissões do perfil de serviço: ação **`CloudFormationStackSet`**

Para CloudFormation StackSets ações, as seguintes permissões mínimas são necessárias.

Para a ação `CloudFormationStackSet`, adicione o seguinte à declaração da política:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackSet",
        "cloudformation:UpdateStackSet",
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:DescribeStackSet",
        "cloudformation:ListStackInstances"
    ],
    "Resource": "resource_ARN"
},
```

Permissões do perfil de serviço: ação **`CloudFormationStackInstances`**

Para a ação `CloudFormationStackInstances`, adicione o seguinte à declaração da política:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation"
    ],
    "Resource": "resource_ARN"
},
```

Modelos de permissões para operações de conjuntos de pilhas

Como CloudFormation StackSets executa operações em várias contas, você deve definir as permissões necessárias nessas contas antes de criar o conjunto de pilhas. Você pode definir permissões por meio de permissões autogerenciadas ou permissões gerenciadas por serviços.

Com permissões autogerenciadas, você cria as duas funções do IAM exigidas por StackSets : uma função de administrador, como a da conta AWSCloudFormationStackSetAdministrationRole em

que você define o conjunto de pilhas e uma função de execução, como a de cada uma das contas AWSCloudFormationStackSetExecutionRole em que você implanta instâncias do conjunto de pilhas. Usando esse modelo de permissões, StackSets pode implantar em qualquer AWS conta na qual o usuário tenha permissões para criar uma função do IAM. Para obter mais informações, consulte Conceder permissões autogerenciadas no Guia do usuário do AWS CloudFormation .

> **Note**
>
> Como CloudFormation StackSets executa operações em várias contas, você deve definir as permissões necessárias nessas contas antes de criar o conjunto de pilhas.

Com permissões gerenciadas por serviços, você pode implantar instâncias de pilha em contas gerenciadas por organizações. AWS Usando esse modelo de permissões, você não precisa criar as funções do IAM necessárias, pois StackSets cria as funções do IAM em seu nome. Com esse modelo, você também pode habilitar implantações automáticas em contas que serão adicionadas à sua organização no futuro. Consulte Habilitar acesso confiável com AWS organizações no Guia AWS CloudFormation do usuário.

Tipos de dados de parâmetro do modelo

Os parâmetros do modelo usados nas operações de conjunto de pilhas incluem os seguintes tipos de dados. Para obter mais informações, consulte DescribeStackSet.

ParameterKey

- Descrição: a chave associada ao parâmetro. Se você não especificar uma chave e um valor para um parâmetro específico, AWS CloudFormation use o valor padrão especificado no modelo.
- Exemplo:

```
"ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket"
```

ParameterValue

- Descrição: o valor de entrada associado ao parâmetro.
- Exemplo:

```
"ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket"
```

UsePreviousValue

- Durante uma atualização da pilha, use o valor do parâmetro existente que a pilha está usando para uma determinada chave de parâmetro. Se você especificar `true`, não especifique um valor de parâmetro.
- Exemplo:

```
"ParameterKey=Asset1,UsePreviousValue=true"
```

Cada conjunto de pilhas tem um modelo e um conjunto de parâmetros de modelo. Ao atualizar o modelo ou os parâmetros do modelo, você os atualiza para todo o conjunto. Em seguida, todos os status de instância são definidos como OUTDATED até que as alterações sejam implantadas nessa instância.

Para substituir valores de parâmetros em instâncias específicas, por exemplo, se o modelo contiver um parâmetro para `stage` com um valor `prod`, você poderá substituir o valor desse parâmetro para que seja `beta` ou `gamma`.

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Tipos de parâmetros — Este capítulo de referência no Guia do AWS CloudFormation usuário fornece mais descrições e exemplos de parâmetros CloudFormation do modelo.
- Práticas recomendadas: para obter mais informações sobre as práticas recomendadas para implantar conjuntos de pilhas, consulte https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-bestpractices.html no Guia do usuário do AWS CloudFormation .
- AWS CloudFormation Referência da API — Você pode consultar as seguintes CloudFormation ações na Referência da AWS CloudFormation API para obter mais informações sobre os parâmetros usados nas operações de conjunto de pilhas:
  - A CreateStackSet ação cria um conjunto de pilhas.
  - A UpdateStackSet ação atualiza o conjunto de pilhas e as instâncias de pilha associadas nas contas e regiões especificadas. Mesmo que a operação de conjunto de pilhas criada pela atualização do conjunto de pilhas falhe (total ou parcialmente, abaixo ou acima de uma tolerância a falhas especificada), o conjunto de pilhas será atualizado com essas alterações.

CreateStackInstances As chamadas subsequentes no conjunto de pilhas especificado usam o conjunto de pilhas atualizado.

- A CreateStackInstances ação cria uma instância de pilha para todas as regiões especificadas em todas as contas especificadas em um modelo de permissão autogerenciado ou em todos os alvos de implantação especificados em um modelo de permissão gerenciado por serviços. Você pode substituir os parâmetros das instâncias criadas por essa ação. Se as instâncias já existirem, CreateStackInstances faça chamadas UpdateStackInstances com os mesmos parâmetros de entrada. Quando você usa essa ação para criar instâncias, ela não altera o status de outras instâncias de pilha.
- A UpdateStackInstances ação atualiza as instâncias da pilha com o conjunto de pilhas para todas as regiões especificadas em todas as contas especificadas em um modelo de permissão autogerenciado ou em todos os alvos de implantação especificados em um modelo de permissão gerenciado por serviços. Você pode substituir os parâmetros das instâncias atualizadas por essa ação. Quando você usa essa ação para criar um subconjunto de instâncias, ela não altera o status de outras instâncias de pilha.
- A DescribeStackSetOperation ação retorna a descrição da operação especificada do conjunto de pilhas.
- A DescribeStackSet ação retorna a descrição do conjunto de pilhas especificado.

AWS CodeBuild referência de ação de construção e teste

Permite executar compilações e testes como parte do pipeline. Quando você executa uma ação de CodeBuild construção ou teste, os comandos especificados na especificação de construção são executados dentro de um contêiner. CodeBuild Todos os artefatos especificados como artefatos de entrada para uma CodeBuild ação estão disponíveis dentro do contêiner que executa os comandos. CodeBuild pode fornecer uma ação de construção ou teste. Para obter mais informações, consulte o Guia do usuário do AWS CodeBuild.

Quando você usa o CodePipeline assistente no console para criar um projeto de compilação, o projeto de CodeBuild compilação mostra que o provedor de origem é CodePipeline. Ao criar um projeto de compilação no CodeBuild console, você não pode especificar CodePipeline como provedor de origem, mas adicionar a ação de compilação ao seu pipeline ajusta a origem no CodeBuild console. Para obter mais informações, consulte ProjectSource na Referência de APIs do AWS CodeBuild .

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Permissões da função de serviço: CodeBuild ação
- Declaração de ação (CodeBuild exemplo)
- Consulte também

Tipo de ação

- Categoria: `Build` ou `Test`
- Proprietário: `AWS`
- Fornecedor: `CodeBuild`
- Versão: `1`

Parâmetros de configuração

ProjectName

Obrigatório: Sim

`ProjectName`é o nome do projeto de construção em CodeBuild.

PrimarySource

Obrigatório: condicional

O valor do `PrimarySource` parâmetro deve ser o nome de um dos artefatos de entrada da ação. CodeBuild procura o arquivo buildspec e executa os comandos buildspec no diretório que contém a versão descompactada desse artefato.

Esse parâmetro será necessário se vários artefatos de entrada forem especificados para uma ação do CodeBuild . Quando houver apenas um artefato de origem para a ação, o artefato `PrimarySource` assumirá esse artefato como padrão.

BatchEnabled

Obrigatório: não

O valor booleano do parâmetro `BatchEnabled` permite que a ação execute várias compilações na mesma execução de compilação.

Quando esta opção está habilitada, a opção `CombineArtifacts` está disponível.

Para exemplos de pipeline com compilações em lote habilitadas, consulte CodePipeline Integração com CodeBuild e compilações em lote.

BuildspecOverride

Obrigatório: não

Uma declaração de buildspec em linha ou do arquivo buildspec que substitui a mais recente definida no projeto de compilação, apenas para esta compilação. O buildspec definido no projeto não é alterado.

Se for definido, esse valor poderá ser um dos seguintes:

- Uma definição de buildspec em linha. Para obter mais informações, consulte a referência de sintaxe em Sintaxe de buildspec.
- O caminho para um arquivo buildspec alternativo relativo ao valor da variável de ambiente `CODEBUILD_SRC_DIR` interna ou ao caminho para um bucket do S3. O bucket deve estar na mesma Região da AWS do projeto de compilação. Especifique o arquivo buildspec usando seu ARN (por exemplo, `arn:aws:s3:::my-codebuild-sample2/buildspec.yml`). Se esse valor não for fornecido ou for definido como uma string vazia, o código-fonte deverá conter um arquivo buildspec no diretório raiz. Para saber mais sobre como adicionar um caminho, consulte Nome do arquivo e local de armazenamento de buildspec.

> **Note**
>
> Como essa propriedade permite a você alterar os comandos de compilação que serão executados no contêiner, observe se uma entidade principal do IAM com a capacidade de chamar essa API e definir esse parâmetro consegue substituir as configurações padrão. Além disso, é recomendável usar um local de buildspec confiável, como um arquivo no repositório de origem ou um bucket do Amazon S3.

CombineArtifacts

Obrigatório: não

O valor booleano do parâmetro `CombineArtifacts` combina todos os artefatos de uma compilação em lote em um único arquivo de artefato para a ação de compilação.

Para usar esta opção, o parâmetro `BatchEnabled` deve estar ativado.

EnvironmentVariables

Obrigatório: não

O valor desse parâmetro é usado para definir variáveis de ambiente para a ação do CodeBuild no pipeline. O valor do parâmetro `EnvironmentVariables` assume a forma de uma matriz JSON de objetos de variáveis de ambiente. Consulte o parâmetro de exemplo em Declaração de ação (CodeBuild exemplo).

Cada objeto tem três partes, todas são strings:

- `name`: o nome ou a chave da variável de ambiente.
- `value`: o valor da variável de ambiente. Ao usar o `SECRETS_MANAGER` tipo `PARAMETER_STORE` ou, esse valor deve ser o nome de um parâmetro que você já armazenou no Armazenamento de parâmetros do AWS Systems Manager ou um segredo que você já armazenou no AWS Secrets Manager, respectivamente.

> **Note**
>
> Nós desencorajamos fortemente o uso de variáveis de ambiente para armazenar valores confidenciais, especialmente AWS credenciais. Quando você usa o CodeBuild console ou a AWS CLI, as variáveis de ambiente são exibidas em texto sem formatação. Para valores confidenciais, recomendamos usar o tipo `SECRETS_MANAGER`.

- `type`: (opcional) o tipo da variável de ambiente. Os valores válidos são `PARAMETER_STORE`, `SECRETS_MANAGER` ou `PLAINTEXT`. Se não especificado, assumirá como padrão `PLAINTEXT`.

> **Note**
>
> Ao inserir `namevalue`, e `type` para sua configuração de variáveis de ambiente, especialmente se a variável de ambiente contiver sintaxe CodePipeline de variável de

saída, não exceda o limite de 1000 caracteres para o campo de valor da configuração. Um erro de validação será retornado quando esse limite for excedido.

Para obter mais informações, consulte EnvironmentVariable na Referência AWS CodeBuild da API. Para ver um exemplo de CodeBuild ação com uma variável de ambiente que resolve para o nome da GitHub ramificação, consulteExemplo: use uma BranchName variável com variáveis de CodeBuild ambiente.

ServiceRoleArnOverride

Obrigatório: não

Uma função do IAM ARN que CodeBuild usa como função de serviço para compilações acionadas por essa ação, substituindo a função de serviço padrão configurada no projeto. CodeBuild Quando ausente, o CodeBuild projeto usa sua função de serviço padrão.

A função especificada deve ter uma política de confiança que permita que o `codebuild.amazonaws.com` serviço a assuma. Tanto a identidade usada para criar ou atualizar o pipeline quanto a função de serviço do pipeline devem ter `iam:PassRole` permissão para a função especificada. Para visualizar um exemplo de política, consulte Permissões da função de serviço: CodeBuild ação.

Como prática recomendada de segurança, quando um CodeBuild projeto é compartilhado em vários pipelines, recomendamos especificar uma função com escopo definido para cada pipeline de forma a seguir o princípio do menor privilégio. Para pipelines que usam ações de CodeConnections origem, veja um Adicione CodeBuild GitClone permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com exemplo de como definir o escopo das permissões de conexão da função com um repositório específico.

> **Note**
>
> A função deve estar na mesma AWS conta do CodeBuild projeto.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `1 to 5`

- Descrição: CodeBuild procura o arquivo buildspec e executa os comandos buildspec no diretório do artefato de origem primária. Quando uma única fonte de entrada é especificada ou quando mais de uma fonte de entrada é especificada para a CodeBuild ação, o artefato único, ou o artefato primário, no caso de várias fontes de entrada, deve ser definido usando o parâmetro de configuração da `PrimarySource` ação in. CodePipeline

  Cada artefato de entrada é extraído para seu próprio diretório, cujos locais são armazenados em variáveis de ambiente. O diretório do artefato de origem primária é disponibilizado com `$CODEBUILD_SRC_DIR`. Os diretórios de todos os outros artefatos de entrada são disponibilizados com `$CODEBUILD_SRC_DIR_yourInputArtifactName`.

  > **Note**
  >
  > O artefato configurado em seu CodeBuild projeto se torna o artefato de entrada usado pela CodeBuild ação em seu funil.

Artefatos de saída

- Número de artefatos: `0 to 5`
- Descrição: eles podem ser usados para disponibilizar os artefatos definidos no arquivo CodeBuild buildspec para ações subsequentes no pipeline. Quando apenas um artefato de saída é definido, esse artefato pode ser definido diretamente na seção `artifacts` do arquivo buildspec. Quando mais de um artefato de saída é especificado, todos os artefatos referenciados devem ser definidos como artefatos secundários no arquivo buildspec. Os nomes dos artefatos de saída CodePipeline devem corresponder aos identificadores de artefatos no arquivo buildspec.

  > **Note**
  >
  > O artefato configurado em seu CodeBuild projeto se torna o artefato CodePipeline de entrada em sua ação de pipeline.

  Se o parâmetro `CombineArtifacts` for selecionado para compilações em lote, o local do artefato de saída conterá os artefatos combinados de várias compilações processadas na mesma execução.

Variáveis de saída

Esta ação produzirá como variáveis todas as variáveis de ambiente que foram exportadas como parte da compilação. Para obter mais detalhes sobre como exportar variáveis de ambiente, consulte EnvironmentVariable o Guia AWS CodeBuild da API.

Para obter mais informações sobre o uso de variáveis de CodeBuild ambiente em CodePipeline, consulte os exemplos emCodeBuild variáveis de saída da ação. Para obter uma lista das variáveis de ambiente que você pode usar CodeBuild, consulte Variáveis de ambiente em ambientes de construção no Guia AWS CodeBuild do usuário.

Permissões da função de serviço: CodeBuild ação

Para obter CodeBuild suporte, adicione o seguinte à sua declaração 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"
        }
    ]
}
```

Se você usar o `ServiceRoleArnOverride` parâmetro, adicione a seguinte `iam:PassRole` permissão para permitir CodePipeline a transmissão da função para CodeBuild:

```
{
```

```
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::111122223333:role/your-codebuild-override-role",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": "codebuild.amazonaws.com"
        }
    }
}
```

Declaração de ação (CodeBuild exemplo)

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"
                }
            ]
        }
    ]
}
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- AWS CodeBuild Guia do usuário — Para ver um exemplo de pipeline com uma CodeBuild ação, consulte Usar CodePipeline com CodeBuild para testar código e executar compilações. Para exemplos de projetos com vários CodeBuild artefatos de entrada e saída, consulte Amostra de CodePipeline integração com CodeBuild várias fontes de entrada e artefatos de saída e amostra de várias fontes de entrada e artefatos de saída.
- Tutorial: crie um pipeline que crie e teste seu aplicativo Android com AWS Device Farm— Este tutorial fornece um exemplo de arquivo buildspec e um aplicativo de amostra para criar um pipeline com uma GitHub fonte que cria e testa um aplicativo Android com e. CodeBuild AWS Device Farm
- Referência de especificação de compilação para CodeBuild — Este tópico de referência fornece definições e exemplos para entender os arquivos CodeBuild buildspec. Para obter uma lista das variáveis de ambiente que você pode usar CodeBuild, consulte Variáveis de ambiente em ambientes de construção no Guia AWS CodeBuild do usuário.

AWS CodePipeline invocar referência de ação

Você usa uma ação de CodePipeline invocação para simplificar o acionamento de execuções de pipeline downstream e a transmissão de variáveis de pipeline e revisões de origem entre pipelines.

> **Note**
>
> Essa ação é compatível somente para pipelines tipo V2.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Permissões da política de função de serviço para a CodePipeline ação de invocação
- Declaração de ação
- Consulte também

Tipo de ação

- Categoria: `Invoke`

- Proprietário: `AWS`
- Fornecedor: `CodePipeline`
- Versão: `1`

Parâmetros de configuração

PipelineName

Obrigatório: Sim

O nome do pipeline que, ao ser executado, iniciará o pipeline de destino atual. Você já deve ter criado o pipeline de invocação. A ação iniciará o pipeline `s3-pipeline-test` (de destino) quando o pipeline (invocação) chamado `my-s3-pipeline` iniciar uma execução.

SourceRevisions

Obrigatório: não

As revisões de origem que você deseja que o pipeline de destino use quando for iniciado pelo pipeline de invocação. Por exemplo, uma ação de origem do S3 fornece variáveis de saída, como ID da versão e chave do objeto do S3. Você pode especificar um valor de revisão a ser usado quando o pipeline for invocado.

Para a CLI, você especifica as revisões de origem como uma string JSON serializada. Para obter mais informações sobre como usar substituições de revisão de origem, consulte o SourceRevisionOverride Guia da CodePipeline API.

O mapeamento usa um formato de string conforme mostrado no seguinte exemplo:

```
[{"actionName":"Source","revisionType":"S3_OBJECT_VERSION_ID","revision
Value":"zq8mjNEXAMPLE"}]
```

Variáveis

Obrigatório: não

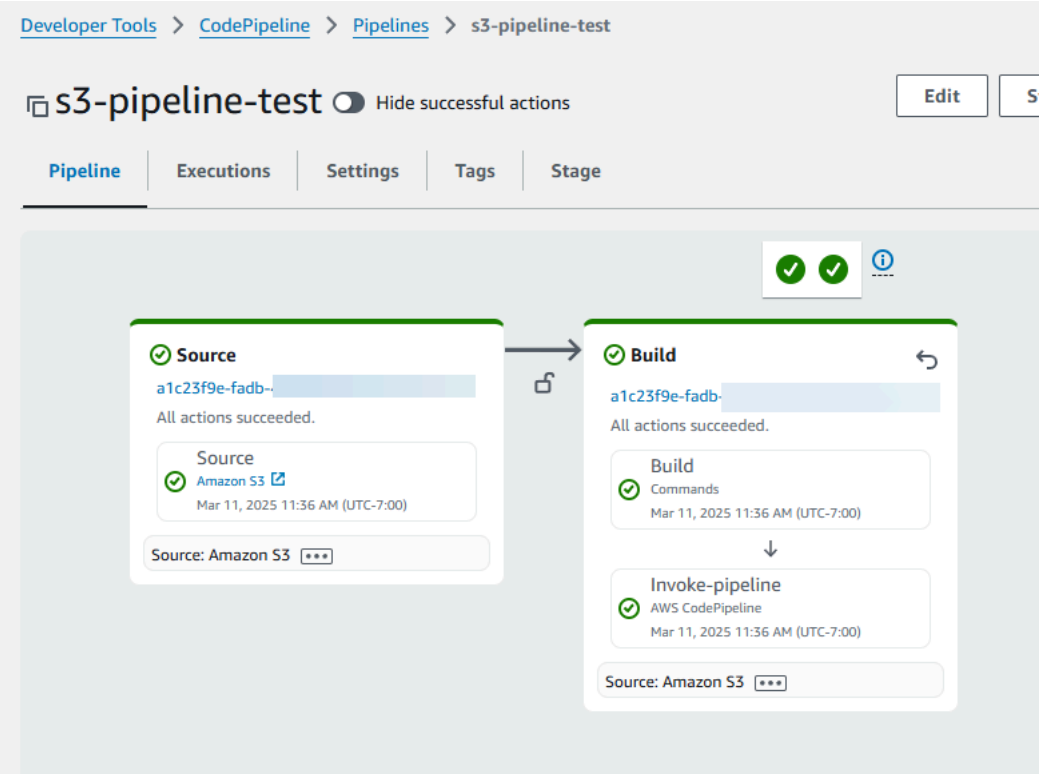
Os nomes e os valores de variáveis com os quais você deseja que a ação seja compatível.

Para a CLI, você especifica variáveis como uma string JSON serializada. Para obter mais informações sobre o uso de variáveis de pipeline, consulte PipelineVariable o Guia de CodePipeline API.

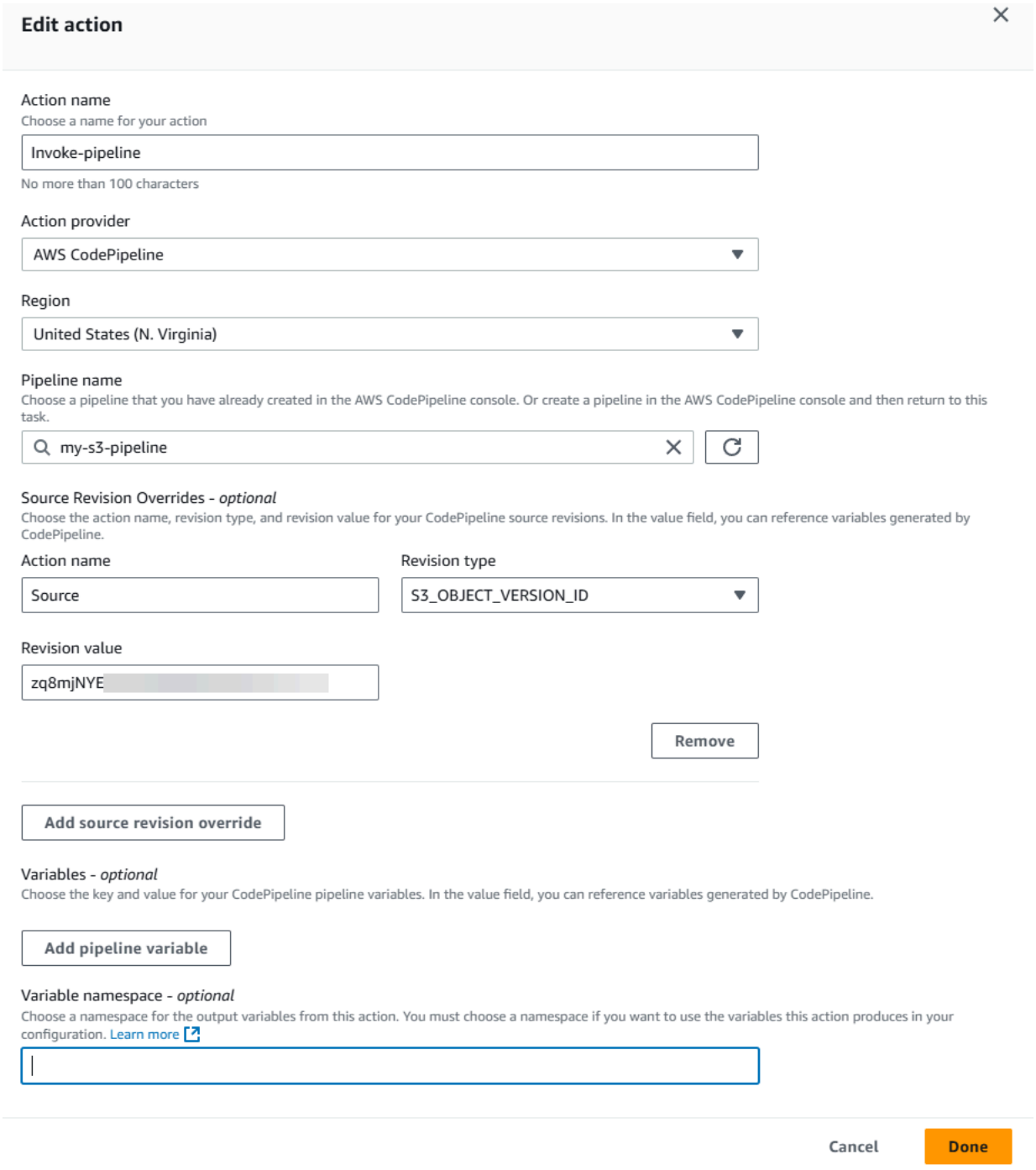
O mapeamento usa um formato de string conforme mostrado no seguinte exemplo:

```
[{"name":"VAR1","value":"VALUE1"}]
```

A imagem a seguir mostra um exemplo da ação adicionada a um pipeline no console.



A imagem a seguir mostra um exemplo da página Editar da ação. No exemplo a seguir, o pipeline chamado `s3-pipeline-test` tem uma ação invocar de pipeline configurada conforme mostrado no console. A ação iniciará o pipeline `s3-pipeline-test` quando o pipeline chamado `my-s3-pipeline` concluir uma execução. O exemplo mostra que a substituição da revisão da origem pela substituição da origem S3_OBJECT_VERSION_ID pelo valor de revisão especificado de `zq8mjNYEexample`.



Input artifacts (Artefatos de entrada)

- Número de artefatos: `0`
- Descrição: os artefatos de entrada não se aplicam a esse tipo de ação.

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Permissões da política de função de serviço para a CodePipeline ação de invocação

Quando CodePipeline executa a ação, a política da função de CodePipeline serviço exige a `codepipeline:StartPipelineExecution` permissão, com escopo adequado até o ARN do recurso do pipeline para manter o acesso com o mínimo de privilégios.

```
{
           "Sid": "StatementForPipelineInvokeAction",
           "Effect": "Allow",
           "Action": "codepipeline:StartPipelineExecution",
           "Resource": [
               "arn:aws:codepipeline:{{region}}:{{AccountId}}:{{pipelineName}}"
           ]
       }
```

Declaração de ação

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\"}]"
    }
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Iniciar um pipeline com uma substituição da revisão de origem— Esta seção descreve como iniciar um pipeline com revisões de fonte manualmente ou por meio do transformador de entrada de EventBridge eventos.

CodeCommit referência de ação de origem

Inicia o pipeline quando uma nova confirmação é feita no CodeCommit repositório e na ramificação configurados.

Se você usar o console para criar ou editar o pipeline, CodePipeline crie uma EventBridge regra que inicia seu pipeline quando ocorre uma alteração no repositório.

> **Note**
>
> Para Amazon ECR, Amazon S3 ou CodeCommit fontes, você também pode criar uma substituição de origem usando a entrada de transformação de entrada para usar o `revisionValue` in EventBridge para seu evento de pipeline, em que `revisionValue` é

derivado da variável de evento de origem para sua chave de objeto, confirmação ou ID de imagem. Para saber mais, consulte a etapa opcional da entrada da transformação de entrada incluída nos procedimentos em Ações e recursos de origem do Amazon ECR EventBridge, Conexão a ações de origem do Amazon S3 com uma fonte habilitada para eventos ou CodeCommit ações de origem e EventBridge.

Você já deve ter criado um CodeCommit repositório antes de conectar o pipeline por meio de uma CodeCommit ação.

Depois que uma alteração de código é detectada, você tem as seguintes opções para passar o código para ações subsequentes:

- Padrão — configura a ação de CodeCommit origem para gerar um arquivo ZIP com uma cópia superficial do seu commit.
- Clone completo: configura a ação de origem para gerar uma referência de URL do Git para o repositório nas ações subsequentes.

  Atualmente, a referência de URL do Git só pode ser usada por CodeBuild ações downstream para clonar o repositório e os metadados do Git associados. A tentativa de passar uma referência de URL do Git para não CodeBuild ações resulta em um erro.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Permissões da função de serviço: CodeCommit ação
- Exemplo de configuração da ação
- Consulte também

Tipo de ação

- Categoria: `Source`

- Proprietário: `AWS`
- Fornecedor: `CodeCommit`
- Versão: `1`

Parâmetros de configuração

RepositoryName

Obrigatório: Sim

O nome do repositório onde as alterações de origem devem ser detectadas.

BranchName

Obrigatório: Sim

O nome da ramificação onde as alterações de origem devem ser detectadas.

PollForSourceChanges

Obrigatório: não

`PollForSourceChanges`controla se CodePipeline pesquisa o CodeCommit repositório em busca de alterações na fonte. Em vez disso, recomendamos que você use CloudWatch Eventos para detectar alterações na fonte. Para obter mais informações sobre como configurar CloudWatch eventos, consulte Migrar canais de votação (CodeCommit fonte) (CLI) ouMigrar canais de votação (fonte) (CodeCommit CloudFormation modelo).

> **Important**
>
> Se você pretende configurar uma regra de CloudWatch eventos, `PollForSourceChanges` defina-a `false` para evitar execuções duplicadas de pipeline.

Os valores válidos para esse parâmetro:

- `true`: Se definido, CodePipeline pesquisa seu repositório em busca de alterações na fonte.

> **Note**
>
> Se você omitir`PollForSourceChanges`, o CodePipeline padrão é pesquisar alterações na fonte em seu repositório. Esse comportamento será o mesmo quando o `PollForSourceChanges` estiver incluído e definido como `true`.

- `false`: Se definido, CodePipeline não pesquisa seu repositório em busca de alterações na fonte. Use essa configuração se você pretende configurar uma regra de CloudWatch eventos para detectar alterações na fonte.

OutputArtifactFormat

Obrigatório: não

Formato do artefato de saída. Os valores podem ser `CODEBUILD_CLONE_REF` ou `CODE_ZIP`. Se não especificado, o padrão será `CODE_ZIP`.

> **Important**
>
> A opção `CODEBUILD_CLONE_REF` só pode ser usada por ações downstream do CodeBuild.
> Se você escolher essa opção, precisará adicionar a `codecommit:GitPull` permissão à sua função de CodeBuild serviço, conforme mostrado emAdicionar CodeBuild GitClone permissões para ações CodeCommit de origem. Você também precisa adicionar a `codecommit:GetRepository` permissão à sua função CodePipeline de serviço, conforme mostrado emAdicionar permissões à função CodePipeline de serviço. Para assistir a um tutorial que mostre como usar a opção Clone completo, consulte Tutorial: use o clone completo com uma fonte de CodeCommit pipeline.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `0`
- Descrição: os artefatos de entrada não se aplicam a esse tipo de ação.

Artefatos de saída

- Número de artefatos: 1

- Descrição: O artefato de saída desta ação é um arquivo ZIP que contém o conteúdo do repositório e ramificação configurados na confirmação especificada como a revisão de origem para a execução do pipeline. Os artefatos gerados a partir do repositório são os artefatos de saída da ação. CodeCommit O ID de confirmação do código-fonte é exibido CodePipeline como a revisão da fonte para a execução do pipeline acionado.

Variáveis de saída

Quando configurada, essa ação produz variáveis que podem ser referenciadas pela configuração de ação de uma ação downstream no pipeline. Esta ação produz variáveis que podem ser visualizadas como variáveis de saída, mesmo que a ação não tenha um namespace. Configure uma ação com um namespace a fim de disponibilizar as variáveis para a configuração de ações downstream.

Para obter mais informações, consulte Referência de variáveis.

CommitId

O ID de CodeCommit confirmação que acionou a execução do pipeline. Os IDs de confirmação são o SHA completo da confirmação.

CommitMessage

A mensagem da descrição, se houver, associada à confirmação que acionou a execução do pipeline.

RepositoryName

O nome do CodeCommit repositório em que a confirmação que acionou o pipeline foi feita.

BranchName

O nome da ramificação do CodeCommit repositório em que a alteração na fonte foi feita.

AuthorDate

A data em que a confirmação foi criada, no formato de timestamp.

CommitterDate

A data em que a confirmação foi confirmada, no formato de timestamp.

Permissões da função de serviço: CodeCommit ação

Quando CodePipeline executa a ação, a política da função de CodePipeline serviço exige as seguintes permissões, com escopo adequado até o ARN do recurso do pipeline para manter o acesso com o mínimo de privilégios. Por exemplo, adicione o seguinte à declaração da 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]]"
            ]
        }
    ]
}
```

Exemplo de configuração da ação

Exemplo de formato de artefato de saída padrão

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"
}
```

Exemplo de formato de artefato de saída de clonagem completa

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"
}
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Tutorial: criar um pipeline simples (CodeCommit repositório)— Este tutorial fornece um AppSpec arquivo de amostra e um exemplo de CodeDeploy aplicativo e grupo de implantação. Use este tutorial para criar um pipeline com uma CodeCommit fonte que é implantada em instâncias do Amazon EC2.

Referência da ação de implantação do AWS CodeDeploy

Você usa uma AWS CodeDeploy ação para implantar o código do aplicativo em sua frota de implantação. Sua frota de implantação pode consistir em instâncias do Amazon EC2, instâncias on-premises ou ambas.

> **Note**
>
> Este tópico de referência descreve a ação CodeDeploy de implantação para CodePipeline onde a plataforma de implantação é o Amazon EC2. Para obter informações de referência sobre o Amazon Elastic Container Service para ações de CodeDeploy blue/green implantação CodePipeline, consulteAmazon Elastic Container Service e referência de ação de implantação CodeDeploy azul-esverdeada.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Permissões da função de serviço: AWS CodeDeploy ação
- Declaração de ação
- Consulte também

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `CodeDeploy`
- Versão: `1`

Parâmetros de configuração

ApplicationName

Obrigatório: Sim

O nome do aplicativo que você criou em CodeDeploy.

DeploymentGroupName

Obrigatório: Sim

O grupo de implantação que você criou em CodeDeploy.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `1`
- Descrição: O AppSpec arquivo CodeDeploy usado para determinar:
  - O que instalar em suas instâncias a partir da revisão do aplicativo no Amazon S3 ou. GitHub
  - quais ganchos de evento de ciclo de vida devem ser executados em resposta a eventos de ciclo de vida de implantação.

  Para obter mais informações sobre o AppSpec arquivo, consulte a Referência CodeDeploy AppSpec do arquivo.

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Permissões da função de serviço: AWS CodeDeploy ação

Para obter AWS CodeDeploy suporte, adicione o seguinte à sua declaração 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": [
```

```
                "*"
            ]
        }
    ]
}
```

Declaração de ação

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"
        }
    ]
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Tutorial: Criar um pipeline simples (bucket do S3)— Este tutorial orienta você na criação de um bucket de origem, instâncias do EC2 e CodeDeploy recursos para implantar um aplicativo de amostra. Em seguida, você cria seu pipeline com uma ação de CodeDeploy implantação que implanta o código mantido em seu bucket do S3 na sua instância do Amazon EC2.
- Tutorial: criar um pipeline simples (CodeCommit repositório)— Este tutorial orienta você na criação do seu repositório de CodeCommit origem, instâncias do EC2 e CodeDeploy recursos para implantar um aplicativo de amostra. Em seguida, você cria seu pipeline com uma ação de CodeDeploy implantação que implanta o código do seu CodeCommit repositório na sua instância do Amazon EC2.
- CodeDeploy AppSpec Referência de arquivo — Este capítulo de referência no Guia AWS CodeDeploy do usuário fornece informações de referência e exemplos de CodeDeploy AppSpec arquivos.

CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com

As ações de origem para conexões são suportadas pelo AWS CodeConnections. AWS CodeConnections permite criar e gerenciar conexões entre AWS recursos e repositórios de terceiros, como GitHub. Aciona um pipeline quando uma nova confirmação é feita em um repositório de código-fonte de terceiros. A ação de origem recupera alterações de código quando um pipeline é executado manualmente ou quando um evento do webhook é enviado pelo provedor de origem.

Você pode configurar ações no pipeline para usar uma configuração do Git que permite iniciar o pipeline com gatilhos. Para definir a configuração de gatilhos do pipeline e aplicar filtros, consulte mais detalhes em Adicionar gatilho com tipos de eventos code push ou pull request.

> **Note**
>
> Esse recurso não está disponível nas regiões Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Melbourne), Ásia-Pacífico (Osaka), África (Cidade do Cabo), Oriente Médio (Bahrein), Oriente Médio (EAU), Europa (Espanha), Europa (Zurique), Israel (Tel Aviv) ou (). AWS GovCloud US-West Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline. Para considerações sobre essa ação na região Europa (Milão), consulte a nota em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

As conexões podem associar seus AWS recursos aos seguintes repositórios de terceiros:

- Bitbucket Cloud (por meio da opção de provedor Bitbucket no CodePipeline console ou do `Bitbucket` provedor na CLI)

  > **Note**
  >
  > Você pode criar conexões para um repositório do Bitbucket Cloud. Não há suporte a tipos de provedores instalados do Bitbucket, como o Bitbucket Server.

- > **Note**
  >
  > Se você estiver usando um espaço de trabalho do Bitbucket, precisará ter acesso de administrador para criar a conexão.

- GitHub e GitHub Enterprise Cloud (por meio da opção de provedor GitHub (via GitHub aplicativo) no CodePipeline console ou do `GitHub` provedor na CLI)

  > **Note**
  >
  > Se o seu repositório estiver em uma GitHub organização, você deverá ser o proprietário da organização para criar a conexão. Se você estiver usando um repositório que não esteja em uma organização, você precisará ser o proprietário do repositório.

- GitHub Enterprise Server (por meio da opção de provedor GitHub Enterprise Server no CodePipeline console ou do `GitHub Enterprise Server` provedor na CLI)
- GitLab.com (por meio da opção de GitLabprovedor no CodePipeline console ou do `GitLab` provedor na CLI)

  > **Note**
  >
  > Você pode criar conexões com um repositório no qual você tem a função de Proprietário e GitLab, em seguida, a conexão pode ser usada com o repositório com recursos como. CodePipeline Para repositórios em grupos, você não precisa ser o proprietário do grupo.

- Self-managed instalação para GitLab (Enterprise Edition ou Community Edition) (por meio da opção de provedor GitLab autogerenciado no CodePipeline console ou do `GitLabSelfManaged` provedor na CLI)

> **Note**
>
> Cada conexão oferece suporte a todos os repositórios que você tem com esse provedor. Você só precisa criar uma nova conexão para cada tipo de provedor.

As conexões permitem que seu pipeline detecte alterações na origem por meio do aplicativo de instalação do provedor de terceiros. Por exemplo, webhooks são usados para assinar tipos de

GitHub eventos e podem ser instalados em uma organização, um repositório ou um aplicativo. GitHub Sua conexão instala um webhook de repositório em seu GitHub aplicativo que se inscreve para GitHub eventos do tipo push.

Depois que uma alteração de código é detectada, você tem as seguintes opções para passar o código para ações subsequentes:

- Padrão: como outras ações de CodePipeline origem existentes, `CodeStarSourceConnection` pode gerar um arquivo ZIP com uma cópia superficial do seu commit.
- Clone completo: a ação `CodeStarSourceConnection` também pode ser configurada para gerar uma referência de URL para o repositório nas ações subsequentes.

  Atualmente, a referência de URL do Git só pode ser usada por CodeBuild ações downstream para clonar o repositório e os metadados do Git associados. A tentativa de passar uma referência de URL do Git para CodeBuild não-ações resulta em um erro.

CodePipeline solicita que você adicione o aplicativo de instalação do AWS Connector à sua conta de terceiros ao criar uma conexão. Você já deve ter criado sua conta e repositório do provedor de terceiros para que possa se conectar por meio da ação `CodeStarSourceConnection`.

> **Note**
>
> Para criar ou anexar uma política ao seu perfil com as permissões necessárias para usar o AWS CodeStar Connections, consulte Referência de permissões do Connections. Dependendo de quando sua função CodePipeline de serviço foi criada, talvez seja necessário atualizar suas permissões para oferecer suporte às AWS CodeStar conexões. Para instruções, consulte Adicionar permissões à função CodePipeline de serviço.

> **Note**
>
> Para usar conexões na Europa (Milão) Região da AWS, você deve:
>
> 1. Instalar um Region-specific aplicativo
> 2. Habilitar a região.

Este Region-specific aplicativo suporta conexões na região da Europa (Milão). Ela é publicada no site do provedor externo e é separada da aplicação existente que permite conexões para outras regiões. Ao instalar essa aplicação, você autoriza provedores externos a compartilhar seus dados somente com o serviço dessa região, mas pode revogar as permissões a qualquer momento ao desinstalá-la.
O serviço não processará nem armazenará seus dados, a menos que você habilite a região. Ao habilitar essa região, você concede ao nosso serviço permissões para processar e armazenar seus dados.
Mesmo que a região não esteja ativada, provedores terceirizados ainda podem compartilhar seus dados com nosso serviço se o Region-specific aplicativo permanecer instalado. Portanto, desinstale o aplicativo depois de desativar a região. Para obter mais informações, consulte Habilitar uma região.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Permissões da função de serviço: AWS CodeConnections ação
- Declaração de ação
- Instalação do aplicativo de instalação e criação de uma conexão
- Consulte também

Tipo de ação

- Categoria: `Source`
- Proprietário: `AWS`
- Fornecedor: `CodeStarSourceConnection`
- Versão: 1

Parâmetros de configuração

ConnectionArn

Obrigatório: Sim

O ARN de conexão configurado e autenticado para o provedor de origem.

FullRepositoryId

Obrigatório: Sim

O proprietário e o nome do repositório onde as alterações de origem devem ser detectadas.

Exemplo: `some-user/my-repo`

> **Important**
>
> Você deve manter as letras maiúsculas e minúsculas corretas para o FullRepositoryIdvalor. Por exemplo, se seu nome de usuário for `some-user` e o nome do repositório for`My-Repo`, o valor recomendado de FullRepositoryIdé`some-user/My-Repo`.

BranchName

Obrigatório: Sim

O nome da ramificação onde as alterações de origem devem ser detectadas.

OutputArtifactFormat

Obrigatório: não

Especifica o formato do artefato de saída. Pode ser `CODEBUILD_CLONE_REF` ou `CODE_ZIP`. Se não especificado, o padrão será `CODE_ZIP`.

> **Important**
>
> A opção `CODEBUILD_CLONE_REF` só pode ser usada por ações downstream do CodeBuild.
> Se você escolher essa opção, precisará atualizar as permissões para sua função de serviço CodeBuild do projeto, conforme mostrado emAdicione CodeBuild GitClone

permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com. Para um tutorial que mostra como usar a opção Clone completo, consulte Tutorial: use o clone completo com uma fonte de GitHub pipeline.

DetectChanges

Obrigatório: não

Determina o acionamento automático do pipeline quando uma nova confirmação é feita no repositório e na ramificação configurados. Se não for especificado, o valor padrão será `true` e o campo não será exibido por padrão. Os valores válidos para esse parâmetro:

- `true`: inicia CodePipeline automaticamente seu funil de novos commits.
- `false`: CodePipeline não inicia seu funil com novos commits.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `0`
- Descrição: os artefatos de entrada não se aplicam a esse tipo de ação.

Artefatos de saída

- Número de artefatos: `1`
- Descrição: os artefatos gerados no repositório são os artefatos de saída para a ação do `CodeStarSourceConnection`. O ID de confirmação do código-fonte é exibido CodePipeline como a revisão da fonte para a execução do pipeline acionado. Você pode configurar o artefato de saída desta ação em:
  - Um arquivo ZIP que contém o conteúdo do repositório e ramificação configurados na confirmação especificada como a revisão de origem para a execução do pipeline.
  - Um arquivo JSON que contém uma referência de URL para o repositório para que as ações downstream possam executar comandos Git diretamente.

> **Important**
>
> Essa opção só pode ser usada por ações CodeBuild posteriores.

Se você escolher essa opção, precisará atualizar as permissões para sua função de serviço CodeBuild do projeto, conforme mostrado emSolução de problemas CodePipeline. Para assistir a um tutorial que mostre como usar a opção Clone completo, consulte Tutorial: use o clone completo com uma fonte de GitHub pipeline.

Variáveis de saída

Quando configurada, essa ação produz variáveis que podem ser referenciadas pela configuração de ação de uma ação downstream no pipeline. Esta ação produz variáveis que podem ser visualizadas como variáveis de saída, mesmo que a ação não tenha um namespace. Configure uma ação com um namespace a fim de disponibilizar as variáveis para a configuração de ações downstream.

Para obter mais informações, consulte Referência de variáveis.

AuthorDate

A data em que a confirmação foi criada, no formato de timestamp.

BranchName

O nome da ramificação do repositório do onde a alteração de origem foi feita.

CommitId

O ID de confirmação do que acionou a execução do pipeline.

CommitMessage

A mensagem da descrição, se houver, associada à confirmação que acionou a execução do pipeline.

ConnectionArn

O ARN de conexão configurado e autenticado para o provedor de origem.

FullRepositoryName

O nome do repositório do onde a confirmação que acionou o pipeline foi feita.

Permissões da função de serviço: AWS CodeConnections ação

Pois AWS CodeConnections, a permissão a seguir é necessária para criar pipelines com uma fonte que usa uma conexão, como o Bitbucket Cloud.

```
{
    "Effect": "Allow",
    "Action": [
        "codeconnections:UseConnection",
        "codestar-connections:UseConnection"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Ambas as ações são necessárias. Operações invocadas por meio de chamada`codestar-connections:UseConnection`, enquanto operações invocadas fora da chamada. `codeconnections:UseConnection` A concessão de ambas as ações garante que a função funcione para os dois caminhos de código, independentemente de o ARN do recurso da conexão usar `codeconnections` o prefixo de serviço `codestar-connections` ou.

Para obter mais informações sobre as permissões do IAM para conexões, consulte Referência de permissões do Connections.

Declaração de ação

No exemplo a seguir, o artefato de saída é definido como o formato ZIP padrão `CODE_ZIP` para a conexão com o 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"
        }
    ]
},
```

Instalação do aplicativo de instalação e criação de uma conexão

Na primeira vez que você usa o console para adicionar uma nova conexão a um repositório de terceiros, você deve autorizar o CodePipeline acesso aos seus repositórios. Você escolhe ou cria um aplicativo de instalação que o ajuda a se conectar à conta em que o repositório de código de terceiros foi criado.

Ao usar o AWS CLI ou um CloudFormation modelo, você deve fornecer o ARN de conexão de uma conexão que já passou pelo handshake de instalação. Caso contrário, o pipeline não é acionado.

> **Note**
>
> Em uma ação de origem `CodeStarSourceConnection`, você não precisa configurar um webhook ou usar a pesquisa como valor padrão. A ação do Connections gerencia a detecção de alterações na origem para você.

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- AWS::CodeStarConnections: :Connection — A referência CloudFormation de modelo para o recurso AWS CodeStar Connections fornece parâmetros e exemplos de conexões em CloudFormation modelos.
- AWS CodeStarReferência da API de AWS CodeStar conexões — A Referência da API de conexões fornece informações de referência para as ações de conexões disponíveis.
- Para visualizar as etapas de criação de um pipeline com as ações de origem compatíveis com o Connections, consulte o seguinte:
  - Para o Bitbucket Cloud, use a opção Bitbucket no console ou a ação `CodestarSourceConnection` na CLI. Consulte Conexões do Bitbucket Cloud.
  - Para GitHub um GitHub Enterprise Cloud, use a opção de GitHubprovedor no console ou a `CodestarSourceConnection` ação na CLI. Consulte GitHub conexões.
  - Para o GitHub Enterprise Server, use a opção GitHub Enterprise Server provider no console ou a `CodestarSourceConnection` ação na CLI. Consulte GitHub Conexões do Enterprise Server.
  - Para GitLab.com, use a opção GitLabprovider no console ou a `CodestarSourceConnection` ação com o `GitLab` provedor na CLI. Consulte GitLab.com conexões.

- Para ver um tutorial de introdução que cria um pipeline com uma fonte do Bitbucket e uma CodeBuild ação, consulte Introdução às conexões.
- Para ver um tutorial que mostra como se conectar a um GitHub repositório e usar a opção de clonagem completa com uma CodeBuild ação posterior, consulte. Tutorial: use o clone completo com uma fonte de GitHub pipeline

Referência da ação Commands

A ação Commands permite que você execute comandos de shell em uma instância de computação virtual. Ao executar a ação, os comandos especificados na configuração dela são executados em um contêiner separado. Todos os artefatos especificados como artefatos de entrada para uma CodeBuild ação estão disponíveis dentro do contêiner que executa os comandos. Essa ação permite que você especifique comandos sem primeiro criar um CodeBuild projeto. Para obter mais informações, consulte ActionDeclaration e OutputArtifact na Referência da API do AWS CodePipeline .

> **Important**
>
> Essa ação usa CodeBuild computação CodePipeline gerenciada para executar comandos em um ambiente de compilação. A execução da ação Comandos incorrerá em cobranças separadas no AWS CodeBuild.

> **Note**
>
> A ação Comandos só está disponível para pipelines do tipo V2.

Tópicos

- Considerações para a ação Commands
- Permissões de política de perfil de serviço
- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída

- Variáveis de ambiente
- Permissões do perfil de serviço: ação Comandos
- Declaração de ação (exemplo)
- Consulte também

Considerações para a ação Commands

As seguintes considerações se aplicam à ação Commands.

- A ação de comandos usa CodeBuild recursos semelhantes à CodeBuild ação, ao mesmo tempo em que permite comandos do ambiente shell em uma instância de computação virtual sem a necessidade de associar ou criar um projeto de construção.

  > **Note**
  >
  > A execução da ação Comandos incorrerá em cobranças separadas no AWS CodeBuild.

- Como a ação Comandos em CodePipeline usa CodeBuild recursos, as compilações executadas pela ação serão atribuídas aos limites de compilação da sua conta em CodeBuild. As compilações executadas pela ação Commands serão contabilizadas nos limites de compilações simultâneas, conforme configurado para essa conta.
- O tempo limite para compilações com a ação Comandos é de 55 minutos, com base nas CodeBuild compilações.
- A instância de computação usa um ambiente de construção isolado em CodeBuild.

  > **Note**
  >
  > Como o ambiente de compilação isolado é usado no nível da conta, uma instância pode ser reutilizada para outra execução de pipeline.

- Todos os formatos são compatíveis, exceto os formatos de várias linhas. Você deve usar o formato de linha única ao inserir comandos.
- A ação de comandos é compatível com ações entre contas. Para adicionar uma ação de comandos entre contas, adicione `actionRoleArn` da conta de destino na declaração de ação.

- Para essa ação, CodePipeline assumirá a função de serviço de pipeline e usará essa função para permitir o acesso aos recursos em tempo de execução. É recomendável configurar o perfil de serviço para que as permissões sejam reduzidas ao nível da ação.
- As permissões adicionadas à função CodePipeline de serviço estão detalhadas emAdicionar permissões à função CodePipeline de serviço.
- A permissão necessária para visualizar logs no console está detalhada em Permissões necessárias para visualizar registros computacionais no console.
- Ao contrário de outras ações em CodePipeline, você não define campos na configuração da ação; você define os campos de configuração da ação fora da configuração da ação.

Permissões de política de perfil de serviço

Quando CodePipeline executa a ação, CodePipeline cria um grupo de registros usando o nome do pipeline da seguinte forma. Isso permite reduzir as permissões para os recursos de log usando o nome do pipeline.

```
/aws/codepipeline/MyPipelineName
```

Caso utilize um perfil de serviço já existente, será preciso incluir as permissões a seguir ao perfil de serviço para habilitar o uso da ação Commands.

- registros: CreateLogGroup
- registros: CreateLogStream
- registros: PutLogEvents

Na declaração de política do perfil de serviço, reduza as permissões até o nível do pipeline, conforme mostrado no exemplo a seguir.

```
{
    "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 visualizar logs no console usando a página de diálogo de detalhes da ação, a permissão para visualizar logs deve ser adicionada ao perfil do console. Para ter mais informações, consulte o exemplo de política de permissões do console em Permissões necessárias para visualizar registros computacionais no console.

Tipo de ação

- Categoria: `Compute`
- Proprietário: `AWS`
- Fornecedor: `Commands`
- Versão: `1`

Parâmetros de configuração

Comandos

Obrigatório: Sim

Você pode fornecer comandos de shell para a ação `Commands` ser executada. No console, os comandos são inseridos em linhas separadas. Na CLI, os comandos são inseridos como strings separadas.

> **Note**
>
> Multi-line os formatos não são suportados e resultarão em uma mensagem de erro. Single-line o formato deve ser usado para inserir comandos no campo Comandos.

> **Important**
>
> Os ComputeType valores de EnvironmentType e correspondem aos de CodeBuild. Oferecemos suporte a um subconjunto dos tipos disponíveis. Para obter mais informações, consulte Tipos de computação de ambiente de compilação.

EnvironmentType

Obrigatório: não

A imagem do sistema operacional para o ambiente de compilação que dá suporte à ação Comandos. Estes são valores válidos para ambientes de compilação:

- LINUX_CONTAINER
- WINDOWS_SERVER_2022_CONTAINER

A seleção para EnvironmentType permitirá então o tipo de computação desse sistema operacional no ComputeType campo. Para obter mais informações sobre os tipos de CodeBuild computação disponíveis para essa ação, consulte a referência de modos e tipos de computação do ambiente de criação no Guia do CodeBuild usuário.

> **Note**
>
> Se não for especificado, a computação assumirá como padrão o seguinte para o ambiente de compilação:
>
> - Tipo de computação: BUILD_GENERAL1_SMALL
> - Tipo de ambiente: LINUX_CONTAINER

ComputeType

Obrigatório: não

Com base na seleção para EnvironmentType, o tipo de computação pode ser fornecido. A seguir estão os valores disponíveis para computação; no entanto, as opções disponíveis podem variar de acordo com o sistema operacional.

- BUILD_GENERAL1_SMALL

- BUILD_GENERAL1_MEDIUM
- BUILD_GENERAL1_LARGE

> **Important**
>
> Alguns tipos de computação não são compatíveis com determinados tipos de ambiente. Por exemplo, WINDOWS_SERVER_2022_CONTAINER não é compatível com BUILD_GENERAL1_SMALL. O uso de combinações incompatíveis causa falha na ação e gera um erro de runtime.

outputVariables

Obrigatório: não

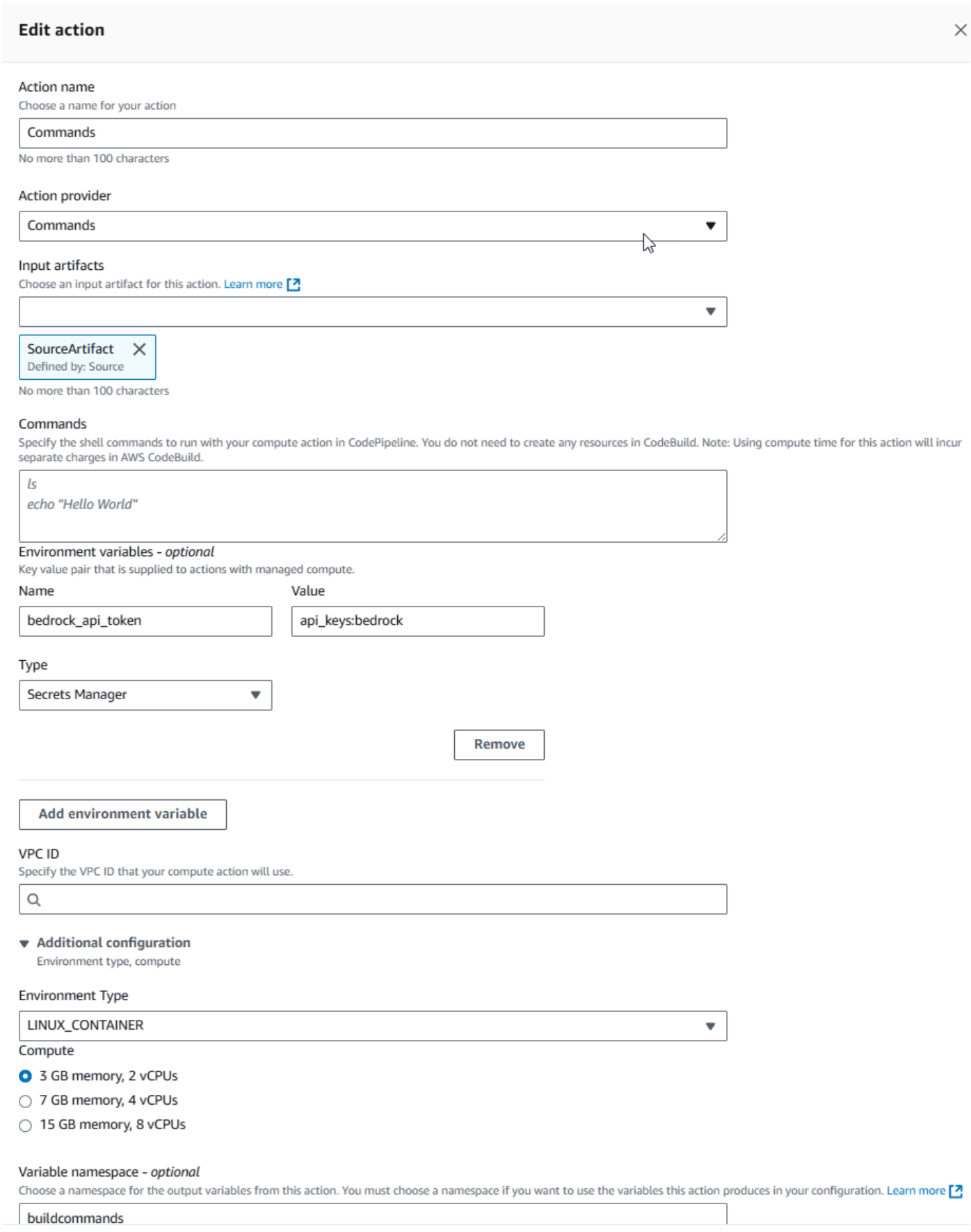
Especifique os nomes das variáveis do ambiente que deseja exportar. Para obter uma referência das variáveis de CodeBuild ambiente, consulte Variáveis de ambiente em ambientes de construção no Guia CodeBuild do usuário.

Arquivos

Obrigatório: não

Você pode fornecer arquivos que deseja exportar como artefatos de saída para a ação.

O formato suportado para arquivos é o mesmo dos padrões de CodeBuild arquivo. Por exemplo, insira `**/` para todos os arquivos. Para obter mais informações, consulte Referência de especificação de compilação CodeBuild no Guia CodeBuild do usuário.


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

Obrigatório: não

O ID da VPC dos recursos.

Sub-redes

Obrigatório: não

As sub-redes da VPC. Este campo é necessário quando os comandos precisam se conectar a recursos em uma VPC.

SecurityGroupIds

Obrigatório: não

Os grupos de segurança da VPC. Este campo é necessário quando os comandos precisam se conectar a recursos em uma VPC.

Veja a seguir um exemplo JSON da ação com campos de configuração mostrados para ambiente e tipo de computação, junto com um exemplo de variável de ambiente.

```
{
           "name": "Commands1",
           "actionTypeId": {
             "category": "Compute",
             "owner": "AWS",
             "provider": "Commands",
             "version": "1"
           },
           "inputArtifacts": [
             {
               "name": "SourceArtifact"
             }
           ],
           "commands": [
             "ls",
             "echo hello",
             "echo $BEDROCK_TOKEN",
           ],
           "configuration": {
             "EnvironmentType": "LINUX_CONTAINER",
```

```
            "ComputeType": "BUILD_GENERAL1_MEDIUM"
          },
          "environmentVariables": [
            {
              "name": "BEDROCK_TOKEN",
              "value": "apiTokens:bedrockToken",
              "type": "SECRETS_MANAGER"
            }
          ],
          "runOrder": 1
        }
```

Input artifacts (Artefatos de entrada)

- Número de artefatos: `1 to 10`

Artefatos de saída

- Número de artefatos: `0 to 1`

Variáveis de ambiente

Chave

A chave em um par de variáveis do ambiente chave-valor, como `BEDROCK_TOKEN`.

Valor

O valor do par chave-valor, como `apiTokens:bedrockToken`. O valor pode ser parametrizado com variáveis de saída a partir de ações ou variáveis do pipeline.

Ao usar o `SECRETS_MANAGER` tipo, esse valor deve ser o nome de um segredo que você já armazenou no AWS Secrets Manager.

Tipo

Especifica o tipo de uso do valor da variável de ambiente. O valor pode ser `PLAINTEXT` ou `SECRETS_MANAGER`. Se o valor for `SECRETS_MANAGER`, informe a referência do Secrets no valor `EnvironmentVariable`. Se não especificado, assumirá como padrão `PLAINTEXT`.

> **Note**
>
> Nós desencorajamos fortemente o uso de variáveis de ambiente de texto simples para armazenar valores confidenciais, especialmente credenciais. AWS Quando você usa o CodeBuild console ou AWS CLI, as variáveis de ambiente de texto sem formatação são exibidas em texto sem formatação. Para valores confidenciais, recomendamos usar o tipo `SECRETS_MANAGER`.

> **Note**
>
> Ao inserir `namevalue`, e `type` para sua configuração de variáveis de ambiente, especialmente se a variável de ambiente contiver sintaxe CodePipeline de variável de saída, não exceda o limite de 1000 caracteres para o campo de valor da configuração. Um erro de validação será retornado quando esse limite for excedido.

Para ver um exemplo de declaração de ação mostrando uma variável de ambiente, consulte Parâmetros de configuração.

> **Note**
>
> - O tipo `SECRETS_MANAGER` é compatível somente com a ação Comandos.
> - Os segredos referenciados na ação Comandos serão editados nos registros de construção de forma semelhante a. CodeBuild Porém, os usuários de pipeline que tiverem acesso Editar ao pipeline ainda poderão acessar esses valores secretos modificando os comandos.
> - Para usar o SecretsManager, você deve adicionar as seguintes permissões à sua função de serviço de pipeline:
>
>   ```
>   {
>               "Effect": "Allow",
>               "Action": [
>                   "secretsmanager:GetSecretValue"
>               ],
>               "Resource": [
>                   "SECRET_ARN"
>               ]
>   ```

```
        }
```

Permissões do perfil de serviço: ação Comandos

Para suporte a Comandos, adicione o seguinte à declaração da 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*"
            ]
        }
    ]
}
```

Declaração de ação (exemplo)

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"
}
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Tutorial: Criar um pipeline que executa comandos com computação (tipo V2): este tutorial fornece um exemplo de pipeline com a ação Commands.

AWS Device Farm referência de ação de teste

Em seu pipeline, você pode configurar uma ação de teste que usa AWS Device Farm para executar e testar seu aplicativo em dispositivos. O Device Farm usa grupos de teste de dispositivos e estruturas de teste para testar aplicações em dispositivos específicos. Para obter informações sobre os tipos de estruturas de teste suportadas pela ação Device Farm, consulte Trabalhando com tipos de teste no AWS Device Farm.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Permissões da função de serviço: AWS Device Farm ação
- Declaração de ação
- Consulte também

Tipo de ação

- Categoria: `Test`

- Proprietário: `AWS`
- Fornecedor: `DeviceFarm`
- Versão: `1`

Parâmetros de configuração

AppType

Obrigatório: Sim

O sistema operacional e o tipo de aplicação que você está testando. Veja a seguir uma lista de valores válidos:

- `iOS`
- `Android`
- `Web`

ProjectId

Obrigatório: Sim

O ID do projeto do Device Farm.

Para encontrar o ID do seu projeto, no console do Device Farm, escolha o projeto. No navegador, copie o URL do novo projeto. O URL contém o ID do projeto. O ID do projeto é o valor na URL após `projects/`. No exemplo a seguir, o ID do projeto é `eec4905f-98f8-40aa-9afc-4c1cfexample`.

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

App

Obrigatório: Sim

O nome e o local do arquivo da aplicação no artefato de entrada. Por exemplo: `s3-ios-test-1.ipa`

TestSpec

Condicional: sim

A localização do arquivo de definição da especificação de teste no artefato de entrada. É necessário para o teste do modo personalizado.

DevicePoolArn

Obrigatório: Sim

O ARN do grupo de dispositivos do Device Farm.

Para obter os ARNs do pool de dispositivos disponíveis para o projeto, incluindo o ARN dos principais dispositivos, use a AWS CLI para inserir o seguinte comando:

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

TestType

Obrigatório: Sim

Especifica a estrutura de teste compatível com seu teste. Veja a seguir uma lista de valores válidos para `TestType`:

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTING
- APPIUM_NODE
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTING
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- BUILTIN_FUZZ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

> **Note**
>
> Os seguintes tipos de teste não são suportados pela ação em CodePipeline: `WEB_PERFORMANCE_PROFILE`e `REMOTE_ACCESS_RECORD`, `REMOTE_ACCESS_REPLAY`.

Para obter informações sobre os tipos de teste do Device Farm, consulte Como trabalhar com tipos de teste no AWS Device Farm.

RadioBluetoothEnabled

Obrigatório: não

Um valor booleano que indica se o Bluetooth deve ser habilitado no início do teste.

RecordAppPerformanceData

Obrigatório: não

Um valor booleano que indica se os dados de desempenho do dispositivo, como CPU, FPS e desempenho da memória, devem ser registrados durante o teste.

RecordVideo

Obrigatório: não

Um valor booleano que indica se o vídeo deve ser gravado durante o teste.

RadioWifiEnabled

Obrigatório: não

Um valor booleano que indica se deve ser ativado Wi-Fi no início do teste.

RadioNfcEnabled

Obrigatório: não

Um valor booleano que indica se o NFC deve ser habilitado no início do teste.

RadioGpsEnabled

Obrigatório: não

Um valor booleano que indica se o NFC deve ser habilitado no início do teste.

Testar

Obrigatório: não

O nome e o caminho do arquivo de definição de teste em seu local de origem. O caminho é relativo à raiz do artefato de entrada para o teste.

FuzzEventCount

Obrigatório: não

O número de eventos de interface do usuário que o teste Fuzz executará, entre 1 e 10.000.

FuzzEventThrottle

Obrigatório: não

O número de milissegundos que o teste Fuzz deve aguardar para realizar o próximo evento de interface do usuário, entre 1 e 1.000.

FuzzRandomizerSeed

Obrigatório: não

Uma propagação que o teste Fuzz usará para randomizar eventos de interface do usuário. O uso do mesmo número de testes Fuzz subsequentes resulta em sequências de eventos idênticas.

CustomHostMachineArtifacts

Obrigatório: não

O local na máquina host em que os artefatos personalizados serão armazenados.

CustomDeviceArtifacts

Obrigatório: não

O local no dispositivo em que os artefatos personalizados serão armazenados.

UnmeteredDevicesOnly

Obrigatório: não

Um valor booleano que indica se os dispositivos de acesso ilimitado devem ser usados somente ao executar testes nesta etapa.

JobTimeoutMinutes

Obrigatório: não

O número de minutos que um teste será executado por dispositivo antes de atingir o tempo limite.

Latitude

Obrigatório: não

A latitude do dispositivo expressa em graus do sistema de coordenadas geográficas.

Longitude

Obrigatório: não

A longitude do dispositivo expressa em graus do sistema de coordenadas geográficas.

Input artifacts (Artefatos de entrada)

- Número de artefatos: 1
- Descrição: o conjunto de artefatos a ser disponibilizado na ação de teste. O Device Farm procura a aplicação de compilação e as definições de teste a serem usadas.

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Permissões da função de serviço: AWS Device Farm ação

Quando CodePipeline executa a ação, a política da função de CodePipeline serviço exige as seguintes permissões, com escopo adequado até o ARN do recurso do pipeline para manter o acesso com o mínimo de privilégios. Por exemplo, adicione o seguinte à declaração da política:

```
{
    "Effect": "Allow",
    "Action": [
        "devicefarm:ListProjects",
```

```
        "devicefarm:ListDevicePools",
        "devicefarm:GetRun",
        "devicefarm:GetUpload",
        "devicefarm:CreateUpload",
        "devicefarm:ScheduleRun"
    ],
    "Resource": "resource_ARN"
},
```

Declaração de ação

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"
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Como trabalhar com tipos de teste no Device Farm: este capítulo de referência no Guia do desenvolvedor do Device Farm fornece mais descrições sobre as estruturas de teste de aplicativos Android, iOS e Web compatíveis com o Device Farm.
- Ações no Device Farm: as chamadas e os parâmetros de API na Referência da API do Device Farm podem ajudar você a trabalhar com projetos do Device Farm.
- Tutorial: crie um pipeline que crie e teste seu aplicativo Android com AWS Device Farm— Este tutorial fornece um exemplo de arquivo buildspec e um aplicativo de amostra para criar um pipeline com uma GitHub fonte que cria e testa um aplicativo Android com CodeBuild o Device Farm.
- Tutorial: crie um pipeline que teste seu aplicativo iOS com AWS Device Farm: este tutorial fornece uma aplicação de exemplo para criar um pipeline com uma origem do Amazon S3 que testa um aplicativo iOS de compilação com o Device Farm.

Referência da ação de implantação do Elastic Beanstalk

O Elastic Beanstalk é uma plataforma interna AWS usada para implantar e escalar aplicativos da web. Você usa uma ação do Elastic Beanstalk para implantar o código do aplicativo no ambiente de implantação.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Permissões do perfil de serviço: ação de implantação do ElasticBeanstalk
- Declaração de ação
- Consulte também

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `ElasticBeanstalk`
- Versão: `1`

Parâmetros de configuração

ApplicationName

Obrigatório: Sim

O nome do aplicativo criado no Elastic Beanstalk.

EnvironmentName

Obrigatório: Sim

O nome do ambiente criado no Elastic Beanstalk. Um ambiente é uma coleção de AWS recursos que executam uma versão do aplicativo. Cada ambiente executa somente uma versão do

aplicativo por vez; no entanto, você pode executar a mesma versão do aplicativo ou versões diferentes do aplicativo em vários ambientes simultaneamente.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `1`
- Descrição: o artefato de entrada da ação.

Artefatos de saída

- Número de artefatos: `0`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Permissões do perfil de serviço: ação de implantação do **`ElasticBeanstalk`**

Para o Elastic Beanstalk, estas são as permissões mínimas necessárias para criar pipelines com uma ação de implantação `ElasticBeanstalk`.

```
{
    "Effect": "Allow",
    "Action": [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
        "cloudformation:*",
        "rds:*",
        "sqs:*",
        "ecs:*"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Você deve substituir os curingas na política de recursos pelos recursos da conta à qual deseja limitar o acesso. Para obter mais informações sobre a criação de uma política que conceda acesso com privilégios mínimos, consulte https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege.

Declaração de ação

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"
        }
    ]
},
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Implantação de um aplicativo Flask no Elastic Beanstalk: este tutorial contém orientações sobre a criação dos recursos de aplicativo e ambiente no Elastic Beanstalk usando um aplicativo Flask de exemplo. Em seguida, compile o pipeline com uma ação de implantação do Elastic Beanstalk que implanta o aplicativo do repositório de origem no ambiente do Elastic Beanstalk.

Referência da ação invocar **`InspectorScan`** do Amazon Inspector

O Amazon Inspector é um serviço de gerenciamento de vulnerabilidades que descobre workloads automaticamente e as verifica continuamente em busca de vulnerabilidades de software e exposições não intencionais da rede. A `InspectorScan` ação CodePipeline automatiza a detecção e a correção de vulnerabilidades de segurança em seu código-fonte aberto. A ação é uma ação de computação gerenciada com recursos de verificação da segurança. Você pode usar InspectorScan com o código-fonte do aplicativo em seu repositório de terceiros, como GitHub o Bitbucket Cloud, ou com imagens para aplicativos de contêiner. Sua ação verificará e relatará os níveis de vulnerabilidade e os alertas configurados por você.

> **Important**
>
> Essa ação usa CodeBuild computação CodePipeline gerenciada para executar comandos em um ambiente de compilação. A execução da ação acarretará cobranças à parte em AWS CodeBuild.

Tópicos

- ID do tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Permissões do perfil de serviço: ação InspectorScan
- Declaração de ação
- Consulte também

ID do tipo de ação

- Categoria: `Invoke`
- Proprietário: `AWS`
- Fornecedor: `InspectorScan`
- Versão: `1`

Exemplo:

```
{
    "Category": "Invoke",
    "Owner": "AWS",
    "Provider": "InspectorScan",
    "Version": "1"
},
```

Parâmetros de configuração

InspectorRunMode

(Obrigatório) A string que indica o modo do escaneamento. O valor válido é `SourceCodeScan | ECRImageScan`.

ECRRepositoryNome

O nome do repositório do Amazon ECR ao qual a imagem foi enviada por push.

ImageTag

A tag usada para a imagem.

Os parâmetros dessa ação verificam os níveis de vulnerabilidade que você especificou. Os seguintes níveis de limites de vulnerabilidade estão disponíveis:

CriticalThreshold

O número de vulnerabilidades de gravidade crítica encontradas em sua fonte, além das quais CodePipeline devem falhar na ação.

HighThreshold

O número de vulnerabilidades de alta gravidade encontradas em sua fonte, além das quais CodePipeline devem falhar na ação.

MediumThreshold

O número de vulnerabilidades de gravidade média encontradas em sua fonte, além das quais CodePipeline devem falhar na ação.

LowThreshold

O número de vulnerabilidades de baixa gravidade encontradas em sua fonte, além das quais CodePipeline devem falhar na ação.

Action name
Choose a name for your action
Scan
No more than 100 characters

Action provider
AWS InspectorScan

Region
US East (N. Virginia)

Input artifacts
Choose an input artifact for this action. Learn more

SourceArtifact
Defined by: Source
No more than 100 characters

Inspector Run mode
InspectorRunMode: SourceCodeScan or ECRImageScan.

Source Code Scan
SourceCodeScan: Scan the source code. Choose an input artifact for this run mode.

ECR Image Scan
ECRImageScan: Scan the ECR image.

Critical Threshold - *optional*
Threshold value for critical severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

High Threshold - *optional*
Threshold value for high severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

Medium Threshold - *optional*
Threshold value for medium severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

Low Threshold - *optional*
Threshold value for low severity vulnerabilities to allow the action to succeed. Default: Integer.MAX_VALUE

Input artifacts (Artefatos de entrada)

- Número de artefatos: 1
- Descrição: o código-fonte para verificar se há vulnerabilidades. Se a verificação for de um repositório ECR, esse artefato de entrada não será necessário.

Artefatos de saída

- Número de artefatos: 1
- Descrição: detalhes da vulnerabilidade da origem na forma de um arquivo da Lista de Materiais de Software (SBOM).

Variáveis de saída

Quando configurada, essa ação produz variáveis que podem ser referenciadas pela configuração de ação de uma ação downstream no pipeline. Esta ação produz variáveis que podem ser visualizadas como variáveis de saída, mesmo que a ação não tenha um namespace. Configure uma ação com um namespace a fim de disponibilizar as variáveis para a configuração de ações downstream.

Para obter mais informações, consulte Referência de variáveis.

HighestScannedSeverity

O resultado de gravidade mais alta da verificação. O valor válido é `medium | high | critical`.

Permissões do perfil de serviço: ação **`InspectorScan`**

Para suporte à ação `InspectorScan`, adicione o seguinte à declaração da política:

```
{
        "Effect": "Allow",
        "Action": "inspector-scan:ScanSbom",
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
```

```
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
        ],
        "Resource": "resource_ARN"
    },
```

Além disso, se ainda não tiver sido adicionada à ação Comandos, adicione as seguintes permissões à sua função de serviço para visualizar CloudWatch os registros.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Restrinja as permissões ao nível de recurso do pipeline aplicando permissões baseadas em recurso na declaração de política do perfil de serviço.

Declaração de ação

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"
                    },
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Para obter mais informações sobre o Amazon Inspector, consulte o Guia do usuário do Amazon Inspector.

AWS Lambda invocar referência de ação

Permite executar uma função do Lambda como uma ação no pipeline. Usando o objeto de evento que é uma entrada para essa função, a função tem acesso à configuração da ação, aos locais dos artefatos de entrada, aos locais dos artefatos de saída e a outras informações necessárias para acessar os artefatos. Para obter um exemplo de evento passado para uma função de invocação do Lambda, consulte Exemplo de evento JSON. Como parte da implementação da função Lambda, deve haver uma chamada para a `PutJobSuccessResult API` ou `PutJobFailureResult API`. Caso contrário, a execução dessa ação trava até que a ação atinja o tempo limite. Se você especificar artefatos de saída para a ação, será necessário fazer upload deles no bucket do S3 como parte da implementação da função.

> **Important**
>
> Não registre o evento JSON CodePipeline enviado ao Lambda, pois isso pode resultar no registro das credenciais do usuário no Logs. CloudWatch A CodePipeline função usa um evento JSON para passar credenciais temporárias para o Lambda no campo. `artifactCredentials` Para obter um evento de exemplo, consulte Exemplo de evento JSON.

Tipo de ação

- Categoria: `Invoke`
- Proprietário: `AWS`
- Fornecedor: `Lambda`
- Versão: `1`

Parâmetros de configuração

FunctionName

Obrigatório: Sim

`FunctionName` é o nome da função criada no Lambda.

UserParameters

Obrigatório: não

Uma string que pode ser processada como entrada pela função do Lambda.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `0 to 5`
- Descrição: o conjunto de artefatos a ser disponibilizado para a função do Lambda.

Artefatos de saída

- Número de artefatos: `0 to 5`
- Descrição: o conjunto de artefatos produzidos como saída pela função do Lambda.

Variáveis de saída

Essa ação produzirá como variáveis todos os pares de valores-chave incluídos na `outputVariables` seção da solicitação da PutJobSuccessResult API.

Para obter mais informações sobre variáveis em CodePipeline, consulteReferência de variáveis.

Exemplo de configuração da ação

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"
        }
    ]
},
```

Exemplo de evento JSON

A ação Lambda envia um evento JSON que contém o ID do trabalho, a configuração da ação do pipeline, os locais dos artefatos de entrada e saída e todas as informações de criptografia dos artefatos. O operador do trabalho acessa esses detalhes para concluir a ação Lambda. Para obter mais informações, consulte detalhes do trabalho. O comando a seguir é um exemplo de evento.

```
{
    "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"
            }
        }
    }
}
```

O evento JSON fornece os seguintes detalhes do trabalho para a ação do Lambda em: CodePipeline

- `id`: o ID exclusivo do trabalho gerado pelo sistema.
- `accountId`: O ID da AWS conta associado ao trabalho.
- `data`: outras informações necessárias para que um operador de trabalho conclua o trabalho.
  - `actionConfiguration`: os parâmetros de ação para a ação Lambda. Para obter definições, consulte Parâmetros de configuração.

- `inputArtifacts`: o artefato fornecido à ação.
  - `location`: o local de armazenamento do artefato.
    - `s3Location`: as informações do local do artefato de entrada para a ação.
      - `bucketName`: o nome do armazenamento de artefatos do pipeline para a ação (por exemplo, um bucket do Amazon S3 chamado codepipeline-us-east-2-1234567890).
      - `objectKey`: o nome do aplicativo (por exemplo, `CodePipelineDemoApplication.zip`).
    - `type`: o tipo de artefato no local. Atualmente, `S3` é o único tipo de artefato válido.
  - `revision`: o ID de revisão do artefato. Dependendo do tipo de objeto, isso pode ser um ID de confirmação (GitHub) ou um ID de revisão (Amazon Simple Storage Service). Para obter mais informações, consulte ArtifactRevision.
  - `name`: o nome do artefato a ser trabalhado, como `MyApp`.
- `outputArtifacts`: a saída da ação.
  - `location`: o local de armazenamento do artefato.
    - `s3Location`: as informações do local do artefato de saída para a ação.
      - `bucketName`: o nome do armazenamento de artefatos do pipeline para a ação (por exemplo, um bucket do Amazon S3 chamado codepipeline-us-east-2-1234567890).
      - `objectKey`: o nome do aplicativo (por exemplo, `CodePipelineDemoApplication.zip`).
    - `type`: o tipo de artefato no local. Atualmente, `S3` é o único tipo de artefato válido.
  - `revision`: o ID de revisão do artefato. Dependendo do tipo de objeto, isso pode ser um ID de confirmação (GitHub) ou um ID de revisão (Amazon Simple Storage Service). Para obter mais informações, consulte ArtifactRevision.
  - `name`: o nome da saída de um artefato, como `MyApp`.
- `artifactCredentials`: as credenciais da AWS sessão usadas para acessar artefatos de entrada e saída no bucket do Amazon S3. Essas credenciais são credenciais temporárias emitidas pelo AWS Security Token Service (AWS STS).
  - `secretAccessKey`: a chave de acesso secreta da sessão.
  - `sessionToken`: o token da sessão.
  - `accessKeyId`: O ID da chave de acesso para a sessão.

- `continuationToken`: um token gerado pela ação. Ações futuras usarão esse token para identificar a instância em execução da ação. Quando a ação for concluída, nenhum token de continuação deverá ser fornecido.
- `encryptionKey`: a chave de criptografia usada para criptografar os dados no armazenamento de artefatos, como uma chave. AWS KMS Se isso não for definido, a chave padrão do Amazon Simple Storage Service será usada.
  - `id`: o ID usado para identificar a chave. Para uma chave do AWS KMS , você pode usar o ID da chave, o ARN da chave ou o ARN do alias.
  - `type`: o tipo de chave de criptografia, como uma chave do AWS KMS .

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- AWS CloudFormation Guia do usuário — Para obter mais informações sobre ações e CloudFormation artefatos do Lambda para pipelines, consulte Usando funções de substituição de parâmetros com CodePipeline pipelines, automatizando a implantação de aplicativos e artefatos. Lambda-based AWS CloudFormation
- Invocar um AWS Lambda função em um pipeline em CodePipeline: este procedimento fornece um exemplo de função do Lambda e mostra como usar o console para criar um pipeline com uma ação de invocação Lambda.

AWS OpsWorks Referência da ação de implantação do

Você usa uma AWS OpsWorks ação para implantar OpsWorks usando seu funil.

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `OpsWorks`
- Versão: `1`

Parâmetros de configuração

App

Obrigatório: Sim

O OpsWorks aplicativo. A aplicação representa o código que você deseja implantar e executar.

Stack

Obrigatório: Sim

A OpsWorks pilha. Pilha é um contêiner para a infraestrutura de aplicação.

Camada

Obrigatório: não

A OpsWorks camada. Uma camada especifica a configuração e os recursos para um conjunto de instâncias.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `1`
- Descrição: este é o artefato de entrada da ação.

Artefatos de saída

- Número de artefatos: `0 to 1`
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Permissões da função de serviço: AWS OpsWorks ação

Para obter AWS OpsWorks suporte, adicione o seguinte à sua declaração de política:

```
{
    "Effect": "Allow",
    "Action": [
        "opsworks:CreateDeployment",
        "opsworks:DescribeApps",
```

```
        "opsworks:DescribeCommands",
        "opsworks:DescribeDeployments",
        "opsworks:DescribeInstances",
        "opsworks:DescribeStacks",
        "opsworks:UpdateApp",
        "opsworks:UpdateStack"
    ],
    "Resource": "resource_ARN"
},
```

Exemplo de configuração da ação

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"
    }
}
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- AWS OpsWorks Guia do usuário — Para obter informações sobre a implantação com AWS OpsWorks, consulte o Guia do AWS OpsWorks usuário.

AWS Service Catalog Referência da ação de implantação do

Você usa uma AWS Service Catalog ação para implantar modelos usando seu funil. Esses são modelos de recursos criados por você no Service Catalog.

Tipo de ação

- Categoria: `Deploy`
- Proprietário: `AWS`
- Fornecedor: `ServiceCatalog`
- Versão: `1`

Parâmetros de configuração

TemplateFilePath

Obrigatório: Sim

O caminho do arquivo do modelo de recurso no local de origem.

ProductVersionName

Obrigatório: Sim

A versão do produto no Service Catalog.

ProductType

Obrigatório: Sim

O tipo de produto no Service Catalog.

ProductId

Obrigatório: Sim

O ID do produto no Service Catalog.

ProductVersionDescription

Obrigatório: não

A descrição da versão do produto no Service Catalog.

Input artifacts (Artefatos de entrada)

- Número de artefatos: 1
- Descrição: este é o artefato de entrada da ação.

Artefatos de saída

- Número de artefatos: 0
- Descrição: os artefatos de saída não se aplicam a esse tipo de ação.

Permissões do perfil de serviço: ação do Service Catalog

Para compatibilidade com o Service Catalog, adicione o seguinte à declaração da 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"
}
```

Configurações de ação de exemplo por tipo do arquivo de configuração

O exemplo a seguir mostra uma configuração válida para uma ação de implantação que usa o Service Catalog, para um pipeline criado no console sem um arquivo de configuração separado:

```
"configuration": {
  "TemplateFilePath": "S3_template.json",
  "ProductVersionName": "devops S3 v2",
  "ProductType": "CLOUD_FORMATION_TEMPLATE",
  "ProductVersionDescription": "Product version description",
  "ProductId": "prod-example123456"
}
```

O exemplo a seguir mostra uma configuração válida para uma ação de implantação que usa o Service Catalog, para um pipeline criado no console com um arquivo de configuração `sample_config.json` separado:

```
"configuration": {
  "ConfigurationFilePath": "sample_config.json",
  "ProductId": "prod-example123456"
}
```

Exemplo de configuração da ação

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"
    }
}
```

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Guia do usuário do Service Catalog: para saber mais sobre recursos e modelos no Service Catalog, consulte o Guia do usuário do Service Catalog.
- Tutorial: Criar um pipeline que realiza a implantação no Service Catalog— Este tutorial mostra como criar e configurar um pipeline para implantar seu modelo de produto no Service Catalog e entregar as alterações que você fez no seu repositório de origem.

AWS Step Functions invocar referência de ação

Uma AWS CodePipeline ação que faz o seguinte:

- Inicia a execução de uma máquina de AWS Step Functions estado a partir do seu pipeline.
- Fornece um estado inicial para a máquina de estado por meio de uma propriedade na configuração de ação ou de um arquivo localizado em um artefato de pipeline a ser transmitido como entrada.
- Opcionalmente, define um prefixo de ID de execução para identificar execuções originadas da ação.
- É compatível com máquinas de estado padrão e expressa.

> **Note**
>
> A ação do Step Functions é executada no Lambda e, por isso, as cotas de tamanho de artefato são iguais às cotas de tamanho de artefato para funções do Lambda. Para ter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do Lambda.

Tipo de ação

- Categoria: `Invoke`
- Proprietário: `AWS`
- Fornecedor: `StepFunctions`
- Versão: 1

Parâmetros de configuração

StateMachineArn

Obrigatório: Sim

O nome de recurso da Amazon (ARN) para a máquina de estado a ser invocada.

ExecutionNamePrefix

Obrigatório: não

Por padrão, o ID de execução da ação é usado como o nome de execução da máquina de estado. Se um prefixo for fornecido, ele será adicionado ao ID de execução da ação com um hífen e será usado como o nome de execução da máquina de estado.

```
myPrefix-1624a1d1-3699-43f0-8e1e-6bafd7fde791
```

Para uma máquina de estado expressa, o nome deve conter apenas 0-9, A-Z, a-z, - e _.

InputType

Obrigatório: não

- Literal (padrão): quando especificado, o valor no campo Input é transmitido diretamente para a entrada da máquina de estado.

  Entrada de exemplo para o campo Input quando Literal é selecionado:

  ```
  {"action": "test"}
  ```

- FilePath: o conteúdo de um arquivo no artefato de entrada especificado pelo campo Entrada é usado como entrada para a execução da máquina de estado. Um artefato de entrada é necessário quando InputTypedefinido como. FilePath

  Exemplo de entrada para o campo FilePathEntrada quando selecionado:

  ```
  assets/input.json
  ```

Input

Obrigatório: condicional

- Literal: quando InputTypedefinido como Literal (padrão), esse campo é opcional.

  Se fornecido, o campo Input será usado diretamente como a entrada para a execução da máquina de estado. Caso contrário, a máquina de estado será invocada com um objeto JSON `{}` vazio.

- FilePath: Quando InputTypedefinido como FilePath, esse campo é obrigatório.

  Um artefato de entrada também é necessário quando InputTypedefinido como. FilePath

  O conteúdo do arquivo no artefato de entrada especificado é usado como entrada para a execução da máquina de estado.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `0 to 1`
- Descrição: Se InputTypeestiver definido como FilePath, esse artefato é necessário e é usado para fornecer a entrada para a execução da máquina de estado.

Artefatos de saída

- Número de artefatos: `0 to 1`
- Descrição:
  - Máquinas de estado padrão: se fornecido, o artefato de saída será preenchido com a saída da máquina de estado. Isso é obtido da `output` propriedade da resposta da DescribeExecution API Step Functions após a conclusão bem-sucedida da execução da máquina de estado.
  - Máquinas de estado expressas: não são compatíveis.

Variáveis de saída

Essa ação produz variáveis de saída que podem ser referenciadas pela configuração de uma ação downstream no pipeline.

Para obter mais informações, consulte Referência de variáveis.

StateMachineArn

O ARN da máquina de estado.

ExecutionArn

O ARN da execução da máquina de estado. Somente máquinas de estado padrão.

Permissões do perfil de serviço: ação **`StepFunctions`**

Para a ação `StepFunctions`, estas são as permissões mínimas necessárias para criar pipelines com uma ação de invocação do Step Functions.

```
{
    "Effect": "Allow",
    "Action": [
```

```
        "states:DescribeStateMachine",
        "states:DescribeExecution",
        "states:StartExecution"
    ],
    "Resource": "resource_ARN"
},
```

Exemplo de configuração da ação

Exemplo de entrada padrão

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"
    }
}
```

Exemplo de 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\"}"
    }
}
```

Exemplo de arquivo 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"
    }
}
```

Comportamento

Durante uma versão, CodePipeline executa a máquina de estado configurada usando a entrada conforme especificado na configuração da ação.

Quando InputTypedefinido como Literal, o conteúdo do campo Configuração da ação de entrada é usado como entrada para a máquina de estado. Quando a entrada literal não é fornecida, a execução da máquina de estado usa um objeto JSON `{}` vazio. Para obter mais informações sobre como executar uma execução de máquina de estado sem entrada, consulte a StartExecutionAPI Step Functions.

Quando InputTypeestá definida como FilePath, a ação descompacta o artefato de entrada e usa o conteúdo do arquivo especificado no campo Configuração da ação de entrada como entrada para a máquina de estado. Quando FilePathespecificado, o campo Entrada é obrigatório e um artefato de entrada deve existir; caso contrário, a ação falhará.

Após uma execução de início bem-sucedida, o comportamento divergirá para os dois tipos de máquina de estado, padrão e expressa.

Máquinas de estado padrão

Se a execução da máquina de estado padrão foi iniciada com sucesso, CodePipeline pesquisa a `DescribeExecution` API até que a execução atinja o status de terminal. Se a execução for concluída com êxito, a ação será bem-sucedida; caso contrário, ela falhará.

Se um artefato de saída for configurado, o artefato conterá o valor de retorno da máquina de estado. Isso é obtido da `output` propriedade da resposta da DescribeExecution API Step Functions após

a conclusão bem-sucedida da execução da máquina de estado. Observe que há restrições de comprimento de saída impostas nesta API.

Tratamento de erros

- Se ocorrer falha na ação ao iniciar uma execução de máquina de estado, a execução da ação falhará.
- Se a execução da máquina de estado não atingir o status do terminal antes que a ação CodePipeline Step Functions atinja seu tempo limite (padrão de 7 dias), a execução da ação falhará. A máquina de estado poderá continuar apesar dessa falha. Para obter mais informações sobre o tempo limite da execução da máquina de estado no Step Functions, consulte Comparação entre os fluxos de trabalho Standard e Express.

  > **Note**
  >
  > É possível solicitar um aumento da cota do tempo limite da ação de invocação para a conta com a ação. No entanto, o aumento da cota aplica-se a todas as ações deste tipo em todas as regiões para essa conta.

- Se a execução da máquina de estado atingir um status de terminal de FAILED, TIMED_OUT ou ABORTED, ocorrerá falha na execução da ação.

Máquinas de estado expressas

Se a execução da máquina de estado expressa foi iniciada com êxito, a execução da ação de invocação será concluída com êxito.

Considerações sobre ações configuradas para máquinas de estado expressa:

- Não é possível designar um artefato de saída.
- A ação não aguarda o término da execução da máquina de estado.
- Depois que a execução da ação é iniciada em CodePipeline, a execução da ação é bem-sucedida mesmo se a execução da máquina de estado falhar.

Tratamento de erros

- Se CodePipeline falhar ao iniciar a execução de uma máquina de estado, a execução da ação falhará. Caso contrário, a ação será executada com êxito imediatamente. A ação é bem-sucedida

CodePipeline independentemente de quanto tempo a execução da máquina de estado leva para ser concluída ou de seu resultado.

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- AWS Step Functions Guia do desenvolvedor — Para obter informações sobre máquinas de estado, execuções e entradas para máquinas de estado, consulte o Guia do AWS Step Functions desenvolvedor.
- Tutorial: use uma ação de AWS Step Functions invocação em um pipeline: este tutorial apresenta o uso de uma máquina de estado padrão de exemplo e mostra como usar o console para atualizar um pipeline adicionando uma ação de invocação do Step Functions.

Referência da estrutura de regras

Esta seção é uma referência somente para a configuração de regras. Para obter uma visão geral conceitual da estrutura do pipeline, consulte CodePipeline referência de estrutura de tubulação.

Cada provedor de regras no CodePipeline usa um conjunto de campos de configuração obrigatórios e opcionais na estrutura do pipeline. Esta seção fornece as seguintes informações de referência por provedor de regras:

- Valores válidos para os campos `RuleType` incluídos no bloco de regras da estrutura do pipeline, como `Owner` e `Provider`.
- Descrições e outras informações de referência para os parâmetros de `Configuration` (obrigatórios e opcionais) incluídos na seção de regras da estrutura de pipeline.
- Exemplos válidos de campos de configuração de regras JSON e YAML.

As informações de referência estão disponíveis para os seguintes provedores de regras:

Tópicos

- CloudWatchAlarm
- Regra do CodeBuild
- Comandos
- DeploymentWindow
- LambdaInvoke
- VariableCheck

CloudWatchAlarm

Ao criar uma condição, você pode adicionar a regra `CloudWatchAlarm`. Esta seção fornece uma referência para os parâmetros da regra. Para ter mais informações sobre regras e condições, consulte Como funcionam as condições do estágio?.

Você já deve ter criado um alarme na Amazon CloudWatch como um recurso separado.

Tópicos

- Tipo de regra

- Parâmetros de configuração
- Exemplo de configuração de regra
- Consulte também

Tipo de regra

- Categoria: `Rule`
- Proprietário: `AWS`
- Fornecedor: `CloudWatchAlarm`
- Versão: `1`

Parâmetros de configuração

AlarmName

Obrigatório: Sim

O nome do CloudWatch alarme. Esse é um recurso separado criado em CloudWatch.

AlarmStates

Obrigatório: não

Os estados CloudWatch de alarme desejados para a regra monitorar. Valores válidos: ALARM, OK ou INSUFFICIENT_DATA.

WaitTime

Obrigatório: não

O tempo de espera em minutos para que ocorra uma alteração de estado antes de executar o resultado da regra. Por exemplo, configure 20 minutos para esperar que um estado de alarme mude para OK antes de aplicar o resultado da regra.

Exemplo de configuração de regra

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"
                        }
                    ]
                }
```

Consulte também

Os recursos relacionados a seguir podem ajudar você a trabalhar com essa regra.

- Criar condições “em caso de falha”: esta seção fornece etapas para criar uma condição “em caso de falha” com uma regra de alarme.

Regra do CodeBuild

Ao criar uma condição, você pode adicionar a regra do CodeBuild. Esta seção fornece uma referência para os parâmetros da regra. Para ter mais informações sobre regras e condições, consulte Como funcionam as condições do estágio?.

Você pode usar a regra do CodeBuild para criar uma condição na qual a execução bem-sucedida do projeto de compilação atenda aos critérios da regra, como a execução bem-sucedida da compilação para uma condição beforeEntry.

> **Note**
>
> Para condições beforeEntry configuradas com o resultado Ignorar, somente as seguintes regras estão disponíveis: `LambdaInvoke` e `VariableCheck`.

Tópicos

- Permissões de política de perfil de serviço
- Tipo de regra
- Parâmetros de configuração
- Exemplo de configuração de regra
- Consulte também

Permissões de política de perfil de serviço

Para permissões para essa regra, adicione o seguinte à declaração da política do perfil de serviço do CodePipeline. Reduza o escopo das permissões até o nível do recurso.

```
{
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
    ],
    "Resource": "resource_ARN"
},
```

Tipo de regra

- Categoria: `Rule`
- Proprietário: `AWS`
- Fornecedor: `CodeBuild`
- Versão: `1`

Parâmetros de configuração

ProjectName

Obrigatório: Sim

`ProjectName` é o nome do projeto de compilação no CodeBuild.

PrimarySource

Obrigatório: condicional

O valor do parâmetro `PrimarySource` deve ser o nome de um dos artefatos de entrada para a ação. O CodeBuild procura o arquivo buildspec e executa os comandos buildspec no diretório que contém a versão descompactada desse artefato.

Esse parâmetro será necessário se vários artefatos de entrada forem especificados para uma ação do CodeBuild. Quando houver apenas um artefato de origem para a ação, o artefato `PrimarySource` assumirá esse artefato como padrão.

BatchEnabled

Obrigatório: Não

O valor booleano do parâmetro `BatchEnabled` permite que a ação execute várias compilações na mesma execução de compilação.

Quando esta opção está habilitada, a opção `CombineArtifacts` está disponível.

Para exemplos de pipeline com compilações em lote habilitadas, consulte Integração do CodePipeline ao CodeBuild e compilações em lote.

CombineArtifacts

Obrigatório: Não

O valor booleano do parâmetro `CombineArtifacts` combina todos os artefatos de uma compilação em lote em um único arquivo de artefato para a ação de compilação.

Para usar esta opção, o parâmetro `BatchEnabled` deve estar ativado.

EnvironmentVariables

Obrigatório: Não

O valor deste parâmetro é usado para definir variáveis de ambiente para a ação do CodeBuild no pipeline. O valor do parâmetro `EnvironmentVariables` assume a forma de uma matriz JSON de objetos de variáveis de ambiente. Consulte o parâmetro de exemplo em Declaração de ação (CodeBuild exemplo).

Cada objeto tem três partes, todas são strings:

- `name`: o nome ou a chave da variável de ambiente.
- `value`: o valor da variável de ambiente. Ao usar o tipo `PARAMETER_STORE` ou `SECRETS_MANAGER`, este valor deve ser o nome de um parâmetro já armazenado no AWS Systems Manager Parameter Store ou um segredo já armazenado no AWS Secrets Manager, respectivamente.

> **Note**
>
> Não recomendamos o uso de variáveis de ambiente para armazenar valores confidenciais, principalmente credenciais da AWS. Quando você usa o console do CodeBuild ou a AWS CLI, as variáveis de ambiente são exibidas em texto sem formatação. Para valores confidenciais, recomendamos usar o tipo `SECRETS_MANAGER`.

- `type`: (opcional) o tipo da variável de ambiente. Os valores válidos são `PARAMETER_STORE`, `SECRETS_MANAGER` ou `PLAINTEXT`. Se não especificado, assumirá como padrão `PLAINTEXT`.

> **Note**
>
> Ao inserir `name`, `value` e `type` na configuração das variáveis de ambiente, principalmente se a variável de ambiente contiver a sintaxe da variável de saída do CodePipeline, não ultrapasse o limite de 1000 caracteres no campo de valor da configuração. Um erro de validação será retornado quando esse limite for excedido.

Para obter mais informações, consulte EnvironmentVariable na Referência da API do AWS CodeBuild. Para ver um exemplo de ação do CodeBuild com uma variável de ambiente que retorna o nome de ramificação do GitHub, consulte Exemplo: use uma BranchName variável com variáveis de CodeBuild ambiente.

Exemplo de configuração de regra

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"
}
```

Consulte também

Os recursos relacionados a seguir podem ajudar você a trabalhar com essa regra.

- Para ter mais informações sobre regras e condições, consulte Condition, RuleTypeId e RuleExecution no Guia de API do CodePipeline.

Comandos

Ao criar uma condição, você pode adicionar a regra `Commands`. Esta seção fornece uma referência para os parâmetros da regra. Para ter mais informações sobre regras e condições, consulte Como funcionam as condições do estágio?.

Você pode usar a regra `Commands` para criar uma condição na qual os comandos bem-sucedidos atendam aos critérios da regra, como a saída e o caminho do arquivo para os comandos serem bem-sucedidos em uma condição beforeEntry.

> **Note**
>
> Em relação a condições beforeEntry configuradas com o resultado Ignorar, somente as seguintes regras estão disponíveis: `LambdaInvoke` e `VariableCheck`.

Tópicos

- Considerações sobre a regra Comands
- Permissões de política de perfil de serviço
- Tipo de regra
- Parâmetros de configuração
- Exemplo de configuração de regra
- Consulte também

Considerações sobre a regra Comands

As considerações a seguir se aplicam à regra Commands.

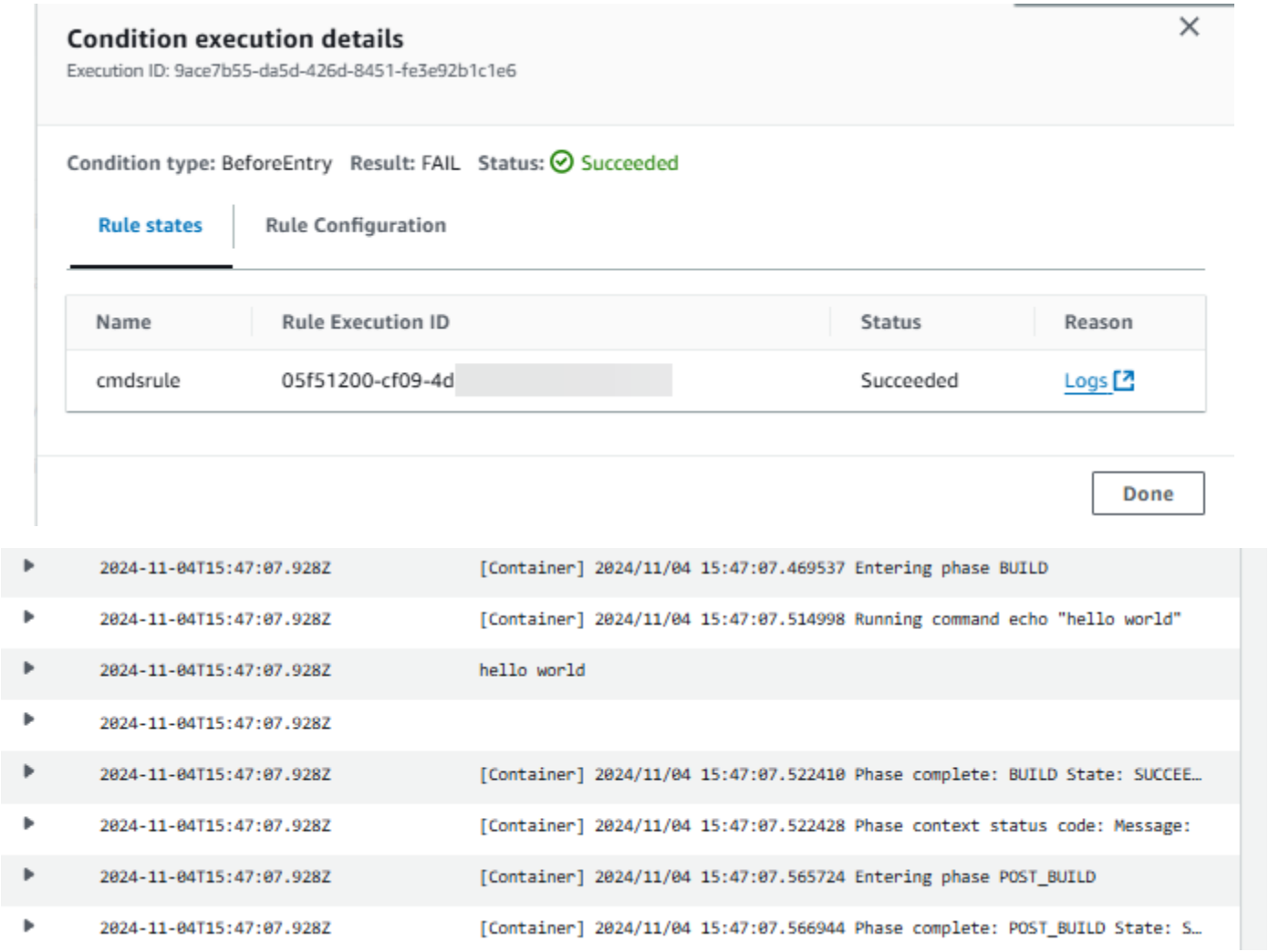
- A regra de comandos usa CodeBuild recursos semelhantes à CodeBuild ação, ao mesmo tempo em que permite comandos do ambiente shell em uma instância de computação virtual sem a necessidade de associar ou criar um projeto de construção.

  > **Note**
  >
  > A execução da regra de comandos acarretará cobranças separadas em AWS CodeBuild.

- Como a regra de Comandos CodePipeline usa CodeBuild recursos, as compilações executadas pela ação serão atribuídas aos limites de criação da sua conta em CodeBuild. As compilações executadas pela regra Commands serão contabilizadas para os limites de compilações simultâneas, conforme configurado para essa conta.
- O tempo limite para compilações com a regra de Comandos é de 55 minutos, com base nas CodeBuild compilações.
- A instância de computação usa um ambiente de compilação isolado em CodeBuild.

  > **Note**
  >
  > Como o ambiente de compilação isolado é usado no nível da conta, uma instância pode ser reutilizada para outra execução de pipeline.

- Todos os formatos são compatíveis, exceto os formatos de várias linhas. Você deve usar o formato de linha única ao inserir comandos.
- Para essa regra, CodePipeline assumirá a função de serviço de pipeline e usará essa função para permitir o acesso aos recursos em tempo de execução. É recomendável configurar o perfil de serviço para que as permissões sejam reduzidas ao nível da ação.
- As permissões adicionadas à função CodePipeline de serviço estão detalhadas emAdicionar permissões à função CodePipeline de serviço.
- A permissão necessária para visualizar logs no console está detalhada em Permissões necessárias para visualizar registros computacionais no console. Nas telas de exemplo a seguir, use o link Registros para ver os registros de uma regra de comandos bem-sucedida nos CloudWatch registros.



- Ao contrário de outras ações em CodePipeline, você não define campos na configuração da ação; você define os campos de configuração da ação fora da configuração da ação.

Permissões de política de perfil de serviço

Ao CodePipeline executar a regra, CodePipeline cria um grupo de registros usando o nome do pipeline da seguinte forma. Isso permite reduzir as permissões para os recursos de log usando o nome do pipeline.

```
/aws/codepipeline/MyPipelineName
```

Caso utilize um perfil de serviço já existente, será preciso incluir as permissões a seguir ao perfil de serviço para habilitar o uso da ação Commands.

- registros: CreateLogGroup

- registros: CreateLogStream
- registros: PutLogEvents

Na declaração de política do perfil de serviço, reduza as permissões até o nível do pipeline, conforme mostrado no exemplo a seguir.

```
{
    "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 visualizar logs no console usando a página de diálogo de detalhes da ação, a permissão para visualizar logs deve ser adicionada ao perfil do console. Para ter mais informações, consulte o exemplo de política de permissões do console em Permissões necessárias para visualizar registros computacionais no console.

Tipo de regra

- Categoria: `Rule`
- Proprietário: `AWS`
- Fornecedor: `Commands`
- Versão: `1`

Parâmetros de configuração

Comandos

Obrigatório: Sim

Você pode fornecer comandos de shell para a regra `Commands` ser executada. No console, os comandos são inseridos em linhas separadas. Na CLI, os comandos são inseridos como strings separadas.

> **Note**
>
> Os formatos de várias linhas não são compatíveis e resultarão em uma mensagem de erro. O formato de linha única deve ser usado para inserir comandos no campo Comandos.

Os detalhes a seguir oferecem a computação padrão usada para a regra Commands. Para obter mais informações, consulte a referência de modos e tipos de computação do ambiente de criação no Guia do CodeBuild usuário.

- CodeBuild imagem: aws/codebuild/amazonlinux 2-x86_64 standard: 5.0
- Tipo de computação: Linux Small
- Valor do ComputeType do ambiente: BUILD_ _SMALL GENERAL1
- Valor do tipo de ambiente: LINUX_CONTAINER

Exemplo de configuração de regra

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"
        }
    ]
}
```

Consulte também

Os recursos relacionados a seguir podem ajudar você a trabalhar com essa regra.

- Para obter mais informações sobre regras e condições, consulte Condição e RuleExecutionno Guia da CodePipeline API. RuleTypeId

DeploymentWindow

Ao criar uma condição, você pode adicionar a regra `DeploymentWindow`. Esta seção fornece uma referência para os parâmetros da regra. Para ter mais informações sobre regras e condições, consulte Como funcionam as condições do estágio?.

Tópicos

- Tipo de regra

- Parâmetros de configuração
- Exemplo de configuração de regra
- Consulte também

Tipo de regra

- Categoria: `Rule`
- Proprietário: `AWS`
- Fornecedor: `DeploymentWindow`
- Versão: `1`

Parâmetros de configuração

Cron

Obrigatório: Sim

A expressão que define os dias e horários em que a implantação será permitida. As expressões cron são compostas por 6 campos obrigatórios e um campo opcional separados por espaço em branco. Os campos de expressão cron permitem que você especifique um padrão de programação com uma expressão cron da seguinte forma.

Nome do campo	Valores permitidos	Caracteres especiais permitidos
Segundos	N/A	*
minutos	0-59	, - * /
Horas	0-23	, - * /
Day-of-month	1-31	, - * ? / L W
Mês	1-12 ou JAN-DEC	, - * /
Day-of-Week	1-7 ou SUN-SAT	, - * ? / L #

Nome do campo	Valores permitidos	Caracteres especiais permitidos
Ano (opcional)	vazio, 1970–2199	, - * /

- O caractere “*” é usado para especificar todos os valores. Por exemplo, “*” no campo de minutos significa “a cada minuto”.
- O caractere “?” é permitido para os campos de dia do mês e dia da semana. É usado para especificar “nenhum valor específico”. Isso é útil quando você precisa especificar algo em um dos dois campos, mas não no outro.
- O caractere '-' é usado para especificar intervalos. Por exemplo, “10-12" no campo de horas significa “as horas 10, 11 e 12".
- O caractere “,” é usado para especificar valores adicionais. Por exemplo, “MON,WED,FRI” no campo do dia da semana significa “os dias segunda-feira, quarta-feira e sexta-feira”.
- O caractere “/” é usado para especificar incrementos. Por exemplo, 0/15 "" no campo de segundos significa “os segundos 0, 15, 30 e 45". E "5/15" no campo de segundos significa “os segundos 5, 20, 35 e 50". Especificar “*” antes de “/” é equivalente a especificar 0 como valor inicial.
- O caractere “L” é permitido nos campos de dia do mês e dia da semana. Esse caractere é a abreviatura de “último”, mas tem um significado diferente em cada um dos dois campos. Por exemplo, o valor “L” no campo dia do mês significa “o último dia do mês”: dia 31 para janeiro, dia 28 para fevereiro em anos não bissextos. Se usado no campo de dia da semana por si só, significa “7” ou “SAT”. Mas se usado no campo dia da semana após outro valor, significa “o último dia <dia_especificado> do mês”. Por exemplo, “6L” significa “a última sexta-feira do mês”. Você também pode especificar uma diferença em relação ao último dia do mês, como "L-3", que significaria o penúltimo dia do mês civil.
- O caractere “W” é permitido para o campo dia do mês. Esse caractere é usado para especificar o dia da semana (Monday-Friday) mais próximo do dia determinado. Por exemplo, se você especificasse “15W” como o valor para o campo dia do mês, o significado seria: “o dia da semana mais próximo do dia 15 do mês”. Então, se o dia 15 for um sábado, o gatilho será acionado na sexta-feira, dia 14. Se o dia 15 for um domingo, o gatilho disparará na segunda-feira, dia 16. Se o dia 15 for uma terça-feira, disparará na terça-feira, dia 15.
- Os caracteres “L” e “W” também podem ser combinados para que a expressão do dia do mês produza “LW”, que se traduz em “último dia útil do mês”.

- O caractere “#” é permitido no campo de dia da semana. Este caractere é usado para especificar “o enésimo” <dia_especificado> dia do mês. Por exemplo, o valor de “6#3” no campo dia da semana significa a terceira sexta-feira do mês (dia 6 = sexta-feira e “#3” = a terceira do mês).
- Os caracteres válidos e os nomes dos meses e dias da semana não diferenciam maiúsculas de minúsculas.

TimeZone

Obrigatório: não

O fuso horário da janela de implantação. A expressão regular corresponde aos padrões nos seguintes formatos:

- Region/City formato. O valor corresponde a um fuso horário no formato Region/City Region/City _City. Por exemplo, `America/New_York` ou `Europe/Berlin`.
- Formato UTC. O valor corresponde à string UTC, seguida opcionalmente por um deslocamento no formato + HH:MM ou -. HH:MM Por exemplo, `UTC`, `UTC+05:30` ou `UTC-03:00`. Esse será o formato padrão se o parâmetro não for definido de outra forma.
- Formato de abreviação. O valor corresponde a uma abreviação de 3 a 5 caracteres para um fuso horário. Por exemplo, `EST` ou `IST`.

  Para obter uma tabela de valores de TimeZone ID válidos, consulte https://docs.oracle.com/middleware/1221/wcs/tag-ref/MISC/TimeZones.html. Observe que certas abreviações são duplicadas, como CST para Horário Padrão Central, Horário Padrão da China e Horário Padrão de Cuba.

Exemplo de configuração de regra

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"
                    }
                ]
```

Consulte também

Os recursos relacionados a seguir podem ajudar você a trabalhar com essa regra.

- Criar condições “em caso de sucesso”: esta seção fornece etapas para criar uma condição “em caso de sucesso” com uma regra de janela de implantação.
- Para obter mais informações sobre regras e condições, consulte Condição RuleTypeId, e RuleExecution no Guia da CodePipeline API.

LambdaInvoke

Ao criar uma condição, você pode adicionar a regra `LambdaInvoke`. Esta seção fornece uma referência para os parâmetros da regra. Para ter mais informações sobre regras e condições, consulte Como funcionam as condições do estágio?.

Você já deve ter criado uma função no Lambda como um recurso separado.

Tópicos

- Tipo de regra
- Parâmetros de configuração
- Exemplo de configuração de regra
- Consulte também

Tipo de regra

- Categoria: `Rule`
- Proprietário: `AWS`
- Fornecedor: `LambdaInvoke`
- Versão: `1`

Parâmetros de configuração

FunctionName

Obrigatório: Sim

O nome da função do Lambda.

UserParameters

Obrigatório: Não

Esses são parâmetros fornecidos como entrada para a função no formato de par de chave/valor.

Exemplo de configuração de regra

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"
    }
]
```

Consulte também

Os recursos relacionados a seguir podem ajudar você a trabalhar com essa regra.

- Criar condições “em caso de falha”: esta seção fornece etapas para criar uma condição “em caso de falha” com uma regra de alarme.

VariableCheck

Ao criar uma condição, você pode adicionar a regra `VariableCheck`. Esta seção fornece uma referência para os parâmetros da regra. Para ter mais informações sobre regras e condições, consulte Como funcionam as condições do estágio?.

A regra `VariableCheck` pode ser utilizada para definir uma condição que verifica a variável de saída em relação a uma expressão especificada. A regra passa na verificação quando o valor da variável atende aos critérios da regra, como o valor ser igual ou maior que uma variável de saída especificada.

Tópicos

- Tipo de regra
- Parâmetros de configuração
- Exemplo de configuração de regra
- Consulte também

Tipo de regra

- Categoria: `Rule`
- Proprietário: `AWS`
- Fornecedor: `VariableCheck`
- Versão: `1`

Parâmetros de configuração

Operador

Obrigatório: Sim

O operador que indica qual operação executar para a verificação da variável.

No exemplo a seguir, a variável de saída do nome do repositório será verificada para verificar se é igual a `MyDemoRepo`.

```
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
```

Os operadores estão disponíveis para criar uma expressão da seguinte maneira.

- Equals: escolha esse operador para verificar se a variável é igual ao valor da string.

Parâmetro da CLI: `EQ`

- Contains: escolha esse operador para verificar se a variável contém o valor da string como uma substring.

  Parâmetro da CLI: `CONTAINS`

- Matches: escolha esse operador para verificar se a variável corresponde a uma determinada expressão regex como o valor da string.

  Todas as expressões regulares estão em CodePipeline conformidade com a sintaxe regex Java. Consulte uma descrição abrangente da sintaxe regex do Java e de seus constructos em java.util.regex.Pattern.

  Parâmetro da CLI: `MATCHES`

- Not equals: escolha esse operador para verificar se a variável não é igual ao valor da string.

  Parâmetro da CLI: `NE`

Variável

Obrigatório: Sim

As variáveis do pipeline a serem verificadas.

Valor

Obrigatório: Sim

O valor da expressão a ser verificada.

No exemplo a seguir, a variável de saída do nome do repositório será verificada para verificar se é igual a `MyDemoRepo`.

```
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
```

No exemplo de JSON a seguir, duas regras separadas são definidas, uma para uma instrução `EQ` (equals) que verifica o nome do repositório e da ramificação formatado como # {. SourceVariables

RepositoryName} e um para `CONTAINS` isso verifica a variável de saída da mensagem de confirmação formatada como # {SourceVariables. CommitMessage} em relação ao valor fornecido “atualizar”.

```
"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"
                              }
                          ]
                      }
```

```
                ]
            }
        }
    ],
```

Exemplo de configuração de regra

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"
    }
]
```

Consulte também

Os recursos relacionados a seguir podem ajudar você a trabalhar com essa regra.

- Tutorial: criar uma regra de verificação de variável para um pipeline como uma condição de entrada: esta seção fornece um tutorial com etapas para criar uma condição Na entrada com uma regra de verificação de variáveis.
- Referência de variáveis: esta seção fornece informações de referência e exemplos para variáveis de pipeline.
- Para obter mais informações sobre regras e condições, consulte Condição e RuleExecutionno Guia da CodePipeline API. RuleTypeId

Referência do modelo de integração

Há várias integrações predefinidas para serviços de terceiros que ajudarão a incorporar as ferramentas existentes do cliente no processo de liberação do pipeline. Parceiros ou provedores de serviços terceirizados usam um modelo de integração para implementar tipos de ação para uso em CodePipeline.

Use essa referência quando estiver planejando ou trabalhando com tipos de ação que são gerenciados com um modelo de integração compatível no CodePipeline.

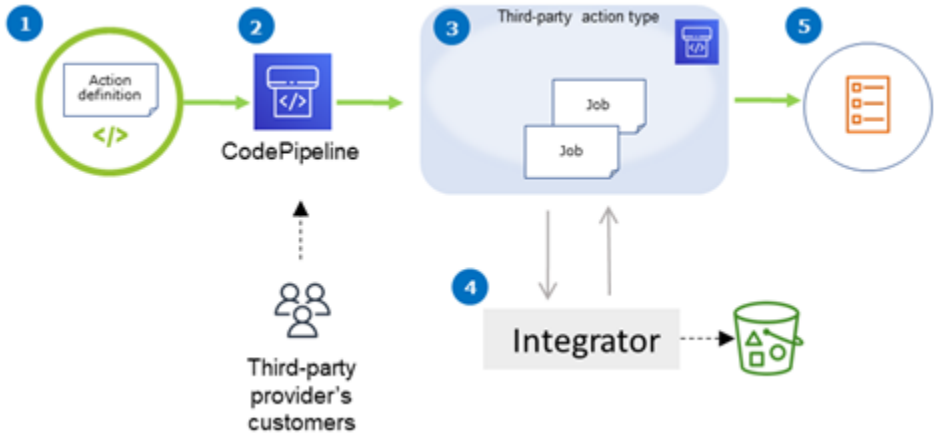
Para certificar seu tipo de ação de terceiros como uma integração de parceiro com CodePipeline, consulte a Rede de AWS parceiros (APN). Essas informações são um complemento à Referência da AWS CLI.

Tópicos

- Como os tipos de ação de terceiros funcionam com o integrador
- Conceitos
- Modelos de integração compatíveis
- Modelo de integração do Lambda
- Modelo de integração do operador de trabalho

Como os tipos de ação de terceiros funcionam com o integrador

Você pode adicionar tipos de ação de terceiros aos pipelines do cliente para concluir tarefas nos recursos do cliente. O integrador gerencia as solicitações de trabalho e executa a ação com CodePipeline. O diagrama a seguir mostra um tipo de ação de terceiros criado para os clientes usarem em seu pipeline. Depois que o cliente configura a ação, ela é executada e cria solicitações de trabalho que são processadas pelo mecanismo de ação do integrador.



O diagrama mostra as seguintes etapas:

1. A definição da ação é registrada e disponibilizada em CodePipeline. A ação de terceiros está disponível para clientes do provedor de terceiros.
2. O cliente do provedor escolhe e configura a ação em. CodePipeline
3. A ação é executada e os trabalhos são colocados na fila. CodePipeline Quando o trabalho está pronto CodePipeline, ele envia uma solicitação de trabalho.
4. O integrador (o operador de trabalho das APIs de sondagem de terceiros ou da função do Lambda) considera a solicitação de trabalho, retorna uma confirmação e trabalha nos artefatos das ações.
5. O integrador retorna a success/failure saída (o funcionário usa success/failure APIs ou a função Lambda envia a success/failure saída) com um resultado do trabalho e um token de continuação.

Para obter informações sobre as etapas que você pode usar para solicitar, visualizar e atualizar um tipo de ação, consulte Como trabalhar com tipos de ação.

Conceitos

Esta seção usa os seguintes termos para tipos de ação de terceiros:

Tipo de ação

Um processo repetível que pode ser reutilizado em pipelines que executam as mesmas workloads de entrega contínua. Os tipos de ação são identificados por `Owner`, `Category`, `Provider` e `Version`. Por exemplo:

```
            {

                "Category": "Deploy",
                "Owner": "AWS",
                "Provider": "CodeDeploy",
                "Version": "1"
            },
```

Todas as ações do mesmo tipo compartilham a mesma implementação.

Ação

Uma única instância de um tipo de ação, um dos processos discretos que ocorrem no estágio de um pipeline. Isso normalmente inclui os valores de usuário específicos do pipeline em que essa ação é executada.

Definição de ação

O esquema de um tipo de ação que define as propriedades necessárias para configurar a ação e os input/output artefatos.

Execução da ação

Uma coleção de trabalhos que foram executados para determinar se a ação no pipeline do cliente foi bem-sucedida ou não.

Mecanismo de execução de ações

Uma propriedade da configuração de execução da ação que define o tipo de integração usado por um tipo de ação. Os valores válidos são `JobWorker` e `Lambda`.

Integração

Descreve um software executado por um integrador para implementar um tipo de ação. CodePipeline suporta dois tipos de integração correspondentes aos dois mecanismos de ação suportados `JobWorker` `Lambda` e.

Integrador

A pessoa que é proprietária da implementação de um tipo de ação.

Trabalho

Um trabalho com pipeline e o contexto do cliente para executar uma integração. A execução de uma ação é composta por um ou mais trabalhos.

Operador de trabalho

O serviço que processa a entrada do cliente e executa um trabalho.

Modelos de integração compatíveis

CodePipeline tem dois modelos de integração:

- Modelo de integração Lambda: esse modelo de integração é a forma preferida de trabalhar com tipos de ação em CodePipeline. O modelo de integração do Lambda usa uma função do Lambda para processar solicitações de trabalho quando sua ação é executada.
- Modelo de integração do operador de trabalho: o modelo de integração do operador de trabalho é o modelo usado anteriormente para integrações de terceiros. O modelo de integração do trabalhador de trabalho usa um funcionário configurado para entrar em contato com as CodePipeline APIs para processar solicitações de trabalho quando sua ação é executada.

Para fins de comparação, a tabela a seguir descreve os atributos dos dois modelos:

	Modelo de integração do Lambda	Modelo de integração do operador de trabalho
Descrição	O integrador grava a integraçã o como uma função do Lambda, que é invocada CodePipeline sempre que há uma tarefa disponível para a ação. A função do Lambda não faz a sondagem dos trabalhos disponíveis, mas espera até que a próxima solicitaç ão de trabalho seja recebida.	O integrador grava a integração como um operador de trabalho que sonda constantemente os trabalhos disponíveis nos pipelines do cliente. Em seguida, o funcionário executa o trabalho e envia o resultado do trabalho de volta usando APIs. CodePipel ine CodePipeline
Infraestrutura	AWS Lambda	Implante o código do operador de trabalho na infraestrutura do integrador, como instâncias do Amazon EC2.

	Modelo de integração do Lambda	Modelo de integração do operador de trabalho
Esforço de desenvolvimento	A integração contém somente a lógica de negócios.	A integração precisa interagir com as CodePipeline APIs, além de conter a lógica de negócios.
Esforço operacional	Menor esforço operacional, pois a infraestrutura consiste apenas em AWS recursos.	Maior esforço operacional porque o operador de trabalho precisa do hardware autônomo.
Tempo máximo de execução do trabalho	Se a integração precisar ser executada ativamente por mais de 15 minutos, esse modelo não poderá ser usado. Essa ação destina-se a integradores que precisam iniciar um processo (por exemplo, iniciar uma compilação no artefato de código do cliente) e retornar um resultado quando ele for concluído. Não é recomendável que o integrador continue aguardando a conclusão da compilação. Em vez disso, retorne uma continuação. CodePipel inecria um novo trabalho em mais 30 segundos se uma continuação for recebida do código do integrador para verificar o trabalho até que ele seja concluído.	Trabalhos de execução muito longa (hours/days) podem ser mantidos usando esse modelo.

Modelo de integração do Lambda

O modelo de integração do Lambda compatível inclui a criação da função do Lambda e a definição da saída para o tipo de ação de terceiros.

Atualize sua função do Lambda para lidar com a entrada de CodePipeline

Crie uma nova função do Lambda. Você pode adicionar lógica de negócios à função do Lambda, que é executada sempre que há um trabalho disponível no pipeline para o tipo de ação. Por exemplo, considerando o contexto do cliente e do pipeline, talvez você queira começar a criar seu serviço para o cliente.

Use os parâmetros a seguir para atualizar sua função do Lambda para lidar com a entrada de CodePipeline.

Formato:

- `jobId`:
  - o ID exclusivo do trabalho gerado pelo sistema.
  - Tipo: string
  - Padrão: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}
- `accountId`:
  - O ID da AWS conta do cliente a ser usada ao realizar o trabalho.
  - Tipo: string
  - Padrão: [0-9]{12}
- `data`:
  - Outras informações sobre um trabalho que uma integração usa para concluir o trabalho.
  - Contém um mapa do seguinte:
    - `actionConfiguration`:
      - Os dados de configuração da ação. Os campos de configuração da ação são um mapeamento de pares de chave-valor para que o cliente insira os valores. As chaves são determinadas pelos parâmetros-chave no arquivo de definição de tipo de ação quando você configura sua ação. Neste exemplo, os valores são determinados pelo usuário da ação, especificando as informações nos campos `Username` e `Password`.
      - Tipo: String para mapa de strings, opcionalmente presente

      Exemplo:

```
"configuration": {
    "Username": "MyUser",
```

```
        "Password": "MyPassword"
    },
```

- `encryptionKey`:
  - Representa informações sobre a chave usada para criptografar dados no armazenamento de artefatos, como uma chave. AWS KMS
  - Conteúdo: Tipo do tipo de dados `encryptionKey`, opcionalmente presente
- `inputArtifacts`:
  - Lista de informações sobre um artefato a ser trabalhado, como um artefato de teste ou de compilação.
  - Conteúdo: Lista do tipo de dados `Artifact`, opcionalmente presente
- `outputArtifacts`:
  - Lista de informações sobre a saída de uma ação.
  - Conteúdo: Lista do tipo de dados `Artifact`, opcionalmente presente
- `actionCredentials`:
  - Representa um objeto AWS de credenciais de sessão. Essas credenciais são credenciais temporárias emitidas pelo AWS STS. Eles podem ser usados para acessar artefatos de entrada e saída no bucket do S3 usado para armazenar artefatos para o pipeline. CodePipeline

    Essas credenciais também têm as mesmas permissões do modelo de declarações de política especificado no arquivo de definição de tipo de ação.
  - Conteúdo: Tipo do tipo de dados `AWSSessionCredentials`, opcionalmente presente
- `actionExecutionId`:
  - O ID externo da execução da ação.
  - Tipo: string
- `continuationToken`:
  - Um token gerado pelo sistema, como uma ID de implantação, exigido por um trabalho para continuar o trabalho de forma assíncrona.
  - Tipo: String, opcionalmente presente

Tipos de dados:

- `encryptionKey`:

  - `id`:
    - O ID usado para identificar a chave. Para uma AWS KMS chave, você pode usar o ID da chave, o ARN da chave ou o alias ARN.
    - Tipo: string
  - `type`:
    - O tipo de chave de criptografia, como uma AWS KMS chave.
    - Tipo: string
    - Valores válidos: `KMS`
- `Artifact`:
  - `name`:
    - O nome do artefato.
    - Tipo: String, opcionalmente presente
  - `revision`:
    - O ID de revisão do artefato. Dependendo do tipo de objeto, isso pode ser um ID de confirmação (GitHub) ou um ID de revisão (Amazon S3).
    - Tipo: String, opcionalmente presente
  - `location`:
    - O local de um artefato.
    - Conteúdo: Tipo do tipo de dados `ArtifactLocation`, opcionalmente presente
- `ArtifactLocation`:
  - `type`:
    - O tipo de artefato no local.
    - Tipo: String, opcionalmente presente
    - Valores válidos: `S3`
  - `s3Location`:
    - O local do bucket do S3 que contém uma revisão.
    - Conteúdo: Tipo do tipo de dados `S3Location`, opcionalmente presente
- `S3Location`:
  - `bucketName`:
    - O nome do bucket do S3.
    - Tipo: string

- `objectKey`:
  - A chave do objeto no bucket do S3, que identifica exclusivamente o objeto no bucket.
  - Tipo: string
- `AWSSessionCredentials`:
  - `accessKeyId`:
    - A chave de acesso da sessão.
    - Tipo: string
  - `secretAccessKey`:
    - A chave de acesso secreta da sessão.
    - Tipo: string
  - `sessionToken`:
    - O token da sessão.
    - Tipo: string

Exemplo:

```
{
    "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"
    }
}
```

Retorne os resultados da sua função Lambda para CodePipeline

O operador de trabalho do integrador deve retornar uma payload válida em casos de êxito, falha ou continuação.

Formato:

- `result`: o resultado do trabalho.
  - Obrigatório
  - Valores válidos (sem distinção entre maiúsculas e minúsculas):
    - `Success`: indica que um trabalho foi bem-sucedido e foi encerrado.
    - `Continue`: indica que um trabalho foi bem-sucedido e deve continuar, por exemplo, se o operador de trabalho for chamado novamente para executar a mesma ação.
    - `Fail`: indica que um trabalho apresentou falha e foi encerrado.
- `failureType`: um tipo de falha a ser associado a um trabalho com falha.

A categoria `failureType` de ações do parceiro descreve o tipo de falha encontrado durante a execução do trabalho. Os integradores definem o tipo junto com a mensagem de falha ao retornar o resultado de uma falha de trabalho para CodePipeline.

- Opcional. Obrigatório se o resultado for `Fail`.
- Deve ser nulo se `result` for `Success` ou `Continue`
- Valores válidos:
  - ConfigurationError
  - JobFailed
  - PermissionsError
  - RevisionOutOfSync
  - RevisionUnavailable
  - SystemUnavailable

- `continuation`: estado de continuação a ser passado para o próximo trabalho na execução da ação atual.
  - Opcional. Obrigatório se o resultado for `Continue`.
  - Deve ser nulo se `result` for `Success` ou `Fail`.
  - Propriedades:
    - `State`: um hash do estado a ser passado.
- `status`: status da execução da ação.
  - Opcional.
  - Propriedades:
    - `ExternalExecutionId`: um ID de execução externa opcional ou ID de confirmação a ser associado ao trabalho.
    - `Summary`: um resumo opcional do que ocorreu. Em cenários de falha, essa será a mensagem de falha exibida para o usuário.
- `outputVariables`: Um conjunto de key/value pares a serem passados para a próxima execução da ação.
  - Opcional.
  - Deve ser nulo se `result` for `Continue` ou `Fail`.

Exemplo:

```
{
    "result": "success",
    "failureType": null,
    "continuation": null,
    "status": {
        "externalExecutionId": "my-commit-id-123",
        "summary": "everything is dandy"
    },
    "outputVariables": {
        "FirstOne": "Nice",
        "SecondOne": "Nicest",
        ...
    }
}
```

Use tokens de continuação para aguardar os resultados de um processo assíncrono

O token `continuation` faz parte da payload e do resultado da função do Lambda. É uma forma de transmitir o estado do trabalho CodePipeline e indicar que o trabalho precisa ser continuado. Por exemplo, depois que um integrador inicia uma compilação para o cliente em seu recurso, ele não espera a conclusão da compilação, mas indica CodePipeline que não tem um resultado terminal retornando o `result` as `continue` e retornando o ID exclusivo da compilação para o `continuation` token CodePipeline as.

> **Note**
>
> As funções do Lambda só podem ser executadas por até 15 minutos. Se o trabalho precisar ser executado por mais tempo, você poderá usar tokens de continuação.

A CodePipeline equipe invoca o integrador após 30 segundos com o mesmo `continuation` token em sua carga para que ele possa verificar se ele está concluído. Se a compilação for concluída, o integrador retornará o success/fail resultado do terminal, caso contrário, continuará.

Forneça CodePipeline as permissões para invocar a função Lambda do integrador em tempo de execução

Você adiciona permissões à função Lambda do integrador para fornecer ao CodePipeline serviço permissões para invocá-lo usando o principal do CodePipeline serviço:. `codepipeline.amazonaws.com` Você pode adicionar permissões usando CloudFormation ou a linha de comando. Para ver um exemplo, consulte Como trabalhar com tipos de ação.

Modelo de integração do operador de trabalho

Após designar o fluxo de trabalho de alto nível, você poderá criar o operador de trabalho. Embora as especificidades da ação de terceiros determine o que é necessário para o operador de trabalho, a maioria dos operadores de trabalho das ações de terceiros incluem a seguinte funcionalidade:

- Pesquisa de empregos a partir do CodePipeline uso`PollForThirdPartyJobs`.
- Reconhecendo trabalhos e retornando resultados ao CodePipeline usar `AcknowledgeThirdPartyJob`PutThirdPartyJobSuccessResult, e. `PutThirdPartyJobFailureResult`
- Recuperação de artefatos ao and/or colocar artefatos no bucket do Amazon S3 para o pipeline. Para fazer download dos artefatos no bucket do Amazon S3, é necessário criar um cliente do Amazon S3 que use a assinatura do Signature versão 4 (Sig V4). O Sig V4 é necessário para. AWS KMS

  Para fazer upload de artefatos para o bucket do Amazon S3, você também deve configurar a `PutObject` solicitação do Amazon S3 para usar criptografia por meio AWS Key Management Service de ().AWS KMS AWS KMS usa AWS KMS keys. Para saber se deve usar a chave gerenciada pelo cliente Chave gerenciada pela AWS ou uma chave gerenciada pelo cliente para fazer upload de artefatos, seu funcionário deve examinar os dados do trabalho e verificar a propriedade da chave de criptografia. Se a propriedade estiver definida, você deverá usar esse ID de chave gerenciado pelo cliente ao configurar. AWS KMS Se a propriedade da chave for nula, você usa o. Chave gerenciada pela AWS CodePipeline usa o, a Chave gerenciada pela AWS menos que seja configurado de outra forma.

  Para ver um exemplo que mostra como criar os AWS KMS parâmetros em Java ou.NET, consulte Especificação do AWS Key Management Service no Amazon S3 usando os AWS SDKs. Para obter mais informações sobre o bucket Amazon S3 para CodePipeline, consulteCodePipeline conceitos.

Escolher e configurar uma estratégia de gerenciamento de permissões para o operador de trabalho

Para desenvolver um profissional de trabalho para sua ação terceirizada em CodePipeline, você precisa de uma estratégia para a integração do gerenciamento de usuários e permissões.

A estratégia mais simples é adicionar a infraestrutura de que você precisa para seu funcionário criando instâncias do Amazon EC2 com uma função de instância AWS Identity and Access Management (IAM), o que permite que você amplie facilmente os recursos necessários para sua integração. Você pode usar a integração integrada com AWS para simplificar a interação entre seu funcionário CodePipeline e.

Saiba mais sobre o Amazon EC2 e determine se essa é a escolha certa para sua integração. Para obter mais informações, consulte Amazon EC2 – Hospedagem de servidor virtual. Para obter informações sobre como configurar uma instância do Amazon EC2, consulte Conceitos básicos das instâncias do Linux do Amazon EC2.

Outra estratégia a ser considerada é usar a federação de identidades com o IAM para integrar o sistema e os recursos existentes do provedor de identidades. Essa estratégia é útil se você já tiver um provedor de identidades corporativas ou já estiver configurado para oferecer suporte aos usuários usando provedores de identidades da web. A federação de identidades permite que você conceda acesso seguro aos AWS recursos CodePipeline, inclusive, sem precisar criar ou gerenciar usuários do IAM. É possível usar recursos e políticas para requisitos de segurança de senhas e rotação de credenciais. Você pode usar aplicações de exemplo como modelos para seu próprio design. Para mais informações, consulte Gerenciar federação.

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos em Centro de Identidade do AWS IAM:

  Crie um conjunto de permissões. Siga as instruções em Criação de um conjunto de permissões no Guia do usuário do Centro de Identidade do AWS IAM .
- Usuários gerenciados no IAM com provedor de identidades:

  Crie um perfil para a federação de identidades. Siga as instruções em Criando um perfil para um provedor de identidades de terceiros (federação) no Guia do Usuário do IAM.
- Usuários do IAM:
  - Crie um perfil que seu usuário possa assumir. Siga as instruções em Criação de um perfil para um usuário do IAM no Guia do usuário do IAM.

- (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em Adição de permissões a um usuário (console) no Guia do usuário do IAM.

Veja a seguir uma política de exemplo que você pode criar para usar com o operador de trabalho de terceiros. Essa política serve somente como exemplo e é fornecida no estado em que encontra.

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/"

      ]
    }
  ]
}
```

Referência de arquivo de definições de imagem

Esta seção é apenas uma referência. Para obter informações sobre a criação de um pipeline com origem ou implantar ações para contêineres, consulte Criar um pipeline, estágios e ações.

AWS CodePipeline funcionários de ações de contêiner, como uma ação de origem do Amazon ECR ou ações de implantação do Amazon ECS, usam arquivos de definições para mapear o URI da imagem e o nome do contêiner para a definição da tarefa. Cada arquivo de definições é um JSON-formatted arquivo usado pelo provedor de ação da seguinte forma:

- As implantações padrão do Amazon ECS requerem um arquivo `imagedefinitions.json` como entrada para a ação de implantação. Para ver um tutorial que usa a ação de implantação padrão do Amazon ECS em CodePipeline, consulteTutorial: Implantação padrão do Amazon ECS com CodePipeline. Para ver outro exemplo de tutorial que usa a ação de implantação padrão do Amazon ECS CodePipeline junto com a ECRBuildAndPublish ação, consulteTutorial: crie e envie uma imagem do Docker para o Amazon ECR com CodePipeline (tipo V2).
- blue/green As implantações do Amazon ECS exigem um `imageDetail.json` arquivo como entrada para a ação de implantação. Para ver um tutorial com um exemplo blue/green de implantação, consulteTutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR.
  - Ações de origem do Amazon ECR geram um arquivo `imageDetail.json`, que é fornecido como uma saída da ação de origem.

Tópicos

- Arquivo imagedefinitions.json para ações de implantação padrão do Amazon ECS
- Detail.json arquivo de imagem para ações de blue/green implantação do Amazon ECS

Arquivo imagedefinitions.json para ações de implantação padrão do Amazon ECS

Um documento de definições de imagem é um arquivo JSON que descreve o nome de contêiner, a imagem e a tag do Amazon ECS. Se você estiver implantando aplicativos baseados em contêineres, deverá gerar um arquivo de definições de imagem para fornecer ao trabalhador o CodePipeline contêiner do Amazon ECS e a identificação da imagem para recuperar do repositório de imagens, como o Amazon ECR.

> **Note**
>
> O nome do arquivo padrão para o arquivo é `imagedefinitions.json`. Se você optar por usar um nome de arquivo diferente, você deve fornecê-lo ao criar o estágio de implantação do pipeline.

Crie o arquivo `imagedefinitions.json` considerando o seguinte:

- O arquivo deve usar UTF-8 codificação.
- O limite máximo de tamanho de um arquivo de definições de imagem é 100 KB.
- Você deve criar o arquivo como origem ou artefato de compilação para que ele seja um artefato de entrada para a ação de implantação. Em outras palavras, certifique-se de que o arquivo seja carregado para seu local de origem, como seu CodeCommit repositório, ou gerado como um artefato de saída construído.

O arquivo `imagedefinitions.json` fornece o nome do contêiner e o URI da imagem. Ele deve ser construído com o seguinte conjunto de pares de chave/valor.

Chave	Valor
name	*container_name*
imageUri	*imageUri*

> **Note**
>
> O campo nome é usado para o nome da imagem do contêiner, ou seja, o nome da imagem do Docker.

Aqui está a estrutura JSON, na qual o nome do contêiner é `sample-app`. O URI da imagem é `ecs-repo` e a tag é `latest`:

```
[
  {
    "name": "sample-app",
```

```
    "imageUri": "11111EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest"
  }
]
```

Você também pode construir o arquivo para listar vários pares de contêiner-imagem.

Estrutura do JSON:

```
[
  {
    "name": "simple-app",
    "imageUri": "httpd:2.4"
  },
  {
    "name": "simple-app-1",
    "imageUri": "mysql"
  },
  {
    "name": "simple-app-2",
    "imageUri": "java1.8"
  }
]
```

Antes de criar o pipeline, use as etapas a seguir para configurar o arquivo `imagedefinitions.json`.

1. Como parte do planejamento de implantação de aplicativos baseados em contêiner para o pipeline, planejar o estágio de origem e o estágio de compilação, se aplicável.
2. Escolha uma das seguintes opções:

   a. Se o pipeline foi projetado para ignorar o estágio de compilação, será necessário criar o arquivo JSON manualmente e submetê-lo a upload para o repositório de origem a fim de que a ação de origem possa fornecer o artefato. Crie o arquivo usando um editor de texto, atribua um nome ao arquivo ou use o nome de arquivo padrão `imagedefinitions.json`. Envie o arquivo de definições de imagem ao repositório de origem.

   > **Note**
   >
   > Se o repositório de origem for um bucket do Amazon S3, lembre-se de compactar o arquivo JSON.

b. Se seu pipeline tiver um estágio de construção, adicione um comando ao seu arquivo buildspec que gera o arquivo de definições de imagem em seu repositório de origem durante a fase de construção. O exemplo a seguir usa o comando printf para criar um arquivo `imagedefinitions.json`. Liste esse comando na seção `post_build` do arquivo `buildspec.yml`:

```
printf '[{"name":"container_name","imageUri":"image_URI"}]' >
imagedefinitions.json
```

Você deve incluir o arquivo de definições de imagem como artefato de saída no arquivo `buildspec.yml`.

3. Ao criar o pipeline no console, na página Deploy (Implantar) do assistente Create Pipeline (Criar pipeline), em Image Filename (Nome da imagem), insira o nome de arquivo de definições da imagem.

Para ver um tutorial detalhado sobre a criação de um pipeline que usa o Amazon ECS como provedor de implantação, consulte Tutorial: implantação contínua com. CodePipeline

Detail.json arquivo de imagem para ações de blue/green implantação do Amazon ECS

Um documento `imageDetail.json` é um arquivo JSON que descreve o URI da imagem do Amazon ECS. Se você estiver implantando aplicativos baseados em contêineres para uma blue/green implantação, deverá gerar o `imageDetail.json` arquivo para fornecer ao Amazon ECS e CodeDeploy ao funcionário a identificação da imagem a ser recuperada do repositório de imagens, como o Amazon ECR.

> **Note**
>
> O nome do arquivo deve ser `imageDetail.json`.

Para obter uma descrição da ação e respectivos parâmetros, consulte Amazon Elastic Container Service e referência de ação de implantação CodeDeploy azul-esverdeada.

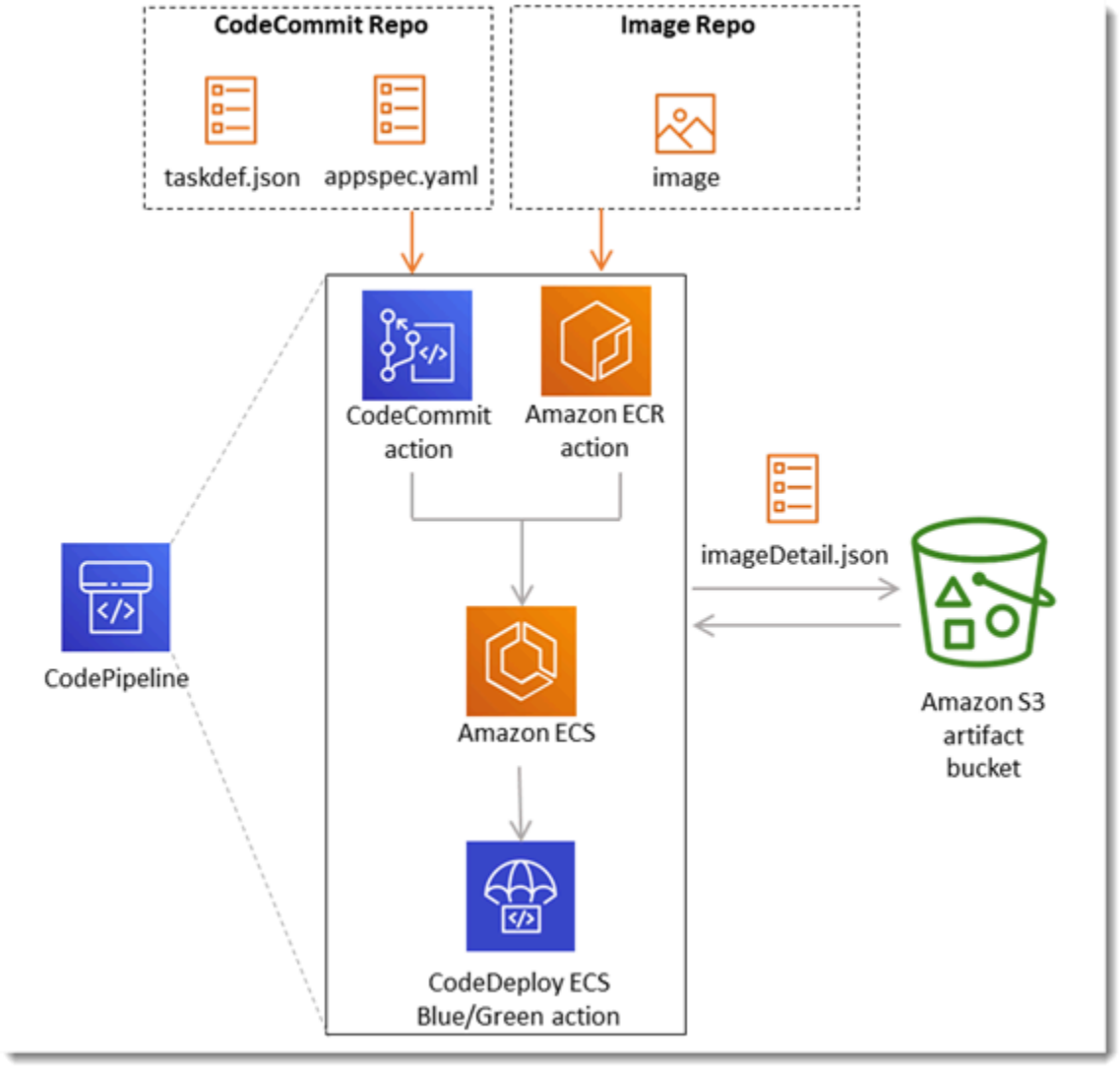
Você deve criar o arquivo `imageDetail.json` como origem ou artefato de compilação para que ele seja um artefato de entrada para a ação de implantação. Você pode usar um destes métodos para fornecer o arquivo `imageDetail.json` no pipeline:

- Inclua o `imageDetail.json` arquivo em seu local de origem para que ele seja fornecido no pipeline como entrada para sua ação de blue/green implantação do Amazon ECS.

  > **Note**
  >
  > Se o repositório de origem for um bucket do Amazon S3, lembre-se de compactar o arquivo JSON.

- Ações de origem do Amazon ECR geram automaticamente um arquivo `imageDetail.json` como artefato de entrada para a próxima ação.

  > **Note**
  >
  > Como a ação de origem do Amazon ECR cria esse arquivo, os pipelines com uma ação de origem do Amazon ECR não precisam fornecer um arquivo `imageDetail.json` manualmente.
  > Para obter um tutorial sobre a criação de um pipeline que inclua um estágio de origem do Amazon ECR, consulte Tutorial: Crie um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR.



O arquivo `imageDetail.json` fornece o URI da imagem. Ele deve ser construído com o seguinte par de chave/valor.

Chave	Valor
ImageURI	*image_URI*

imageDetail.json

Esta é a estrutura JSON, em que o URI da imagem é `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)

Um arquivo `imageDetail.json` é gerado automaticamente pela ação de origem do Amazon ECR cada vez que uma alteração é enviada por push ao repositório de imagens. O `imageDetail.json` gerado por ações de origem do Amazon ECR é fornecido como um artefato de saída da ação de origem para a próxima ação no pipeline.

Esta é a estrutura JSON, em que o nome do repositório é `dk-image-repo`, o URI da imagem é `ecs-repo`, e a tag de imagem é `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"
    ]
}
```

O arquivo `imageDetail.json` mapeia o URI da imagem e nome do contêiner para a definição de tarefa do Amazon ECS da seguinte maneira:

- `ImageSizeInBytes`: O tamanho, em bytes, da imagem no repositório.
- `ImageDigest`: A compilação `sha256` do manifesto da imagem.
- `Version`: A versão da imagem.
- `ImagePushedAt`: A data e a hora em que a última imagem foi enviada ao repositório.
- `RegistryId`: O ID da AWS conta associado ao registro que contém o repositório.
- `RepositoryName`: o nome do repositório do Amazon ECR ao qual a imagem foi enviada por push.

- `ImageURI`: O URI para a imagem.
- `ImageTags`: A tag usada para a imagem.

Antes de criar o pipeline, use as etapas a seguir para configurar o arquivo `imageDetail.json`.

1. Como parte do planejamento da blue/green implantação do aplicativo baseado em contêiner para seu pipeline, planeje o estágio de origem e o estágio de criação, se aplicável.
2. Escolha uma das seguintes opções:

   a. Se seu pipeline pulou a fase de criação, você deve criar manualmente o arquivo JSON e enviá-lo para seu repositório de origem, por exemplo, para que a ação de origem possa fornecer o artefato. CodeCommit Crie o arquivo usando um editor de texto, atribua um nome ao arquivo ou use o nome de arquivo padrão `imageDetail.json`. Envie o arquivo `imageDetail.json` para o repositório de origem.

   b. Se o pipeline tiver um estágio de compilação, execute o seguinte:

      i. Adicione um comando ao seu arquivo buildspec que gera o arquivo de definições de imagem em seu repositório de origem durante a fase de construção. O exemplo a seguir usa o comando printf para criar um arquivo `imageDetail.json`. Liste esse comando na seção `post_build` do arquivo buildspec.yml:

      ```
      printf '{"ImageURI":"image_URI"}' > imageDetail.json
      ```

      Você deve incluir o arquivo `imageDetail.json` como um artefato de saída no arquivo `buildspec.yml`.

      ii. Adicione o `imageDetail.json` como artefato no arquivo `buildspec.yml`.

      ```
      artifacts:
        files:
          - imageDetail.json
      ```

Referência de variáveis

Esta seção é apenas uma referência. Para obter informações sobre como criar variáveis, consulte Trabalhar com variáveis.

As variáveis permitem configurar as ações de pipeline com valores determinados no momento da execução do pipeline ou da ação.

Alguns provedores de ação produzem um conjunto definido de variáveis. Escolha entre chaves de variáveis padrão para esse provedor de ação, como ID de confirmação.

> **Important**
>
> Ao transmitir parâmetros secretos, não insira o valor diretamente. O valor é renderizado como texto sem formatação e, portanto, é legível. Por motivos de segurança, não use texto sem formatação com segredos. É altamente recomendável que você use AWS Secrets Manager para armazenar segredos.

Para ver exemplos passo a passo do uso de variáveis:

- Para assistir a um tutorial com uma variável no nível do pipeline que é passada no momento da execução do pipeline, consulte. Tutorial: Usar variáveis no nível do pipeline
- Para ver um tutorial com uma ação do Lambda que usa variáveis de uma ação upstream (CodeCommit) e gera variáveis de saída, consulte. Tutorial: Usar variáveis com ações de invocação do Lambda
- Para ver um tutorial com uma CloudFormation ação que faz referência às variáveis de saída da pilha de uma CloudFormation ação upstream, consulte. Tutorial: criar um pipeline que usa variáveis de AWS CloudFormation ações de implantação
- Para ver um exemplo de ação de aprovação manual com texto de mensagem que faz referência a variáveis de saída que são resolvidas para o CodeCommit ID de confirmação e a mensagem de confirmação, consulteExemplo: Usar variáveis em aprovações manuais.
- Para ver um exemplo de CodeBuild ação com uma variável de ambiente que resolve para o nome da GitHub ramificação, consulteExemplo: use uma BranchName variável com variáveis de CodeBuild ambiente.

- CodeBuild as ações produzem como variáveis todas as variáveis de ambiente que foram exportadas como parte da construção. Para obter mais informações, consulte CodeBuild variáveis de saída da ação.

Limites de variáveis

Para obter informações sobre limites, consulte Cotas em AWS CodePipeline.

> **Note**
>
> Ao inserir a sintaxe da variável nos campos de configuração da ação, não ultrapasse o limite de mil caracteres dos campos de configuração. Um erro de validação será retornado quando esse limite for excedido.

Tópicos

- Conceitos
- Casos de uso de variáveis
- Configurar variáveis
- Resolução de variáveis
- Regras para variáveis
- Variáveis disponíveis para ações de pipeline

Conceitos

Esta seção lista os principais termos e conceitos relacionados a variáveis e namespaces.

Variáveis

As variáveis são pares chave-valor que podem ser usados para configurar dinamicamente ações em seu pipeline. No momento, existem três maneiras de disponibilizar essas variáveis:

- Há um conjunto de variáveis que estão implicitamente disponíveis no início de cada execução do pipeline. Atualmente, esse conjunto inclui `PipelineExecutionId`, o ID da execução atual do pipeline.

- As variáveis em nível de pipeline são definidas quando o pipeline é criado e resolvido no runtime do pipeline.

  Você deve especificar variáveis em nível do pipeline quando o pipeline é criado e pode fornecer valores no momento da execução do pipeline.

- Existem tipos de ação que produzem conjuntos de variáveis quando são executados. Você pode ver as variáveis produzidas por uma ação inspecionando o `outputVariables` campo que faz parte da ListActionExecutions API. Para obter uma lista de nomes de chaves disponíveis por provedor de ação, consulte Variáveis disponíveis para ações de pipeline. Para ver quais variáveis cada tipo de ação produz, consulte CodePipeline Referência da estrutura da ação o.

Para fazer referência a essas variáveis na configuração da ação, você deve usar a sintaxe de referência de variável com o namespace correto.

Para obter um fluxo de trabalho variável de exemplo, consulte Configurar variáveis.

Namespaces

Para garantir que as variáveis possam ser referenciadas exclusivamente, elas devem ser atribuídas a um namespace. Depois de um conjunto de variáveis serem atribuídas a um namespace, elas podem ser referenciadas em uma configuração de ação usando o namespace e a chave de variável com a seguinte sintaxe:

```
#{namespace.variable_key}
```

Existem três tipos de namespaces sob os quais as variáveis podem ser atribuídas:

- O namespace reservado do codepipeline

  Esse é o namespace atribuído ao conjunto de variáveis implícitas disponíveis no início de cada execução do pipeline. Esse namespace é o `codepipeline`. Exemplo de referência de variável:

  ```
  #{codepipeline.PipelineExecutionId}
  ```

- O namespace das variáveis no nível do pipeline

  Esse é o namespace atribuído às variáveis no nível do pipeline. O namespace de todas as variáveis em nível de pipeline é `variables`. Exemplo de referência de variável:

```
#{variables.variable_name}
```

- Namespace atribuído à ação

  Esse é um namespace que você atribui a uma ação. Todas as variáveis produzidas pela ação se enquadram nesse namespace. Para tornar as variáveis produzidas por uma ação disponíveis para uso em uma configuração de ação downstream, você deve configurar a ação de produção com um namespace. Os namespaces devem ser exclusivos na definição do pipeline e não podem entrar em conflito com nenhum nome de artefato. Veja uma referência variável de exemplo para uma ação configurada com um namespace de `SourceVariables`.

  ```
  #{SourceVariables.VersionId}
  ```

Casos de uso de variáveis

Veja a seguir alguns dos casos de uso mais comuns de variáveis em nível de pipeline para ajudar você a determinar como usar as variáveis para necessidades específicas.

- As variáveis no nível do pipeline são para CodePipeline clientes que desejam usar o mesmo pipeline todas as vezes, com pequenas variações nas entradas da configuração da ação. Qualquer desenvolvedor que inicie um pipeline inclui o valor da variável na interface do usuário quando o pipeline é iniciado. Com essa configuração, você deve transmitir parâmetros somente para essa execução.
- Com variáveis em nível de pipeline, é possível transmitir entradas dinâmicas para ações no pipeline. Você pode migrar seus pipelines parametrizados para CodePipeline sem precisar manter versões diferentes do mesmo pipeline ou criar pipelines complexos.
- É possível usar variáveis em nível de pipeline para transmitir parâmetros de entrada que permitem reutilizar um pipeline a cada execução, como quando você deseja especificar qual versão deseja implantar em um ambiente de produção, para não precisar duplicar os pipelines.
- É possível usar um único pipeline para implantar recursos em vários ambientes de criação e implantação. Por exemplo, para um pipeline com um CodeCommit repositório, a implantação a partir de uma ramificação específica e de um ambiente de implantação de destino pode ser feita com CodeBuild CodeDeploy parâmetros passados no nível do pipeline.

> **Note**
>
> Para Amazon ECR, Amazon S3 ou CodeCommit fontes, você também pode criar uma substituição de origem usando a entrada de transformação de entrada para usar o `revisionValue` in EventBridge para seu evento de pipeline, em que `revisionValue` é derivado da variável de evento de origem para sua chave de objeto, confirmação ou ID de imagem. Para saber mais, consulte a etapa opcional da entrada da transformação de entrada incluída nos procedimentos em Ações e recursos de origem do Amazon ECR EventBridge, Conexão a ações de origem do Amazon S3 com uma fonte habilitada para eventos ou CodeCommit ações de origem e EventBridge.

Configurar variáveis

É possível configurar variáveis em nível de pipeline ou em nível de ação na estrutura do pipeline.

Configurar variáveis em nível de pipeline

É possível adicionar uma ou mais variáveis em nível de pipeline. Você pode referenciar esse valor na configuração das CodePipeline ações. É possível adicionar os nomes das variáveis, os valores padrão e as descrições ao criar o pipeline. As variáveis são resolvidas no momento da execução.

> **Note**
>
> Se um valor padrão não for definido para uma variável em nível de pipeline, a variável será considerada necessária. É necessário especificar substituições para todas as variáveis necessárias ao iniciar um pipeline, caso contrário, a execução do pipeline falhará com um erro de validação.

Forneça variáveis no nível do pipeline usando o atributo variables na estrutura do pipeline. No exemplo a seguir, o valor da variável `Variable1` é `Value1`.

```
    "variables": [
        {
            "name": "Variable1",
            "defaultValue": "Value1",
            "description": "description"
        }
```

```
        ]
```

Para ver um exemplo de estrutura JSON do pipeline, consulte Criar um pipeline, estágios e ações.

Para assistir a um tutorial com uma variável no nível do pipeline que é passada no momento da execução do pipeline, consulte. Tutorial: Usar variáveis no nível do pipeline

Observe que o uso de variáveis em nível de pipeline em qualquer tipo de ação de origem não é aceito.

> **Note**
>
> Se o namespace `variables` já estiver sendo usado em algumas ações no pipeline, você deverá atualizar a definição da ação e escolher outro namespace para a ação conflitante.

Configuração de variáveis no nível da ação

Configure uma ação para produzir variáveis declarando um namespace para a ação. A ação já deve ser um dos provedores da ação que gera variáveis. Caso contrário, as variáveis disponíveis serão variáveis em nível de pipeline.

Declare o namespace:

- Na página Edit action (Editar ação) do console, digitando um namespace em Variable namespace (Namespace de variáveis).
- Inserindo um namespace no campo do parâmetro `namespace` na estrutura do pipeline JSON.

Neste exemplo, você adiciona o `namespace` parâmetro à ação de CodeCommit origem com o nome`SourceVariables`. Isso configura a ação para produzir as variáveis disponíveis para esse provedor de ação, 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
        }
    ]
},
```

Depois, configure a ação downstream para usar as variáveis produzidas pela ação anterior. Faça isso:

- Na página Edit action (Editar ação) do console, inserindo a sintaxe de variável (para a ação downstream) nos campos de configuração da ação.
- Entrando a sintaxe de variável (da ação downstream) nos campos de configuração da ação na estrutura do pipeline JSON

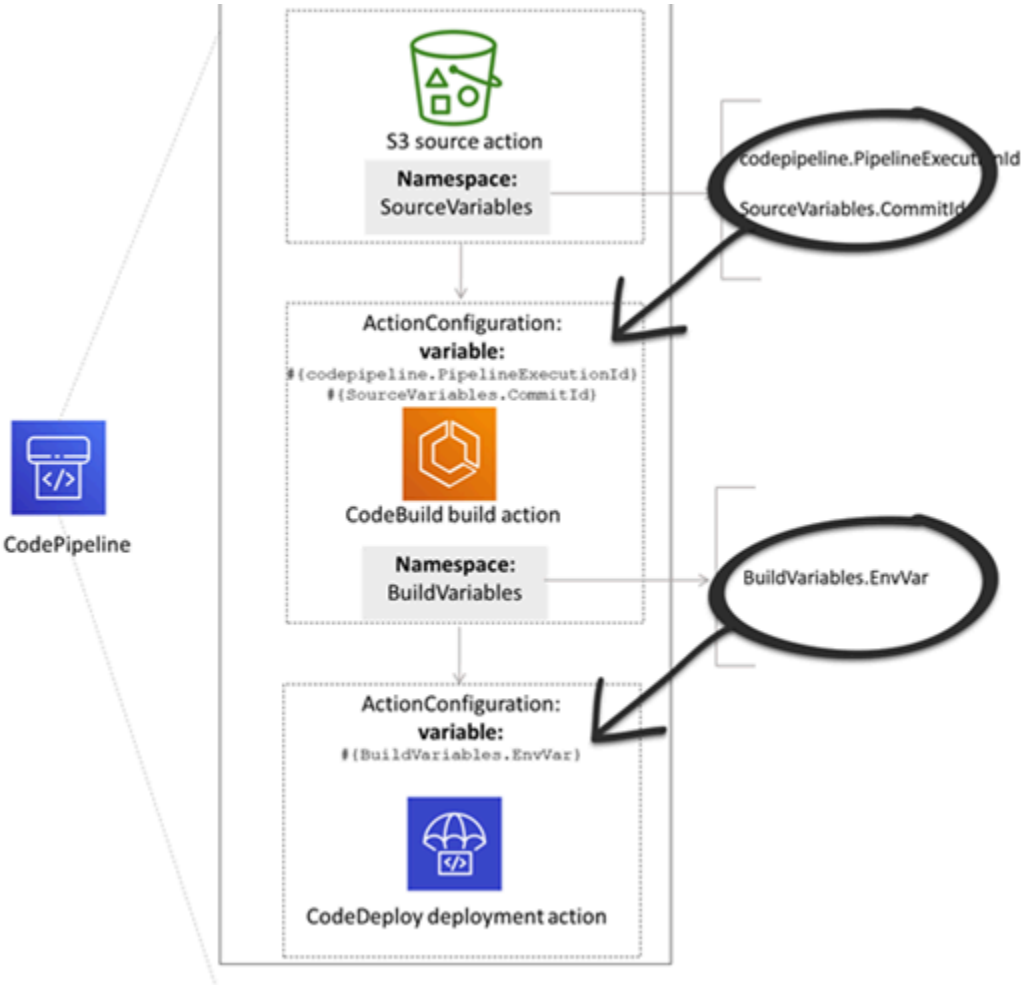
Neste exemplo, o campo de configuração da ação de compilação mostra variáveis de ambiente que são atualizadas na execução da ação. O exemplo especifica o namespace e a variável para o ID de execução com `#{codepipeline.PipelineExecutionId}` e o namespace e a variável do ID de confirmação com **`#{SourceVariables.CommitId}`**.

```
{
    "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
        }
    ]
},
```

Resolução de variáveis

Cada vez que uma ação é executada como parte de uma execução do pipeline, as variáveis que ela produz estão disponíveis para uso em qualquer ação que, garantidamente, ocorrerá após a ação de produção. Para usar essas variáveis em uma ação de consumo, você pode adicioná-las à configuração da ação de consumo usando a sintaxe mostrada no exemplo anterior. Antes de executar uma ação de consumo, CodePipeline resolve todas as referências de variáveis presentes na configuração antes de iniciar a execução da ação.



Regras para variáveis

As regras a seguir ajudam você na configuração de variáveis:

- Especifique o namespace e a variável para uma ação por meio de uma nova propriedade de ação ou editando uma ação.
- Quando você usa o assistente de criação de pipeline, o console gera um namespace para cada ação criada com o assistente.
- Se o namespace não for especificado, as variáveis produzidas por essa ação não poderão ser referenciadas em nenhuma configuração de ação.
- Para referenciar variáveis produzidas por uma ação, a ação de referência deve ocorrer após a ação que produz as variáveis. Isso significa que ela está em um estágio posterior à ação que produz as variáveis, ou no mesmo estágio, mas em uma ordem de execução posterior.

Variáveis disponíveis para ações de pipeline

O provedor de ação determina quais variáveis podem ser geradas pela ação.

Para obter os procedimentos passo a passo para gerenciamento de variáveis, consulte Trabalhar com variáveis.

Ações com chaves variáveis definidas

Ao contrário de um namespace que você pode escolher, as ações a seguir usam chaves de variáveis não podem ser editadas. Por exemplo, para o provedor de ação do Amazon S3, apenas as chaves de variáveis `ETag` e `VersionId` estão disponíveis.

Cada execução também tem um conjunto de variáveis do CodePipeline-generated pipeline que contêm dados sobre a execução, como o ID da versão do pipeline. Essas variáveis podem ser consumidas por qualquer ação no pipeline.

Tópicos

- CodePipeline variável de ID de execução
- Variáveis de saída de ação do Amazon ECR
- CloudFormation StackSets variáveis de saída da ação
- CodeCommit variáveis de saída da ação
- CodeStarSourceConnection variáveis de saída da ação
- GitHub variáveis de saída de ação GitHub (por meio do aplicativo OAuth) ação)
- Variáveis de saída da ação do S3

CodePipeline variável de ID de execução

CodePipeline variável de ID de execução

Fornecedor	Chave de variável	Valor de exemplo	Exemplo de sintaxe de variável
codepipeline	`PipelineExecutionId`	8abc75f0-fbf8-4f4c-bfEXAMPLE	`#{codepipeline.PipelineExecutionId}`

Variáveis de saída de ação do Amazon ECR

Variáveis do Amazon ECR

Chave de variável	Valor de exemplo	Exemplo de sintaxe de variável
`ImageDigest`	sha256:EXAMPLE1122334455	`#{SourceVariables.ImageDigest}`
`ImageTag`	mais recente	`#{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 variáveis de saída da ação

CloudFormation StackSets variáveis

Chave de variável	Valor de exemplo	Exemplo de sintaxe de variável
`OperationId`	11111111-2bbb-111-2bbb-11111example	`#{DeployVariables.OperationId}`
`StackSetId`	my-stackset:1111aaaa-1111-2222-2bbb-11111example	`#{DeployVariables.StackSetId}`

CodeCommit variáveis de saída da ação

CodeCommit variáveis

Chave de variável	Valor de exemplo	Exemplo de sintaxe de variável
`AuthorDate`	2019-10-29 T03:32:21Z	`#{SourceVariables.AuthorDate}`
`BranchName`	acontecimento	`#{SourceVariables.BranchName}`
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Corrigido um erro (tamanho máximo de 100 KB)	`#{SourceVariables.CommitMessage}`
`CommitterDate`	2019-10-29 T03:32:21Z	`#{SourceVariables.CommitterDate}`
`RepositoryName`	meu CodeCommitRepo	`#{SourceVariables.RepositoryName}`

CodeStarSourceConnection variáveis de saída da ação

As variáveis de saída disponíveis dependem do tipo de acionador (push, pull request ou tag). As variáveis na tabela a seguir marcadas como Todas são produzidas para cada tipo de acionador. As variáveis marcadas com um tipo de acionador específico só são produzidas quando a execução do pipeline é iniciada por esse acionador.

`CodeStarSourceConnection`variáveis (Bitbucket Cloud GitHub, GitHub Enterprise Repository e) GitLab.com

Chave de variável	Valor de exemplo	Exemplo de sintaxe de variável	Tipo de gatilho
`AuthorDate`	2019-10-29 T03:32:21Z	`#{SourceVariables.AuthorDate}`	Todos
`AuthorDisplayName`	Jane Doe	`#{SourceVariables.AuthorDisplayName}`	Todos
`AuthorEmail`	jane.doe@example.com	`#{SourceVariables.AuthorEmail}`	Todos
`AuthorId`	janedoe	`#{SourceVariables.AuthorId}`	Todos
`BranchName`	acontecimento	`#{SourceVariables.BranchName}`	Empurrar, marcar
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`	Todos
`CommitMessage`	Corrigido um erro (tamanho máximo de 100 KB)	`#{SourceVariables.CommitMessage}`	Todos
`ConnectionArn`	arn:aws:codestar-connections:region ::conexão	`#{SourceVariables.ConnectionArn}`	Todos

Chave de variável	Valor de exemplo	Exemplo de sintaxe de variável	Tipo de gatilho
	/ *account-id* *connection-id*		
DestinationBranchName	main	#{SourceVariables.DestinationBranchName}	Solicitação pull
FullRepositoryName	username/GitHubRepo	#{SourceVariables.FullRepositoryName}	Todos
ProviderType	GitHub	#{SourceVariables.ProviderType}	Todos
PullRequestId	1	#{SourceVariables.PullRequestId}	Solicitação pull
PullRequestTitle	Adicionar novo recurso	#{SourceVariables.PullRequestTitle}	Solicitação pull
SourceBranchName	ramo de características	#{SourceVariables.SourceBranchName}	Solicitação pull
TagName	v1.0.0	#{SourceVariables.TagName}	Tag

GitHub variáveis de saída de ação GitHub (por meio do aplicativo OAuth) ação)

GitHub variáveis GitHub (por meio do aplicativo OAuth) ação)

Chave de variável	Valor de exemplo	Exemplo de sintaxe de variável
`AuthorDate`	2019-10-29 T03:32:21Z	`#{SourceVariables.AuthorDate}`
`BranchName`	main	`#{SourceVariables.BranchName}`
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Corrigido um erro (tamanho máximo de 100 KB)	`#{SourceVariables.CommitMessage}`
`CommitterDate`	2019-10-29 T03:32:21Z	`#{SourceVariables.CommitterDate}`
`CommitUrl`		`#{SourceVariables.CommitUrl}`
`RepositoryName`	meu GitHubRepo	`#{SourceVariables.RepositoryName}`

Variáveis de saída da ação do S3

Variáveis do S3

Chave de variável	Valor de exemplo	Exemplo de sintaxe de variável
`ETag`	example28be1c3	`#{SourceVariables.ETag}`

Chave de variável	Valor de exemplo	Exemplo de sintaxe de variável
`VersionId`	exampleta_IUQCv	`#{SourceVariables.VersionId}`

Ações com chaves variáveis configuradas pelo usuário

Para ações CodeBuild CloudFormation,, e Lambda, as chaves variáveis são configuradas pelo usuário.

Tópicos

- CloudFormation variáveis de saída da ação
- CodeBuild variáveis de saída da ação
- Variáveis de saída de ação do Lambda

CloudFormation variáveis de saída da ação

CloudFormation variables

Chave de variável	Exemplo de sintaxe de variável
Para CloudFormation ações, as variáveis são produzidas a partir de quaisquer valores designados na `Outputs` seção de um modelo de pilha. Observe que os únicos modos de CloudFormation ação que geram saídas são aqueles que resultam na criação ou atualização de uma pilha, como criação de pilha, atualizações de pilha e execução de conjuntos de alterações. Os modos de ação correspondentes que geram variáveis são: • CREATE_UPDATE • CHANGE_SET_EXECUTE • CHANGE_SET_REPLACE • REPLACE_ON_FAILURE	`#{DeployVariables.StackName}`

Chave de variável	Exemplo de sintaxe de variável
Para obter mais informações sobre esses modos de ação, consulte Referência da ação de implantação do CloudFormation. Para ver um tutorial que mostra como criar um pipeline com uma ação de CloudFormation implantação em um pipeline que usa variáveis CloudFormation de saída, consulteTutorial: criar um pipeline que usa variáveis de AWS CloudFormation ações de implantação.	

CodeBuild variáveis de saída da ação

CodeBuild variáveis

Chave de variável	Exemplo de sintaxe de variável
Para CodeBuild ações, as variáveis são produzidas a partir dos valores gerados pelas variáveis de ambiente exportadas. Configure uma variável de CodeBuild ambiente editando sua CodeBuild ação CodePipeline ou adicionando a variável de ambiente ao buildspec. Adicione instruções ao seu CodeBuild buildspec para adicionar a variável de ambiente na seção de variáveis exportadas. Consulte env/exported -variables no Guia do AWS CodeBuild usuário.	`#{BuildVariables.EnvVar}`

Variáveis de saída de ação do Lambda

Variáveis do Lambda

Chave de variável	Exemplo de sintaxe de variável
A ação Lambda produzirá como variáveis todos os pares de valores-chave incluídos na `outputVariables` seção da solicitação da PutJobSuccessResult API.	`#{TestVariables.testRunId}`

Chave de variável	Exemplo de sintaxe de variável
Para ver um tutorial com uma ação do Lambda que usa variáveis de uma ação upstream (CodeCommit) e gera variáveis de saída, consulte. Tutorial: Usar variáveis com ações de invocação do Lambda	

Trabalhar com padrões glob na sintaxe

Ao especificar os arquivos ou caminhos usados nos artefatos do pipeline ou nos locais de origem, você pode especificar o artefato dependendo do tipo de ação. Por exemplo, para a ação do S3, você especifica a chave do objeto do S3.

Para acionadores, você pode especificar filtros. Você pode usar padrões glob para especificar filtros. Veja os exemplos a seguir.

Quando a sintaxe é "glob", a representação String do caminho é combinada usando uma linguagem de padrões limitada com uma sintaxe que se assemelha a expressões regulares. Por exemplo:

- `*.java` Especifica um caminho que representa um nome de arquivo terminado em .java
- `*.*` Especifica nomes de arquivo que contêm um ponto
- `*.{java,class}` Especifica nomes de arquivo que terminam com .java ou .class
- `foo.?` Especifica nomes de arquivo que começam com foo. e uma extensão de um único caractere

As regras a seguir são usadas para interpretar padrões glob:

- Para especificar zero ou mais caracteres de um componente de nome até os limites do diretório, use `*`.
- Para especificar zero ou mais caracteres de um componente de nome que ultrapassa os limites do diretório, use `**`.
- Para especificar um caractere de um componente de nome, use `?`.
- Para realizar o escape de caracteres que não podem ser interpretados como caracteres especiais, use o caractere de barra invertida (\).
- Para especificar um único caractere de um conjunto de caracteres, use `[ ]`.
- Para especificar um único arquivo que esteja na raiz do local de compilação ou do local do repositório de origem, use `my-file.jar`.
- Para especificar um único arquivo em um subdiretório, use `directory/my-file.jar` ou `directory/subdirectory/my-file.jar`.
- Para especificar todos os arquivos, use `"**"`. O padrão glob `**` indica que corresponde a qualquer número de subdiretórios.

- Para especificar todos os arquivos e diretórios em um diretório chamado `directory`, use `"directory/**"`. O padrão glob `**` indica que corresponde a qualquer número de subdiretórios.
- Para especificar todos os arquivos em um diretório chamado `directory`, mas não em nenhum de seus subdiretórios, use `"directory/*"`.
- Em uma expressão de colchetes, os caracteres `*`, `?` e `\` são correspondentes entre si. O caractere de hífen () corresponderá a si mesmo se for o primeiro caractere dentro dos colchetes ou o primeiro caractere após o `!` quando você estiver fazendo uma negação.
- Os caracteres `{ }` são um grupo de subpadrões, e o grupo será correspondente se houver correspondência com qualquer subpadrão no grupo. Um caractere `","` é usado para separar os subpadrões. Grupos não podem ser aninhados.

Atualizar pipelines de sondagem para o método de detecção de alterações recomendado

Se você tiver um pipeline que usa sondagem para reagir às alterações na origem, poderá atualizá-lo para usar o método de detecção recomendado. Para ver um guia de migração com instruções para atualizar seus canais de pesquisa para usar o método recomendado de detecção de alterações com base em eventos, consulte. Migrar pipelines de sondagem para usar a detecção de alterações baseada em eventos

Atualizar uma ação de origem GitHub (via aplicativo OAuth) para uma ação de origem GitHub (via GitHub aplicativo)

Em AWS CodePipeline, há duas versões suportadas da ação de GitHub origem:

- Recomendado: a ação GitHub (via GitHub aplicativo) usa autenticação GitHub baseada em aplicativo apoiada por um recurso. CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com Ele instala um aplicativo AWS CodeStar Connections em sua GitHub organização para que você possa gerenciar o acesso em GitHub.
- Não recomendado: a ação GitHub (por meio do aplicativo OAuth) usa tokens OAuth para se autenticar GitHub e usa um webhook separado para detectar alterações. Este método não é mais recomendado.

> **Note**
>
> As conexões não estão disponíveis nas regiões Ásia-Pacífico (Hong Kong), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Melbourne), Ásia-Pacífico (Osaka), África (Cidade do Cabo), Oriente Médio (Bahrein), Oriente Médio (EAU), Europa (Espanha), Europa (Zurique), Israel (Tel Aviv) ou (). AWS GovCloud US-West Para fazer referência a outras ações disponíveis, consulte Integrações de produtos e serviços com CodePipeline. Para considerações sobre essa ação na região Europa (Milão), consulte a nota em CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.

Há algumas vantagens importantes em usar a ação GitHub (via GitHub aplicativo) em vez da ação GitHub (via aplicativo OAuth):

- Com as conexões, CodePipeline não são mais necessários aplicativos OAuth ou tokens de acesso pessoal para acessar seu repositório. Ao criar uma conexão, você instala um GitHub aplicativo que gerencia a autenticação no seu GitHub repositório e permite permissões no nível da organização. É necessário autorizar tokens do OAuth como usuário para acessar o repositório. Para obter mais informações sobre o OAuth-based GitHub acesso em contraste com o App-based GitHub acesso,

consultehttps://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps.

- Quando você gerencia GitHub (via GitHub aplicativo) ações na CLI ou CloudFormation não precisa mais armazenar seu token de acesso pessoal como segredo no Secrets Manager. Você não precisa mais referenciar dinamicamente o segredo armazenado em sua configuração de CodePipeline ação. Em vez disso, você adiciona o ARN da conexão à configuração de ação. Para obter um exemplo de configuração de ação, consulte CodeStarSourceConnection para Bitbucket Cloud GitHub, GitHub Enterprise Server e ações GitLab autogerenciadas GitLab.com.
- Ao criar um recurso de conexão para usar com sua ação GitHub (via GitHub App) em CodePipeline, você pode usar o mesmo recurso de conexão para associar outros serviços compatíveis, como o CodeGuru Reviewer, ao seu repositório.
- Em GitHub (via GitHub aplicativo), você pode clonar repositórios para acessar metadados do git em CodeBuild ações subsequentes, enquanto em GitHub (via aplicativo OAuth) você só pode baixar a fonte.
- Um administrador instala a aplicação nos repositórios da organização. Não é mais necessário rastrear tokens do OAuth que dependam da pessoa que criou o token.

Todas as aplicações instaladas em uma organização têm acesso ao mesmo conjunto de repositórios. Para alterar quem pode acessar cada repositório, modifique a política do IAM para cada conexão. Para ver um exemplo, consulte Exemplo: uma política de restrição de acesso para usar conexões com um repositório especificado.

Você pode usar as etapas deste tópico para excluir sua ação de origem GitHub (por meio do aplicativo OAuth) e adicionar uma ação de origem GitHub (via GitHub aplicativo) do CodePipeline console.

Tópicos

- Etapa 1: Substitua sua ação (por meio do aplicativo OAuth) GitHub
- Etapa 2: criar uma conexão com GitHub
- Etapa 3: Salve sua ação GitHub de origem

Etapa 1: Substitua sua ação (por meio do aplicativo OAuth) GitHub

Use a página de edição do pipeline para substituir sua ação (via aplicativo OAuth) por uma GitHub ação GitHub (via GitHub aplicativo).

Para substituir sua ação (por meio do aplicativo OAuth) GitHub

1. Faça login no CodePipeline console.
2. Selecione o pipeline e escolha Editar. Selecione Editar estágio no estágio de origem. É exibida uma mensagem recomendando que você atualize a ação.
3. Em Provedor de ações, escolha GitHub (via GitHub aplicativo).
4. Execute um destes procedimentos:

   - Em Conexão, se você ainda não tiver criado uma conexão com seu provedor, escolha Conectar GitHub a. Prossiga para a Etapa 2: Crie uma conexão com GitHub.
   - Em Conexão, se você já tiver criado uma conexão com seu provedor, escolha a conexão. Vá para a Etapa 3: Salvar a ação de origem para sua conexão.

Etapa 2: criar uma conexão com GitHub

Depois de escolher criar a conexão, a GitHub página Conectar a será exibida.

Para criar uma conexão com GitHub

1. Nas configurações de GitHub conexão, o nome da conexão é mostrado em Nome da conexão.

   Em GitHub Aplicativos, escolha uma instalação de aplicativo ou escolha Instalar um novo aplicativo para criar um.

   > **Note**
   >
   > Você instala uma aplicação para todas as suas conexões com um provedor específico. Se você já instalou o GitHub aplicativo, escolha-o e pule esta etapa.

2. Se a página de autorização for GitHub exibida, faça login com suas credenciais e escolha continuar.
3. Na página de instalação do aplicativo, uma mensagem mostra que o AWS CodeStar aplicativo está tentando se conectar à sua GitHub conta.

   > **Note**
   >
   > Você só instala o aplicativo uma vez para cada GitHub conta. Se você instalou a aplicação anteriormente, poderá escolher Configure (Configurar) para prosseguir para

uma página de modificação para a instalação da aplicação ou usar o botão Back (Voltar) para retornar ao console.

4. Na página Instalar AWS CodeStar, escolha Instalar.
5. Na GitHub página Conectar a, o ID de conexão da sua nova instalação é exibido. Selecione Conectar.

Etapa 3: Salve sua ação GitHub de origem

Conclua as atualizações na página Editar ação para salvar a nova ação de origem.

Para salvar sua ação GitHub de origem

1. Em Nome do repositório, escolha o nome do repositório de terceiros. Em Ramificação, insira a ramificação onde deseja que o pipeline detecte alterações de origem.

   Note

   No Repositório, digite `owner-name/repository-name` conforme mostrado neste exemplo:

   ```
   my-account/my-repository
   ```

2. Em Formato de artefato de saída, escolha o formato dos seus artefatos.

   - Para armazenar artefatos de saída da GitHub ação usando o método padrão, escolha CodePipeline default. A ação acessa os arquivos do GitHub repositório e armazena os artefatos em um arquivo ZIP no armazenamento de artefatos do pipeline.
   - Para armazenar um arquivo JSON que contém uma referência de URL ao repositório para que as ações downstream possam executar comandos Git diretamente, escolha Full clone (Clone completo). Esta opção só pode ser usada por ações downstream do CodeBuild .

     Se você escolher essa opção, precisará atualizar as permissões para sua função de serviço de CodeBuild projeto, conforme mostrado emAdicione CodeBuild GitClone permissões para conexões com o Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com. Para um tutorial que mostra como usar a opção Clone completo, consulte Tutorial: use o clone completo com uma fonte de GitHub pipeline.

3. Em Artefatos de saída, é necessário manter o nome do artefato de saída para essa ação, como `SourceArtifact`. Escolha Concluído para fechar a página Editar ação.
4. Escolha Concluído para fechar a página de edição do estágio. Escolha Salvar para fechar a página de edição do pipeline.

Cotas em AWS CodePipeline

CodePipeline tem cotas para o número de pipelines, estágios, ações e webhooks que uma AWS conta pode ter em cada região. AWS

As cotas a seguir são aplicadas por região e podem ser aumentadas. Até duas semanas podem ser necessárias para o processamento das solicitações de aumento da cota.

Recurso	Padrão
O período de tempo antes de uma ação expirar (Estes são tempos limite configuráveis. Veja a tabela a seguir para tempos limite não configurá veis)	CloudFormation ação de implantação: 3 dias CodeDeploy e ações de implantação do CodeDeploy ECS (blue/green): 5 dias AWS Lambda ação de invocação: 24 horas **Note** Enquanto a ação estiver em execução, entre em contato CodePipeline periodicamente com a Lambda para obter um status. A função do Lambda responde com o status da execução da ação: bem-sucedida, falha ou em andamento. Se a função do Lambda não enviar nenhuma resposta após 20 minutos, a ação expirará. Se, durante os 20 minutos, a função Lambda responder que a ação ainda está em andamento, CodePipeline reinicia o cronômetr o de 20 minutos e tenta novamente . Se não for bem-sucedido após 24 horas, CodePipeline define o estado

Recurso	Padrão
	da ação de invocação do Lambda como Falha. O Lambda tem um tempo limite separado para as funções do Lambda que não está relacionado ao tempo limite da ação. CodePipeline Ação de implantação do Amazon S3: 90 minutos **Note** Se o upload para o S3 expirar durante a implantação de um arquivo ZIP grande, a ação falhará com um erro de tempo limite. Tente dividir o arquivo ZIP em arquivos menores. Tempo limite padrão do nível de conta de ação de aprovação manual: 7 dias **Note** O tempo limite padrão para a ação de aprovação manual pode ser substituído para uma ação específica no pipeline e é configurável a até 86.400 minutos (60 dias) com um valor mínimo de 5 minutos. Para obter mais informações, consulte

Recurso	Padrão
	ActionDeclaration na Referência de APIs do CodePipeline . Quando configurado, esse tempo limite é aplicado para a ação. Caso contrário, o padrão de nível de conta é usado. Todas as outras ações: 1 hora **Note** O tempo limite da ação de implantação do Amazon ECS é configurável até uma hora (o tempo limite padrão).
Número máximo de pipelines totais por região em uma conta AWS	1000 **Note** Os pipelines configurados para a sondagem ou a detecção de alterações baseada em eventos são contados para essa cota.

Recurso	Padrão
Número máximo de pipelines definidos para pesquisa de alterações de origem, por região da AWS	300 **Note** Essa é uma cota fixa e não pode ser alterada. Se você atingir o limite de pesquisa de pipelines, ainda poderá configurar pipelines adicionais que usam a detecção de alterações baseada em eventos. Para obter mais informações, consulte Ações de origem e métodos de detecção de alterações.[1]
Número máximo de webhooks por região em uma conta AWS	300
Número de ações personalizadas por região em uma AWS conta	50

[1]Com base no provedor de origem, use as instruções a seguir para atualizar os pipelines de sondagem para usar a detecção de alteração baseada em eventos:

- Para atualizar uma ação CodeCommit de origem, consulteMigre canais de pesquisa (CodeCommit ou fonte do Amazon S3) (console).
- Para atualizar uma ação de origem do Amazon S3, consulte Migre canais de pesquisa (CodeCommit ou fonte do Amazon S3) (console).
- Para atualizar uma ação GitHub de origem, consulteMigre canais de pesquisa para webhooks (por meio do aplicativo OAuthGitHub) (ações de origem) (console).

As cotas a seguir AWS CodePipeline se aplicam à disponibilidade da região, às restrições de nomenclatura e aos tamanhos de artefatos permitidos. Essas cotas são fixadas e não podem ser alteradas.

Para obter uma lista dos endpoints de CodePipeline serviço para cada região, consulte AWS CodePipeline endpoints e cotas na AWS Referência geral.

Para obter informações sobre os requisitos estruturais, consulte CodePipeline referência de estrutura de tubulação.

AWS Regiões onde você pode criar um pipeline	Leste dos EUA (Norte da Virgínia) Leste dos EUA (Ohio) Oeste dos EUA (N. da Califórnia) Oeste dos EUA (Oregon) Canadá (Central) Europa (Frankfurt) Europa (Zurique)* Europa (Estocolmo) Europa (Milão)* Europa (Espanha) * Europa (Irlanda) Europa (Londres) Europa (Paris) Israel (Tel Aviv) * África (Cidade do Cabo)* Ásia-Pacífico (Hong Kong)* Ásia-Pacífico (Mumbai) Ásia-Pacífico (Hyderabad) * Ásia-Pacífico (Tóquio)

	Ásia-Pacífico (Seul) Asia Pacific (Osaka) Ásia-Pacífico (Singapura) Ásia-Pacífico (Sydney) Ásia-Pacífico (Jacarta) * Ásia-Pacífico (Melbourne) * Ásia-Pacífico (Malásia) * Ásia-Pacífico (Nova Zelândia) * América do Sul (São Paulo) Oriente Médio (Bahrein)* Oriente Médio (EAU) * AWS GovCloud (US-West) AWS GovCloud (US-East)
Caracteres permitidos no nome de uma ação	Os nomes de ações não pode exceder 100 caracteres. Os caracteres permitidos incluem: Letras minúsculas `a` a `z`, inclusive. Letras maiúsculas `A` a `Z`, inclusive. Números `0` a 9 inclusive. Os caracteres especiais `.` (ponto), `@` (arroba), `-` (sinal de menos) e `_` (sublinhado). Outros caracteres, como espaços, não são permitidos.

Caracteres permitidos em tipos de ações	O nomes dos tipos de ações não podem exceder 25 caracteres. Os caracteres permitidos incluem: Letras minúsculas de a a z, inclusive. Letras maiúsculas de A a Z, inclusive. Números de 0 a 9, inclusive. Caracteres especiais . (ponto), @ (arroba), - (sinal de menos) e _ (sublinhado). Outros caracteres, como espaços, não são permitidos.
Caracteres permitidos nos nomes de artefatos	Os nomes de artefatos não pode exceder 100 caracteres. Os caracteres permitidos incluem: Letras minúsculas a a z, inclusive. Letras maiúsculas A a Z, inclusive. Números 0 a 9 inclusive. Caracteres especiais - (sinal de subtração) e _ (sublinhado). Outros caracteres, como espaços, não são permitidos.

Caracteres permitidos em nomes de ações de parceiros	Os nomes de ações de parceiros devem seguir as mesmas convenções e restriçõe s de nomeação que outros nomes de ações no CodePipeline. Especificamente, não podem exceder 100 caracteres. Os caracteres permitidos incluem: Letras minúsculas de a a z, inclusive. Letras maiúsculas de A a Z, inclusive. Números de 0 a 9, inclusive. Caracteres especiais . (ponto), @ (arroba), - (sinal de menos) e _ (sublinhado). Outros caracteres, como espaços, não são permitidos.
Caracteres permitidos no nome de um pipeline	Os nomes de pipelines não podem exceder 100 caracteres. Os caracteres permitidos incluem: Letras minúsculas de a a z, inclusive. Letras maiúsculas de A a Z, inclusive. Números de 0 a 9, inclusive. Caracteres especiais . (ponto), @ (arroba), - (sinal de menos) e _ (sublinhado). Outros caracteres, como espaços, não são permitidos.

Caracteres permitidos no nome de um estágio	Os nomes de estágio não podem exceder 100 caracteres. Os caracteres permitidos incluem: Letras minúsculas de a a z, inclusive. Letras maiúsculas de A a Z, inclusive. Números de 0 a 9, inclusive. Caracteres especiais . (ponto), @ (arroba), - (sinal de menos) e _ (sublinhado). Outros caracteres, como espaços, não são permitidos.
O período de tempo antes de uma ação expirar	CodeBuild ação de construção: 36 horas Ação de teste: 8 horas Ações personalizadas: 24 horas Ação de invocação do Step Functions: 7 dias Tempo limite de compilação para a ação Commands: 55 minutos
Tamanho máximo da chave de configuração da ação (por exemplo, as chaves de configuração do CodeBuild são `ProjectName` , `PrimarySource` e `EnvironmentVariables`)	50 caracteres
Tamanho máximo do valor da configuração da ação (por exemplo, o valor da `RepositoryName` configuração na configuração da CodeCommit ação deve ter menos de 1000 caracteres): `"RepositoryName": "my-repo-name-less-than-1000-characters" )`	1.000 caracteres

Número máximo de ações por pipeline	1000
Número máximo de execuções simultâneas de pipeline por pipeline (modo QUEUED|PARALLEL)	50
Número máximo de execuções de ações simultâneas por execução de pipeline no modo PARALLEL	5
Número máximo de arquivos para um objeto do Amazon S3	100.000
Número máximo de ações paralelas em um estágio	100
Número máximo de ações de sequenciais em um estágio	100
Tamanho máximo de artefatos em um estágio de origem	Artefatos armazenados nos buckets do Amazon S3: 7 GB Artefatos armazenados em CodeCommit nossos GitHub repositórios: 1 GB Exceção: se você estiver usando AWS Elastic Beanstalk para implantar aplicativos, o tamanho máximo do artefato é sempre 512 MB. Exceção: se você estiver usando CloudFormation para implantar aplicativos, o tamanho máximo do artefato é sempre 256 MB. Exceção: se você estiver usando a ação `CodeDeployToECS` para implantar aplicativos, o tamanho máximo dos artefatos será sempre de 3 MB.

O tamanho máximo do arquivo JSON de definições de imagem usado nos pipelines que implantam contêineres e imagens do Amazon ECS	100 KB
Tamanho máximo dos artefatos de entrada para ações CloudFormation	256 MB
Tamanho máximo de artefatos de entrada para a ação do `CodeDeployToECS`	3 MB
Tamanho máximo de artefatos de entrada para a ação do `Step Functions`	A ação do Step Functions é executada no Lambda e, por isso, as cotas de tamanho de artefato são iguais às cotas de tamanho de artefato para funções do Lambda. Para ter mais informações, consulte Cotas do Lambda no Guia do desenvolvedor do Lambda.
O tamanho máximo do objeto JSON que pode ser armazenado na propriedade `Parameter 0verrides` .	Para uma ação de CodePipeline implantaç ão com CloudFormation como provedor, a `ParameterOverrides` propriedade é usada para armazenar um objeto JSON que especifica valores para o arquivo de configura ção do CloudFormation modelo. Há um limite de tamanho máximo de 1 KB para o objeto JSON que pode ser armazenado na proprieda de `ParameterOverrides` .
Número de ações em um estágio	No mínimo 1, no máximo 50
Número de artefatos permitidos para cada ação	Para o número de artefatos de entrada e saída permitidos para cada ação, consulte Artefatos de entrada e saída válidos para cada tipo de ação
Número de meses em que as informações do histórico de execução do pipeline são retidas	12

Número de estágios em um pipeline	No mínimo 2, no máximo 50
Tags do pipeline	As tags diferenciam letras maiúsculas de minúsculas. Máximo de 50 por recurso.
Nomes de chaves de tag do pipeline	Qualquer combinação de letras, números, espaços e caracteres permitidos em Unicode UTF-8 entre 1 e 128 caracteres. Os caracteres permitidos são + - = . _ : / @ Os nomes de chaves de tag devem ser exclusivos, e cada chave só pode ter um valor. Uma tag não pode: • comece com AWS: • consistir apenas de espaços • terminar com um espaço • conter emojis ou um dos caracteres a seguir: ? ^ * [\ ~ ! # $ % & * ( ) > < | " '
Valores de tags do pipeline	Qualquer combinação de letras, números, espaços e caracteres permitidos em Unicode UTF-8 entre 1 e 256 caracteres. Os caracteres permitidos são + - = . _ : / @ Uma chave pode ter apenas um valor, mas várias chaves podem ter o mesmo valor. Uma tag não pode: • comece com AWS: • consistir apenas de espaços • terminar com um espaço • conter emojis ou um dos caracteres a seguir: ? ^ * [\ ~ ! # $ % & * ( ) > < | " '

Acionadores	Há um máximo de 50 gatilhos em uma definição de pipeline no `push` e na configuração do `pull request`. Há no máximo três filtros por push de gatilho e solicitação pull de gatilho. **Note** Não são permitidas duplicatas de filtros na mesma matriz de tipo de evento. É possível adicionar até 8 padrões de inclusão e 8 padrões de exclusão, bem como ramificações e caminhos de arquivos, para cada tipo de evento (push, solicitação pull). Todos os tipos de caracteres são permitidos em patternvalues. Nos padrões de inclusão e exclusão, há um tamanho máximo de 255 caracteres. Nos nomes de tags, há um tamanho máximo de 255 caracteres. O tamanho máximo da matriz de `triggers` não deve ultrapassar 200 KB

Filtros de gatilho	Caminhos de arquivo: • Quantidade de padrões: é possível adicionar até 8 padrões de inclusão e 8 de exclusão. • Tamanho do padrão: cada padrão de inclusão ou exclusão pode ter até 255 caracteres. Ramificações: • Quantidade de padrões: é possível adicionar até 8 padrões de inclusão e 8 de exclusão. • Tamanho do padrão: cada padrão de inclusão ou exclusão pode ter até 255 caracteres. Solicitações pull: Ramificações: • Quantidade de padrões: é possível adicionar até 8 padrões de inclusão e 8 de exclusão. • Tamanho do padrão: cada padrão de inclusão ou exclusão pode ter até 255 caracteres.
Exclusividade de nomes	Em uma única AWS conta, cada funil que você cria em uma AWS região deve ter um nome exclusivo. Você pode reutilizar nomes de pipelines em diferentes regiões da AWS . Os nomes dos estágios em um pipeline devem ser exclusivos. Os nomes das ações em um estágio devem ser exclusivos.

Cotas para variáveis de saída e namespaces	Há um limite de tamanho máximo de 122880 bytes para todas as variáveis de saída combinadas para uma ação específica. Há um limite de tamanho máximo de 100 KB para a configuração de ação resolvida total para uma ação específica. Os nomes de variáveis de saída diferenciam minúsculas de maiúsculas. Os namespaces diferenciam minúsculas de maiúsculas. Os caracteres permitidos incluem: • Letras minúsculas de a a z, inclusive. • Letras maiúsculas de A a Z, inclusive. • Números de 0 a 9, inclusive. • Caracteres especiais ^ (circunflexo), @ (arroba), - (sinal de menos), _ (sublinhado), [(colchete esquerdo),] (colchete direito), * (asterisco), $ (cifrão). Outros caracteres, como espaços, não são permitidos.

Cotas para variáveis no nível do pipeline	Há, no máximo, 50 variáveis no nível do pipeline por pipeline. Os nomes das variáveis no nível do pipeline devem ter: • Um tamanho máximo de 128 caracteres • Letras minúsculas de a a z, inclusive. • Letras maiúsculas de A a Z, inclusive. • Números de 0 a 9, inclusive. • Caracteres especiais `@\-_]+` Outros caracteres, como espaços, não são permitidos. Para valores de variáveis, há um tamanho máximo de 1000 caracteres Para valores de variáveis, todos os caracteres são permitidos. Para valores de variáveis, há um tamanho máximo de 200 caracteres

* Você deve habilitar esta região para que possa utilizá-la.

Apêndice A: GitHub (por meio do aplicativo OAuth) ações de origem

Este apêndice fornece informações sobre a ação GitHub (por meio do aplicativo OAuth) em. CodePipeline

> **Note**
>
> Embora não seja recomendável usar a ação GitHub (por meio do aplicativo OAuth), os pipelines existentes com a ação GitHub (por meio do aplicativo OAuth) continuarão funcionando sem nenhum impacto. Para um pipeline com uma ação GitHub (via aplicativo OAuth), CodePipeline usa OAuth-based tokens para se conectar ao seu GitHub repositório. Por outro lado, a GitHub ação (via GitHub App) usa um recurso de conexão para associar AWS recursos ao seu GitHub repositório. O recurso de conexão usa tokens baseados em aplicativos para estabelecer conexões. Para obter mais informações sobre como atualizar seu funil para a GitHub ação recomendada que usa uma conexão, consulteAtualizar uma ação de origem GitHub (via aplicativo OAuth) para uma ação de origem GitHub (via GitHub aplicativo). Para obter mais informações sobre o OAuth-based GitHub acesso em contraste com o GitHub acesso baseado em aplicativos, consulte. https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps

Para integrar com GitHub, CodePipeline usa um aplicativo GitHub OAuth para seu pipeline. CodePipelineusa webhooks para gerenciar a detecção de alterações em seu pipeline com a ação de origem GitHub (por meio do aplicativo OAuth).

> **Note**
>
> Ao configurar uma ação de origem GitHub (via GitHub aplicativo) em CloudFormation, você não inclui nenhuma informação de GitHub token nem adiciona um recurso de webhook. Você configura um recurso de conexões conforme mostrado em AWS::CodeStarConnections: :Connection no Guia do CloudFormation usuário.

Essa referência contém as seguintes seções para a ação GitHub (por meio do aplicativo OAuth):

- Para obter informações sobre como adicionar uma ação de origem e um webhook GitHub (por meio do aplicativo OAuth) a um pipeline, consulte. Adicionar uma ação de GitHub origem (por meio do aplicativo OAuth)
- Para obter informações sobre os parâmetros de configuração e exemplos de YAML/JSON trechos de uma ação de origem GitHub (por meio do aplicativo OAuth), consulte. GitHub (via aplicativo OAuth) referência de ação de origem

Important

Ao criar CodePipeline webhooks, não use suas próprias credenciais nem reutilize o mesmo token secreto em vários webhooks. Para garantir um nível ideal de segurança, gere um token secreto exclusivo para cada webhook que você criar. O token secreto é uma string arbitrária que você fornece, GitHub usada para computar e assinar as cargas do webhook enviadas para CodePipeline, a fim de proteger a integridade e a autenticidade das cargas do webhook. Usar as próprias credenciais ou reutilizar o mesmo token em vários webhooks pode levar a vulnerabilidades de segurança.

Note

Se um token secreto tiver sido fornecido, ele será redigido na resposta.

Tópicos

- Adicionar uma ação de GitHub origem (por meio do aplicativo OAuth)
- GitHub (via aplicativo OAuth) referência de ação de origem

Adicionar uma ação de GitHub origem (por meio do aplicativo OAuth)

Você adiciona ações de origem GitHub (por meio do aplicativo OAuth) por meio de CodePipeline :

- Usando o CodePipeline console Create Pipeline Wizard (Criar um pipeline personalizado (console)) ou a página de ação Editar para escolher a opção GitHub de provedor. O console cria um webhook que aciona seu pipeline quando a origem é alterada.

- Usando a CLI para adicionar a configuração da ação `GitHub` e criando recursos adicionais por meio do(a):
  - Exemplo de configuração da ação `GitHub` em GitHub (via aplicativo OAuth) referência de ação de origem para criar a ação conforme mostrado em Criar um pipeline (CLI).
  - Desabilitação das verificações periódicas e criação da detecção de alterações manualmente, porque o método de detecção de alterações assume como padrão o acionamento do pipeline pesquisando a origem. Você migra seu canal de pesquisa para webhooks para ações GitHub (por meio do aplicativo OAuth).

GitHub (via aplicativo OAuth) referência de ação de origem

> **Note**
>
> Embora não seja recomendável usar a ação GitHub (por meio do aplicativo OAuth), os pipelines existentes com a ação GitHub (por meio do aplicativo OAuth) continuarão funcionando sem nenhum impacto. Para um pipeline com uma ação de origem GitHub (via aplicativo OAuth), CodePipeline usa OAuth-based tokens para se conectar ao seu GitHub repositório. Por outro lado, a nova GitHub ação (via GitHub App) usa um recurso de conexão para associar AWS recursos ao seu GitHub repositório. O recurso de conexão usa tokens baseados em aplicativos para estabelecer conexões. Para obter mais informações sobre como atualizar seu funil para a GitHub ação recomendada que usa uma conexão, consulteAtualizar uma ação de origem GitHub (via aplicativo OAuth) para uma ação de origem GitHub (via GitHub aplicativo).

Aciona o pipeline quando uma nova confirmação é feita no GitHub repositório e na ramificação configurados.

Para integrar com GitHub, CodePipeline usa um aplicativo OAuth ou um token de acesso pessoal para seu funil. Se você usar o console para criar ou editar seu pipeline, CodePipeline crie um GitHub webhook que inicia seu funil quando ocorre uma alteração no repositório.

Você já deve ter criado uma GitHub conta e um repositório antes de conectar o pipeline por meio de uma GitHub ação.

Se você quiser limitar o acesso aos CodePipeline repositórios, crie uma GitHub conta e conceda à conta acesso somente aos repositórios com os quais você deseja se integrar. CodePipeline Use

essa conta ao configurar o uso CodePipeline de GitHub repositórios para estágios de origem em pipelines.

Para obter mais informações, consulte a documentação do GitHub desenvolvedor no GitHub site.

Tópicos

- Tipo de ação
- Parâmetros de configuração
- Input artifacts (Artefatos de entrada)
- Artefatos de saída
- Variáveis de saída
- Declaração de ação (GitHub exemplo)
- Conectando-se a GitHub (OAuth)
- Consulte também

Tipo de ação

- Categoria: `Source`
- Proprietário: `ThirdParty`
- Fornecedor: `GitHub`
- Versão: `1`

Parâmetros de configuração

Proprietário

Obrigatório: Sim

O nome do GitHub usuário ou da organização que possui o GitHub repositório.

Repositório

Obrigatório: Sim

O nome do repositório onde as alterações de origem devem ser detectadas.

Ramificação

Obrigatório: Sim

O nome da ramificação onde as alterações de origem devem ser detectadas.

OAuthToken

Obrigatório: Sim

Representa o token de GitHub autenticação que CodePipeline permite realizar operações no seu GitHub repositório. A entrada é sempre exibida como uma máscara de quatro asteriscos. Representa um dos seguintes valores:

- Quando você usa o console para criar o pipeline, CodePipeline usa um token OAuth para registrar a GitHub conexão.
- Ao usar o AWS CLI para criar o pipeline, você pode passar seu token de acesso GitHub pessoal nesse campo. Substitua os asteriscos (****) pelo seu token de acesso pessoal copiado. GitHub Quando `get-pipeline` é executado para visualizar a configuração da ação, a máscara de quatro asteriscos é exibida para esse valor.
- Ao usar um CloudFormation modelo para criar o pipeline, você deve primeiro armazenar o token como um segredo em AWS Secrets Manager. Inclua o valor desse campo como uma referência dinâmica ao segredo armazenado no Secrets Manager, como `{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}`.

Para obter mais informações sobre GitHub escopos, consulte a Referência da API do GitHub desenvolvedor no GitHub site.

PollForSourceChanges

Obrigatório: não

`PollForSourceChanges`controla se CodePipeline pesquisa o GitHub repositório em busca de alterações na fonte. Recomendamos que você use webhooks para detectar alterações na origem. Para obter mais informações sobre como configurar webhooks, consulte Migre canais de pesquisa para webhooks (por meio do aplicativo OAuthGitHub) (ações de origem) (CLI) ou Atualize pipelines para eventos push GitHub (por meio do aplicativo OAuth) (ações de origem) (CloudFormation modelo).

> **Important**
>
> Quando planejar configurar webhooks, você deve definir `PollForSourceChanges` como `false` para evitar execuções duplicadas do pipeline.

Os valores válidos para esse parâmetro:

- `True`: Se definido, CodePipeline pesquisa seu repositório em busca de alterações na fonte.

  > **Note**
  >
  > Se você omitir`PollForSourceChanges`, o CodePipeline padrão é pesquisar alterações na fonte em seu repositório. Esse comportamento é o mesmo de quando o `PollForSourceChanges` está definido como `true`.

- `False`: Se definido, CodePipeline não pesquisa seu repositório em busca de alterações na fonte. Use essa configuração quando você planejar configurar um webhook para detectar alterações na origem.

Input artifacts (Artefatos de entrada)

- Número de artefatos: `0`
- Descrição: os artefatos de entrada não se aplicam a esse tipo de ação.

Artefatos de saída

- Número de artefatos: `1`
- Descrição: O artefato de saída desta ação é um arquivo ZIP que contém o conteúdo do repositório e ramificação configurados na confirmação especificada como a revisão de origem para a execução do pipeline. Os artefatos gerados a partir do repositório são os artefatos de saída da ação. GitHub O ID de confirmação do código-fonte é exibido CodePipeline como a revisão da fonte para a execução do pipeline acionado.

Variáveis de saída

Quando configurada, essa ação produz variáveis que podem ser referenciadas pela configuração de ação de uma ação downstream no pipeline. Esta ação produz variáveis que podem ser visualizadas como variáveis de saída, mesmo que a ação não tenha um namespace. Configure uma ação com um namespace a fim de disponibilizar as variáveis para a configuração de ações downstream.

Para obter mais informações sobre variáveis em CodePipeline, consulteReferência de variáveis.

CommitId

O ID de GitHub confirmação que acionou a execução do pipeline. Os IDs de confirmação são o SHA completo da confirmação.

CommitMessage

A mensagem da descrição, se houver, associada à confirmação que acionou a execução do pipeline.

CommitUrl

O endereço URL da confirmação que acionou o pipeline.

RepositoryName

O nome do GitHub repositório em que a confirmação que acionou o pipeline foi feita.

BranchName

O nome da ramificação do GitHub repositório em que a alteração na fonte foi feita.

AuthorDate

A data em que a confirmação foi criada, no formato de timestamp.

CommitterDate

A data em que a confirmação foi confirmada, no formato de timestamp.

Declaração de ação (GitHub exemplo)

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"
        }
    ]
},
```

Conectando-se a GitHub (OAuth)

Na primeira vez que você usa o console para adicionar um GitHub repositório a um pipeline, é solicitado que você autorize o CodePipeline acesso aos seus repositórios. O token exige os seguintes GitHub escopos:

- O escopo `repo`, que é usado para se obter o controle total para ler e efetuar pull de artefatos de repositórios públicos e privados para um pipeline.
- O escopo `admin:repo_hook`, que é usado para se obter o controle total dos ganchos do repositório.

Ao usar a CLI ou um CloudFormation modelo, você deve fornecer o valor de um token de acesso pessoal que você já criou. GitHub

Consulte também

Os recursos relacionados a seguir podem ajudar você à medida que trabalha com esta ação.

- Referência de recursos para o Guia AWS CloudFormation do Usuário AWS::CodePipeline: :Webhook — Isso inclui definições de campo, exemplos e trechos do recurso em. CloudFormation
- Referência de recursos para o Guia AWS CloudFormation do usuário AWS::CodeStar:: GitHubRepository — Isso inclui definições de campo, exemplos e trechos do recurso em. CloudFormation
- Tutorial: crie um pipeline que crie e teste seu aplicativo Android com AWS Device Farm— Este tutorial fornece um exemplo de arquivo buildspec e um aplicativo de amostra para criar um pipeline com uma fonte. GitHub Ele cria e testa um aplicativo Android com CodeBuild e. AWS Device Farm

AWS CodePipeline Histórico do documento do Guia do Usuário

A tabela a seguir descreve as mudanças importantes em cada versão do Guia CodePipeline do usuário. Para receber notificações sobre atualizações dessa documentação, é possível inscrever-se em um feed RSS.

- Versão da API: 09-07-2015
- Última atualização da documentação: 23 de maio de 2025

Alteração	Descrição	Data
Documentação corrigida do comportamento do filtro de gatilho de tubulação	Foi corrigida a documenta ção dos exemplos de filtros de gatilho para descrever com precisão a relação entre inclusões e exclusões. As exclusões têm precedência sobre as inclusões para filtros de ramificação e caminho de arquivo. Para obter mais informações, consulte Exemplos de filtros de gatilho.	16 de março de 2026
Support adicionado para o Azure DevOps	Support adicionado para configurar conexões de AWS recursos para interagir com o Azure DevOps. Para obter mais informações, consulte Conexões para o Azure DevOps.	5 de agosto de 2025
Documentação atualizada para o arquivo de especific	Novas informações adicionad as sobre como usar o arquivo de especificação da implantaç	16 de maio de 2025

ação de implantação da ação de implantação do EC2	ão para a ação de implantaç ão do EC2 com opções de configuração e exemplos atualizados. Consulte a página de referência da ação na Referência de ação do Amazon EC2.	
Documentação adicionada para a ação de implantação do Lambda	Suporte adicionado para a nova ação de implantação do Lambda com opções de configuração e exemplos atualizados. Consulte a página de referência da ação na referência da ação de implantação do AWS Lambda.	16 de maio de 2025
Adicionar campos à ação Comandos para tipo de computação, tipo de ambiente e tipo de variável do ambiente	Informações adicionadas sobre novos campos que permitem a você especificar um ambiente de compilaçã o e um tipo de computaçã o para a ação Comandos. Um novo campo de tipo para variáveis de ambiente também foi adicionado para a ação Comandos. Veja a página de referência da ação em CodePipeline Comands action.	8 de maio de 2025

Novo tópico para documentar a política do perfil de serviço padrão	Foram adicionadas informações sobre a política mínima de função de serviço e atualizamos a tabela de links para permissões adicionais de função de serviço para cada ação em CodePipeline. Consulte a nova página de referência da regra na política CodePipeline de funções de serviço.	26 de março de 2025
Nova ação invocar `CodePipeline`	Informações adicionadas sobre a nova ação invocar `CodePipeline` . Consulte a página de referência da ação em CodePipeline invoke action reference.	14 de março de 2025
Nova regra `CodeBuild`	Informações adicionais sobre a nova regra `CodeBuild` disponível para executar projetos de compilação como uma regra para condições de estágio. Consulte a nova página de referência da CodeBuild regra na referência da regra.	14 de março de 2025

Compartilhando conexões em AWS CodeConnections	Para pipelines com conexões usando AWS CodeConnections, você pode usar conexões configuradas para compartilhamento entre Contas da AWS. Você pode configurar conexões compartilhadas no AWS Resource Access Manager. Para obter mais informações, consulte Usar uma conexão compartilhada com outra Conta da AWS.	6 de março de 2025
EventBridge entrada de transformação de entrada para revisões de origem	Para pipelines com CodeCommit fontes do Amazon ECR e do Amazon S3, você pode EventBridge usar a entrada de transformação de entrada para revisões de origem, que `revisionValue` é derivada da variável de evento de origem para sua chave de objeto (S3), commit () ou ID de imagem CodeCommit (ECR). Veja ações e EventBridge recursos de origem do Amazon ECR, Conectando-se às ações de origem do Amazon S3 com uma fonte habilitada para eventos CodeCommit e ações de origem e. EventBridge	3 de março de 2025

Crie uma substituição de especificação para a ação AWS CodeBuild	Você pode optar por substituir a especificação de construção da CodeBuild ação e, em vez disso, inserir comandos diretamente. Consulte a página de referência da ação na referência da ação de compilação e teste do AWS CodeBuild. Consulte também Criar um pipeline, estágios e ações.	3 de março de 2025
Nova ação de implantação do EC2	Informações adicionadas sobre a nova ação de implantação do EC2. Consulte a página de referência da ação na Referência de ação do Amazon EC2. Para ver um tutorial, consulte Tutorial: Implantar em instâncias do EC2 com CodePipeline.	21 de fevereiro de 2025
Nova ação de implantação do EKS	Informações adicionadas sobre a nova ação de implantação do EKS. Consulte a página de referência da ação na ação de implantação do EKS. Para ver um tutorial, consulte Tutorial: Implante no Amazon EKS com CodePipeline.	20 de fevereiro de 2025
Novas CloudWatch métricas e dimensões para CodePipeline	Foram adicionadas métricas para pipelines nas CloudWatch métricas. Veja CodePipeline CloudWatch as métricas.	13 de fevereiro de 2025

Nova regra `Commands`	Informações adicionais sobre a nova regra `Commands` disponível para executar comandos de shell como uma regra para condições de estágio. Consulte a nova página de referência da regra em Referência da regra Commands.	17 de dezembro de 2024
Informações de referência e exemplos expandidos para gatilhos	Descrições adicionais para filtros de eventos pull request por provedor em Eventos pull request para gatilhos por provedor. Exemplos expandidos adicionados para gatilhos com informações mais detalhadas sobre inclusões e exclusões em eventos push e eventos pull request em Eventos pull request para gatilhos por provedor. Informações de referência JSON adicionais incluídas sobre inclusões e exclusões em Gatilhos.	17 de dezembro de 2024
Novas ações no catálogo de ações	Agora você pode usar as ações `ECRBuildAndPublish` e `InspectorScan`. Para obter mais informações, consulte as páginas de referência ECRBuildAndPublishe de InspectorScanações.	22 de novembro de 2024

Nova configuração automática para repetição de estágio em caso de falha	É possível configurar um estágio para repetir automatic amente um estágio com falha ou ações que falharam dentro do estágio. Para ter mais informações, consulte Configure a stage for automatic retry on failure.	15 de outubro de 2024
Novo resultado `Skip` para condições de entrada	Adição de informações sobre o resultado `Skip` disponíve l para as condições de entrada. Você pode usar as regras `VariableCheck` e `LambdaInvoke` com essa configuração. Veja as etapas em Creating Entry condition s with Skip result. Para conferir uma lista de considera ções sobre condições com resultados Ignorar, consulte Considerations for results configured for stage condition s.	15 de outubro de 2024
Novas etapas do console para criar pipelines usando modelos estáticos	No CodePipeline console, você pode usar um novo assistente de criação de pipeline para escolher entre vários modelos estáticos para gerar recursos de pipeline CloudFormation. Para ter mais informações, consulte Create a pipeline from static templates.	9 de outubro de 2024

Nova ação `Commands`	Adição de informações sobre a nova ação `Commands` disponível para executar comandos de shell como uma ação no pipeline. Consulte a nova página de referênci a da ação em Ação de comandos e as permissõe s da função de serviço em Adicionar permissões à função CodePipeline de serviço. Para conferir um tutorial, consulte Tutorial: Create a pipeline that runs commands with compute.	3 de outubro de 2024
Nova regra `VariableCheck` para condições	Adição de informações sobre a regra `VariableC heck` para condições de estágio. Veja a nova página de referência da regra em VariableCheck. Para conferir um tutorial, consulte Tutorial: Create a variable check rule for a pipeline as an entry condition.	27 de setembro de 2024

Atualizações nas conexões para GitHub	Foram adicionadas informaçõ es sobre o uso do token de acesso GitHub do usuário com conexões para GitHub (ações GitHub V2). Os tokens de acesso do usuário são usados com CodeBuild projetos. Consulte GitHub conexões e o tutorial: Use o clone completo com uma fonte de GitHub pipeline.	16 de setembro de 2024
Atualizações para reestruturar a referência JSON do pipeline e o índice do guia	O guia foi reestruturado, incluindo algumas alterações nos títulos das seções, para melhorar a usabilidade das seções de referência e tarefas.	16 de agosto de 2024
Atualização do campo de token secreto na resposta para ações `PutWebhook` e `ListWebhooks`	O campo de token secreto para as ações `PutWebhook` e `ListWebhooks` foi atualizado. Se um token secreto tiver sido fornecido, ele será redigido na resposta. Veja as notas adicionadas ao Apêndice A: ações de origem da GitHub versão 1. Para atualizações relacionadas no Guia CodePipeline da API, consulte PutWebhookListWebhookse.	6 de agosto de 2024

Adição de novo conteúdo para regras e condições de estágio	Agora você pode configurar condições de estágio e regras para pipelines do tipo V2. Consulte Concepts, How do stage conditions work? e Configure conditions for a stage. Foi adicionado um capítulo que fornece informações de referência para as regras. Veja a referência da CodePipeline regra.	30 de julho de 2024
Adição de novas informações de referência para tipos de pipeline e recursos relacionados	Um novo script de análise está disponível para avaliar o custo de migração para pipelines do tipo V2. Consulte Que tipo de pipeline é ideal para mim?. Foi adicionada uma tabela de referência que fornece links para toda a documentação do CodePipeline serviço por recurso. Veja a referência do CodePipeline recurso.	11 de julho de 2024

Atualizações na ação de origem do S3 para adicionar uma nova opção para substituições de origem	Uma nova opção para substituições de origem chamada `S3_OBJECT_KEY` está disponível para a ação de origem do S3. Um novo parâmetro `AllowOverrideForS3ObjectKey` foi adicionado para a ação de origem do S3. Consulte a página de referência Amazon S3 source action e Start a pipeline with a source revision override.	7 de junho de 2024
Atualizações na ação de origem do S3 para adicionar novas variáveis de saída	Novas variáveis de saída chamadas `BucketName` e `ObjectKey` estão disponíveis para a ação de origem do S3. Consulte a página de referência Amazon S3 source action.	5 de junho de 2024
Atualizações das ações `CloudFormationStackSet` e `CloudFormationStackInstances`	O parâmetro `CallAs` foi adicionado para as ações `CloudFormationStackSet` e `CloudFormationStackInstances`. Consulte a página de referência da ação.	2 de maio de 2024
Suporte para reversões de estágio	Você pode reverter um estágio de maneira manual ou automática para uma execução de pipeline bem-sucedida anterior do estágio. Consulte Configuring stage rollback e Concepts.	26 de abril de 2024

Atualizações na disponibilidade da região StackSets e ações do Step Functions	As ações StackSets e Step Functions agora estão disponíveis em todas as regiões em CodePipeline que estão disponíveis. Consulte a referência de CloudFormation StackSets ação e a referência de ação do AWS Step Functions.	27 de março de 2024
Atualizações da política gerenciada	A política AWS `AWSCodePipeline_FullAccess` gerenciada foi atualizada. Consulte Políticas gerenciadas pela AWS para o AWS CodePipeline.	15 de março de 2024
Suporte a tempo limite configurável para ações de aprovação manual	Adição de informações de cota para o novo campo de tempo limite configurável para ações de aprovação manual. Para mais informações, consulte Cotas do .	15 de fevereiro de 2024
Suporte à filtragem de gatilho por ramificações e caminhos de arquivo	Adição de suporte à configuração de gatilho que permite filtragem por status de solicitação pull, ramificações e caminhos de arquivo para pipelines do tipo V2. Para ter mais informações, consulte Filtering triggers on code push or pull requests, Triggers, Filter on feature branches to start your pipeline e Quotas.	8 de fevereiro de 2024

Suporte a novos modos de execução de pipeline	Adição de suporte aos modos de execução de pipeline PARALLEL e QUEUED. Para ter mais informações, consulte Set the pipeline execution mode, How executions are processed in QUEUED mode, How executions are processed in PARALLEL mode e Quotas.	8 de fevereiro de 2024
Atualizações nas páginas do console para visualizar os detalhes das ações, revisar as ações de aprovação manual e a página de lista de pipelines	Atualizações do console documentadas para o novo botão e caixa de diálogo Visualizar detalhes, nova caixa de diálogo de aprovação manual e novas colunas para execuções recentes na página de lista de pipelines. Para obter mais informações, consulte Visualizar pipelines (console), Visualizar detalhes da ação em um pipeline e Gerenciar ações de aprovação em pipelines.	10 de janeiro de 2024
Support para GitLab autogerenciamento	Support adicionado para configurar conexões para que AWS os recursos interajam com o GitLab autogeren ciamento. Para obter mais informações, consulte Conexões para GitLab autogerenciamento.	28 de dezembro de 2023

Atualizações das ações `CloudFormationStackSet` e `CloudFormationStackInstances`	O parâmetro `ConcurrencyMode` foi adicionado para as ações `CloudFormationStackSet` e `CloudFormationStackInstances`. Consulte a página de referência da ação.	19 de dezembro de 2023
Atualizações nos parâmetros de AWS Device Farm ação em CodePipeline	Os parâmetros para a AWS Device Farm ação em CodePipeline foram atualizados. Para obter mais informações, consulte a Referência da ação do AWS Device Farm.	18 de dezembro de 2023
Support adicionado para mensagens de erro detalhadas para a CloudFormation ação em CodePipeline	CloudFormation mensagens de erro de ação agora podem mostrar detalhes sobre recursos que falharam. Para obter mais informações, consulte a Referência da ação do CloudFormation.	15 de dezembro de 2023
Atualizações para iniciar um pipeline com substituições de revisão de código-fonte em CodePipeline	Agora é possível iniciar um pipeline com uma revisão de origem especificada. Para obter mais informações, consulte Start a pipeline with a source revision override.	17 de novembro de 2023

Novas regiões compatíveis	CodePipeline agora está disponível nas regiões Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Jacarta), Ásia-Pacífico (Melbourne), Ásia-Pacífico (Osaka), Oriente Médio (EAU), Europa (Espanha) e Israel (Tel Aviv). Os tópicos Referência do bucket de espaço reservado de eventos e Endpoints do AWS service (Serviço da AWS) foram atualizados.	13 de novembro de 2023
Atualizações para campos de eventos na Amazon EventBridge	Agora você pode ver os campos de eventos atualizados na Amazon EventBridge. Para obter mais informações, consulte Monitoramento de CodePipeline eventos.	9 de novembro de 2023

Atualizações para novos pipelines do tipo V2 de pipeline, gatilhos em tags Git e variáveis de pipeline em CodePipeline	Agora você pode escolher um tipo de funil CodePipeline. Para um pipeline do tipo V2, agora você pode usar uma configuração de gatilho para iniciar seu pipeline nas tags Git. Com os pipelines do tipo V2, você também pode usar variáveis no nível do pipeline para transmitir parâmetros de entrada na execução de um pipeline. Para obter mais informações, consulte Variáveis, Tutorial: Usar variáveis no nível do pipeline e Tutorial: Usar tags Git para iniciar seu pipeline. Para obter mais informações sobre os tipos de pipeline, consulte Tipos de pipeline.	24 de outubro de 2023
CodePipeline permite repetir todas as ações em um estágio com falha	Em caso de falha em um estágio CodePipeline, você pode repetir o estágio sem executar novamente o pipeline. Você faz isso repetindo as ações que falharam em um estágio ou repetindo todas as ações do estágio desde a primeira ação. Para ter mais informações, consulte.	17 de outubro de 2023

Support para GitLab grupos	Support adicionado para configurar conexões para AWS recursos interagirem com GitLab grupos. Para obter mais informações, consulte GitLab conexões.	15 de setembro de 2023
CodePipeline suporta conexões com GitLab .com	Você pode usar conexões para configurar AWS recursos para interagir com GitLab .com. Você também pode escolher a opção de clonagem completa para usar os comandos e metadados do Git em ações downstrea m. Para obter mais informaçõ es, consulte GitLab conexões e o tópico de referência da estrutura de CodeStarS ourceConnection ação.	10 de agosto de 2023
Atualização da ação `CloudFormationStac kInstances`	O parâmetro `RegionCon currencyType` foi adicionado para a ação `CloudFormationStac kInstances`. Consulte a página de referência da ação `CloudFormationStac kInstances`.	8 de agosto de 2023

Atualização da ação `CloudFormationStackSet`	O parâmetro `RegionConcurrencyType` foi adicionado para a ação `CloudFormationStackSet`. Consulte a página de referência da ação `CloudFormationStackSet`.	24 de julho de 2023
Atualizações da política gerenciada	A política AWS `AWSCodePipeline_FullAccess` gerenciada foi atualizada. Consulte Políticas gerenciadas pela AWS para o AWS CodePipeline.	21 de junho de 2023
Atualizações dos procedimentos de migração para pipelines de pesquisa	Os procedimentos para migrar (atualizar) os pipelines de votação para usar a detecção de alterações baseada em eventos foram atualizados com as etapas dos pipelines que usam um bucket do Amazon S3 habilitado para notificações. EventBridge Para obter mais informações, consulte Migrar pipelines de pesquisa para usar a detecção de alterações baseada em eventos.	12 de junho de 2023

Atualizações das políticas gerenciadas	As políticas AWS `AWSCodePipeline_FullAccess` gerenciadas `AWSCodePipeline_ReadOnlyAccess` foram atualizadas com uma permissão adicional. Para obter mais informações, consulte AWS CodePipeline atualizações nas políticas AWS gerenciadas.	16 de maio de 2023
Atualizações das políticas gerenciadas	As políticas AWS `AWSCodePipelineFullAccess` gerenciadas `AWSCodePipelineReadOnlyAccess` estão obsoletas. Use as políticas `AWSCodePipeline_FullAccess` e `AWSCodePipeline_ReadOnlyAccess`. Consulte Atualizações do AWS CodePipeline nas políticas gerenciadas pela AWS.	17 de novembro de 2022

Atualizações nos procedimentos que usam CloudTrail	Todos os procedimentos do console, exemplos de comandos da CLI e exemplos de CloudFormation trechos e modelos de um pipeline com uma fonte do S3 foram atualizados com a opção de escolher Gravar e selecionar falso para eventos de gerenciamento em. CloudTrail Veja os exemplos atualizados em Iniciando um pipeline, Tutorial: Criar um pipeline com CloudFormation, Editar pipelines para usar eventos push e Atualizar pipelines de enquete.	27 de abril de 2022
Nova integração ao Snyk compatível	Você pode usar a ação de invocação do Snyk CodePipeline para automatizar a verificação de segurança do seu código-fonte aberto. Para obter mais informações, consulte Referência de ação do Snyk e Integrações.	10 de junho de 2021
Nova região compatível: Europa (Milão)	CodePipeline agora está disponível na Europa (Milão). Os tópicos Limites e AWS service (Serviço da AWS) Endpoints do foram atualizados.	27 de janeiro de 2021

A detecção de alterações pode ser desativada para ações de origem com conexões	Você pode usar a CLI ou o SDK para atualizar a ação de origem `CodeStarSourceConnection` para desativar a detecção automática de alterações no repositório de origem. O tópico de referência da estrutura de CodeStarSourceConnection ação foi atualizado com uma descrição do `DetectChanges` parâmetro.	8 de janeiro de 2021
CodePipeline agora oferece suporte a ações CloudFormation StackSets de implantação	Um novo tutorial, Tutorial: Crie um pipeline usado CloudFormation StackSets como provedor de implantação, fornece etapas a serem usadas CloudFormation StackSets para criar e atualizar seus conjuntos de pilhas e instâncias de pilha com seu pipeline. O tópico de referência da estrutura de CloudFormation StackSets ação também foi adicionado.	30 de dezembro de 2020
Nova região compatível: Ásia-Pacífico (Hong Kong)	CodePipeline agora está disponível na Ásia-Pacífico (Hong Kong). Os tópicos Limites e AWS service (Serviço da AWS) Endpoints do foram atualizados.	22 de dezembro de 2020

Veja os padrões de EventBridge eventos atualizados em CodePipeline	Padrões e status de eventos atualizados para eventos de pipeline, estágio e nível de ação foram adicionados aos CodePipeline eventos de monitoramento.	21 de dezembro de 2020
Veja as execuções do pipeline de entrada em CodePipeline	Você pode usar o console ou a CLI para visualizar execuções de entrada. Para obter mais informações, consulte Visualizar uma execução de entrada (console) e Visualizar status de execução de entrada (CLI).	16 de novembro de 2020
A ação CodeCommit de origem em CodePipeline suporta a opção de clonagem completa	Ao usar uma ação de CodeCommit origem, você pode escolher a opção de clonagem completa para usar os comandos e metadados do Git para ações posteriores. CodeBuild Para obter mais informações, consulte a referência da CodeCommit ação e o Tutorial: Use o clone completo com uma fonte de CodeCommit pipeline.	11 de novembro de 2020

CodePipeline suporta conexões com GitHub um GitHub Enterprise Server	Você pode usar conexões para configurar AWS recursos para interagir com GitHub o GitHub Enterprise Cloud e o GitHub Enterprise Server. Você também pode escolher a opção de clonagem completa para usar os comandos e metadados do Git em ações downstream. Para obter mais informações, consulte GitHub conexões, conexões do GitHub Enterprise Server e Tutorial: Use o clone completo com uma origem de GitHub pipeline. Se você tiver um pipeline existente com uma ação de GitHub origem, consulte Atualizar uma ação de origem GitHub (via OAuth aplicativo) para uma ação de origem GitHub (via GitHub aplicativo).	30 de setembro de 2020
A CodeBuild ação oferece suporte à ativação de compilações em lote no AWS CodePipeline	Para CodeBuild ações em seu pipeline, você pode permitir que compilações em lote executem várias compilaçõ es em uma única execução. Para obter mais informaçõ es, consulte a referência da estrutura de CodeBuild ação e Criar um pipeline (console).	30 de julho de 2020

AWS CodePipeline agora oferece suporte a ações AWS AppConfig de implantação	Um novo tutorial, Tutorial: Crie um pipeline usado AWS AppConfig como provedor de implantação, fornece etapas a serem usadas AWS AppConfig para implantar arquivos de configuração com seu pipeline. O tópico de referência da estrutura de AWS AppConfig ação também foi adicionado.	25 de junho de 2020
AWS CodePipeline agora oferece suporte ao Amazon VPC em AWS GovCloud (Oeste dos EUA)	Agora você pode se conectar diretamente AWS CodePipeline por meio de um endpoint privado da Amazon VPC em AWS GovCloud (Oeste dos EUA). Para obter mais informações, consulte Usar CodePipeline com a Amazon Virtual Private Cloud.	2 de junho de 2020

AWS CodePipeline agora suporta ações de AWS Step Functions invocação	Agora você pode criar um pipeline CodePipeline que use AWS Step Functions como provedor de ação de invocação. Um novo tutorial, Tutorial: Use uma ação de AWS Step Functions invocação em um pipeline, fornece etapas para iniciar a execução de uma máquina de estado a partir do seu pipeline. O tópico Referência da estrutura de ação do AWS Step Functions também foi adicionado.	28 de maio de 2020
Visualização, listagem e atualização de conexões	É possível listar, excluir e atualizar conexões no console. Consulte Listar conexões em CodePipeline.	21 de maio de 2020
As conexões são compatíveis com a marcação de recursos de conexões na CLI	Os recursos de conexões agora oferecem suporte à marcação na AWS CLI. As conexões agora se integram com AWS CodeGuru o. Consulte Referência de permissões do IAM para conexões.	6 de maio de 2020
CodePipeline agora está disponível em AWS GovCloud (Oeste dos EUA)	Agora você pode usar CodePipeline em AWS GovCloud (Oeste dos EUA). Para mais informações, consulte Cotas do .	8 de abril de 2020

O tópico de cotas mostra quais cotas CodePipeline de serviço são configuráveis	O tópico de CodePipeline cotas foi reformatado. A documentação mostra quais cotas de serviço são configurá veis e quais não são. Veja as cotas em AWS CodePipeline.	12 de março de 2020
O tempo limite da ação de implantação do Amazon ECS é configurável	O tempo limite da ação de implantação do Amazon ECS é configurável até uma hora (o tempo limite padrão). Consulte Cotas em AWS CodePipeline.	5 de fevereiro de 2020
Novos tópicos descrevem como você pode interromper a execução de um pipeline	É possível interromper a execução de um pipeline no CodePipeline. Você pode especificar que a execução é interrompida após a conclusão das ações em andamento ou pode especificar para interromper a execução imediatamente e abandonar as ações em andamento. Consulte Como as execuções de pipeline são interrompidas e Interromper a execução de um pipeline no CodePipeline.	21 de janeiro de 2020

CodePipeline suporta conexões	Você pode usar conexões para configurar AWS recursos para interagir com repositór ios de código externos. Cada conexão é um recurso que pode ser usado por serviços como CodePipeline a conexão com um repositór io de terceiros, como o Bitbucket Cloud. Para obter mais informações, consulte Trabalhando com conexões em CodePipeline.	18 de dezembro de 2019
Atualização dos tópicos sobre segurança, autenticação e controle de acesso	As informações de segurança , autenticação e controle de acesso do CodePipeline foram organizadas em um novo capítulo de Segurança. Para obter mais informações, consulte Segurança.	17 de dezembro de 2019
Novos tópicos descrevem como é possível usar variáveis em seus pipelines	Agora você pode configura r namespaces para ações e gerar variáveis cada vez que a execução da ação for concluída. É possível configurar ações downstrea m para fazer referência a esses namespaces e variáveis . Consulte Trabalhar com variáveis e Variáveis.	14 de novembro de 2019

Novos tópicos descrevem como funcionam as execuções do pipeline, por que os estágios são bloqueados durante uma execução e quando as execuções do pipeline são substituídas	Vários tópicos foram adicionados à seção Bem-vindo para descrever como as execuções de pipeline funcionam, incluindo por que os estágios são bloqueados durante uma execução e o que acontece quando as execuções de pipeline são substituídas. Esses tópicos incluem uma lista de conceitos, um exemplo DevOps de fluxo de trabalho e recomendações sobre como um pipeline deve ser estruturado. Os tópicos a seguir foram adicionados: termos do pipeline, exemplo do DevOps pipeline e Como as execuções do pipeline funcionam.	11 de novembro de 2019
CodePipeline suporta regras de notificação	Agora você pode usar regras de notificação para notificar os usuários sobre alterações importantes em pipelines. Para obter mais informações, consulte Criar uma regra de notificação.	5 de novembro de 2019

CodeBuild variáveis de ambiente disponíveis em CodePipeline	Você pode definir variáveis de CodeBuild ambiente na ação de CodeBuild criação do seu pipeline. Você pode usar o console ou a CLI para adicionar o parâmetro `EnvironmentVariables` à estrutura do pipeline. O tópico Criar um pipeline (console) foi atualizado. Os exemplos de configuração de ação na referência de ação do CodeBuild também foram atualizados.	14 de outubro de 2019
Nova região	CodePipeline agora está disponível na Europa (Estocolmo). Os tópicos Limites e AWS service (Serviço da AWS) Endpoints do foram atualizados.	5 de setembro de 2019
Especifique o controle automático ACLs e de cache para ações de implantação do Amazon S3	Agora você pode especificar opções predefinidas de ACL e controle de cache ao criar uma ação de implantação do Amazon S3 em. CodePipeline Os tópicos a seguir foram atualizados: Criar um pipeline (console), Referência da estrutura do CodePipeline pipeline e Tutorial: Crie um pipeline que use o Amazon S3 como provedor de implantação.	27 de junho de 2019

Agora você pode adicionar tags aos recursos no AWS CodePipeline	Agora você pode usar a marcação para rastrear e gerenciar AWS CodePipeline recursos como pipelines, ações personalizadas e webhooks. Os novos tópicos a seguir foram adicionados: Marcação de recursos, Uso de tags para controlar o acesso aos CodePipeline recursos, Marcar um pipeline CodePipeline, Marcar uma ação personalizada em CodePipeline e Marcar um webhook em. CodePipeline Os tópicos a seguir foram atualizados para mostrar como usar a CLI para marcar recursos: Criar um pipeline (CLI), Criar uma ação personalizada (CLI) e Criar um webhook para uma fonte. GitHub	15 de maio de 2019

Agora você pode ver o histórico de execução da ação no AWS CodePipeline	Agora é possível visualizar detalhes sobre execuções anteriores de todas as ações em um pipeline. Esses detalhes incluem os horários de início e término, a duração, o ID de execução da ação, o status, os detalhes da localização de artefatos de entrada e saída e detalhes de recursos externos. O tópico Visualizar detalhes e histórico do pipeline foi atualizado para refletir esse suporte.	20 de março de 2019
AWS CodePipeline agora suporta a publicação de aplicativos no AWS Serverless Application Repository	Agora você pode criar um pipeline CodePipeline que publique seu aplicativo sem servidor no. AWS Serverless Application Repository Um novo tutorial, Tutorial: Publish applications to the AWS Serverless Application Repository, fornece etapas para criar e configurar um pipeline para entregar continuamente seu aplicativo sem servidor ao. AWS Serverless Application Repository	8 de março de 2019

AWS CodePipeline agora oferece suporte a ações entre regiões no console	Agora você pode gerenciar ações entre regiões no AWS CodePipeline console. O tópico Adicionar uma ação entre regiões foi atualizado com as etapas de adição, edição ou exclusão de uma ação que esteja em uma região da AWS diferente da do pipeline. Os tópicos Criar um pipeline, Editar um pipeline e Referência da estrutura do pipeline do CodePipeline foram atualizados.	14 de fevereiro de 2019
AWS CodePipeline agora suporta implantações do Amazon S3	Agora você pode criar um pipeline CodePipeline que usa o Amazon S3 como provedor de ações de implantação. Um novo tutorial, Tutorial: Crie um pipeline que usa o Amazon S3 como provedor de implantação, fornece etapas para implantar arquivos de amostra em seu bucket do Amazon S3 com. CodePipeline O tópico Referência da estrutura do pipeline do CodePipeline também foi atualizado.	16 de janeiro de 2019

AWS CodePipeline agora suporta implantações do Alexa Skills Kit	Agora você pode usar o CodePipeline Alexa Skills Kit para a implantação contínua das habilidades da Alexa. Um novo tutorial, Tutorial: Crie um pipeline que implanta uma habilidade da Amazon Alexa, contém etapas para criar credenciais que permitam conectar-se AWS CodePipeline à sua conta de desenvolvedor do Alexa Skills Kit e, em seguida, criar um pipeline que implanta uma habilidade de amostra. O tópico Referência da estrutura do pipeline do CodePipeline foi atualizado.	19 de dezembro de 2018
AWS CodePipeline agora oferece suporte a endpoints Amazon VPC desenvolvidos por AWS PrivateLink	Agora você pode se conectar diretamente AWS CodePipeline por meio de um endpoint privado em sua VPC, mantendo todo o tráfego dentro da VPC e da rede. AWS Para obter mais informações, consulte Usar CodePipeline com a Amazon Virtual Private Cloud.	6 de dezembro de 2018

AWS CodePipeline agora suporta ações de origem e ações de ECS-to-CodeDeploy implantação do Amazon ECR	Agora você pode usar CodePipeline e CodeDeploy com o Amazon ECR e o Amazon ECS para a implantação contínua de aplicativos baseados em contêineres. Um novo tutorial, Criar um pipeline com uma fonte e ECS-to-CodeDeploy implantação do Amazon ECR, contém etapas para usar o console para criar um pipeline que implanta aplicativos de contêiner armazenados em um repositório de imagens em um cluster do Amazon ECS com roteamento de tráfego. CodeDeploy Os tópicos Criar um pipeline e Referência da estrutura do pipeline do CodePipeline foram atualizados.	27 de novembro de 2018
AWS CodePipeline agora oferece suporte a ações entre regiões em um pipeline	Um novo tópico, Adicionar uma ação entre regiões, contém etapas para usar AWS CLI ou CloudFormation adicionar uma ação que esteja em uma região diferente do seu pipeline. Os tópicos Criar um pipeline, Editar um pipeline e Referência da estrutura do pipeline do CodePipeline foram atualizados.	12 de novembro de 2018

AWS CodePipeline agora se integra ao Service Catalog	Agora, é possível adicionar o Service Catalog como uma ação de implantação ao pipeline. Isso permite que você configure um pipeline para publicar atualizaç ões do produto no Service Catalog quando você faz uma alteração no repositório de origem. O tópico Integrações foi atualizado para refletir esse suporte ao Service Catalog. Dois tutoriais do Service Catalog foram adicionados à seção Tutoriais do AWS CodePipeline.	16 de outubro de 2018
AWS CodePipeline agora se integra com AWS Device Farm	Agora você pode adicionar AWS Device Farm como uma ação de teste ao seu pipeline. Isso permite que você configure um pipeline para testar aplicativos móveis. O tópico Integrações foi atualizado para refletir esse suporte para AWS Device Farm. Dois tutoriais do AWS Device Farm foram adicionad os à seção Tutoriais do AWS CodePipeline.	19 de julho de 2018

AWS CodePipeline Notificações de atualização do Guia do Usuário agora disponíveis por meio de RSS	A versão HTML do Guia do CodePipeline Usuário agora oferece suporte a um feed RSS de atualizações que estão documentadas na página Histórico de atualizações da documentação. O feed RSS inclui as atualizações feitas após 30 de junho de 2018. As atualizações anunciadas anteriormente ainda estão disponíveis na página Histórico de atualizações da documentação. Use o botão RSS no menu superior do painel para assinar o feed.	30 de junho de 2018

Atualizações anteriores

A tabela a seguir descreve mudanças importantes em cada versão do Guia do CodePipeline usuário em 30 de junho de 2018 e anteriores.

Alteração	Descrição	Alterado em
Use webhooks para detectar mudanças na fonte nos pipelines GitHub	Quando você cria ou edita um pipeline no console, CodePipeline agora cria um webhook que detecta alterações no seu repositório de GitHub origem e, em seguida, inicia seu pipeline. Para obter informações sobre como migrar seu pipeline, consulte Configurar seus GitHub pipelines para usar webhooks para detecção de alterações. Para obter mais informações, consulte Iniciar a execução de um pipeline no CodePipeline.	1.º de maio de 2018
Tópicos atualizados	Quando você cria ou edita um pipeline no console, CodePipeline agora cria uma regra do Amazon CloudWatch Events e uma AWS CloudTrail trilha que detecta	22 de março de 2018

Alteração	Descrição	Alterado em
	alterações em seu bucket de origem do Amazon S3 e, em seguida, inicia seu pipeline. Para obter informações sobre como migrar um pipeline, consulte Ações de origem e métodos de detecção de alterações. Tutorial: Criar um pipeline simples (bucket do S3)Foi atualizado para mostrar como a regra e a trilha do Amazon CloudWatch Events são criadas quando você seleciona uma fonte do Amazon S3. Criar um pipeline, estágios e açõese também Editar um pipeline no CodePipeline foram atualizados. Para obter mais informações, consulte Inicie um funil em CodePipeline.	
Tópico atualizado	CodePipeline agora está disponível na Europa (Paris). O tópico Cotas em AWS CodePipeline foi atualizado.	21 de fevereiro de 2018
Tópicos atualizados	Agora você pode usar o CodePipeline Amazon ECS para a implantação contínua de aplicativos baseados em contêineres. Ao criar um pipeline, você pode selecionar o Amazon ECS como um provedor de implantação. Uma alteração de código no repositório de controle de origem faz com que o pipeline crie uma nova imagem do Docker, a envie por push ao registro de contêiner e, em seguida, implante a imagem atualizada em um serviço do Amazon ECS. Os tópicos Integrações de produtos e serviços com CodePipeline, Criar um pipeline, estágios e ações e CodePipeline referência de estrutura de tubulação foram atualizados para refletir essa compatibilidade do Amazon ECS.	12 de dezembro de 2017

Alteração	Descrição	Alterado em
Tópicos atualizados	Quando você cria ou edita um pipeline no console, CodePipeline agora cria uma regra do Amazon CloudWatch Events que detecta alterações no seu CodeCommit repositório e, em seguida, inicia automaticamente o funil. Para obter informações sobre como migrar um pipeline, consulte Ações de origem e métodos de detecção de alterações. Tutorial: criar um pipeline simples (CodeCommit repositório)Foi atualizado para mostrar como a regra e a função do Amazon CloudWatch Events são criadas quando você seleciona um CodeCommit repositório e uma filial. Criar um pipeline, estágios e ações e também Editar um pipeline no CodePipeline foram atualizados. Para obter mais informações, consulte Inicie um funil em CodePipeline.	11 de outubro de 2017
Tópicos novos e atualizados	CodePipeline agora fornece suporte integrado para notificações de alteração do estado do pipeline por meio do Amazon CloudWatch Events e do Amazon Simple Notification Service (Amazon SNS). O novo tutorial Tutorial: configurar uma regra de CloudWatch eventos para receber notificações por e-mail sobre mudanças no estado do pipeline foi adicionado. Para obter mais informações, consulte Monitoramento de CodePipeline eventos.	8 de setembro de 2017

Alteração	Descrição	Alterado em
Tópicos novos e atualizados	Agora você pode adicionar CodePipeline como alvo as ações da Amazon CloudWatch Events. As regras do Amazon CloudWatch Events podem ser configuradas para detectar alterações na origem para que o pipeline comece assim que essas alterações ocorrerem, ou podem ser configuradas para executar execuções programadas do pipeline. Foram adicionadas informações para a opção de configuração da ação de PollForSourceChanges origem. Para obter mais informações, consulte Inicie um funil em CodePipeline.	5 de setembro de 2017
Novas regiões da	CodePipeline agora está disponível na Ásia-Pacífico (Seul) e Ásia-Pacífico (Mumbai). Os tópicos Cotas em AWS CodePipeline e Regiões e endpoints foram atualizados.	27 de julho de 2017
Novas regiões da	CodePipeline agora está disponível no Oeste dos EUA (Norte da Califórnia), Canadá (Central) e Europa (Londres) . Os tópicos Cotas em AWS CodePipeline e Regiões e endpoints foram atualizados.	29 de junho de 2017
Tópicos atualizados	Agora você pode visualizar detalhes sobre execuções anteriores de um pipeline, não apenas a execução mais recente. Esses detalhes incluem horas de início e de término, duração e ID de execução. Detalhes estão disponíveis para um máximo de 100 execuções de pipeline durante os últimos 12 meses. Os tópicos Veja os pipelines e os detalhes em CodePipeline, referência de permissões e Cotas em AWS CodePipeline foram atualizados para refletir este suporte.	22 de junho de 2017
Tópico atualizado	Nouvola foi adicionado à lista de ações disponíveis em Integrações de ações de teste.	18 de maio de 2017

Alteração	Descrição	Alterado em
Tópicos atualizados	No AWS CodePipeline assistente, a página Etapa 4: Beta foi renomeada para Etapa 4: Implantação. O nome padrão do estágio criado por esta etapa foi alterado de "Beta" para "Preparação". Vários tópicos e capturas de tela foram atualizados para refletir tais alterações.	7 de abril de 2017
Tópicos atualizados	Agora você pode adicionar AWS CodeBuild como uma ação de teste a qualquer estágio de um pipeline. Isso permite que você use com mais facilidade AWS CodeBuild para executar testes de unidade em seu código. Antes dessa versão, você podia usar AWS CodeBuild para executar testes de unidade somente como parte de uma ação de compilação. Uma ação de build requer um artefato de saída de build, o que os testes de unidade geralmente não produzem. Os tópicosIntegrações de produtos e serviços com CodePipeline,Editar um pipeline no CodePipeline, e CodePipeline referência de estrutura de tubulação foram atualizados para refletir esse suporte para AWS CodeBuild .	8 de março de 2017

Alteração	Descrição	Alterado em
Tópicos novos e atualizados	O índice foi reorganizado para incluir seções sobre pipelines, ações e transições de estágio. Uma nova seção foi adicionada para CodePipeline tutoriais. Para obter melhor usabilidade, Integrações de produtos e serviços com CodePipeline foi dividido em tópicos mais curtos. Uma nova seção, Autorização e Controle de Acesso, fornece informações abrangentes sobre o uso AWS Identity and Access Management (IAM) e ajuda CodePipel ine a proteger o acesso aos seus recursos por meio do uso de credenciais. Essas credenciais fornecem as permissõe s necessárias para acessar AWS recursos, como colocar e recuperar artefatos dos buckets do Amazon S3 e integrar pilhas em seus pipelines. OpsWorks	8 de fevereiro de 2017
Nova região da	CodePipeline agora está disponível na Ásia-Pacífico (Tóquio). Os tópicos Cotas em AWS CodePipeline e Regiões e endpoints foram atualizados.	14 de dezembro de 2016
Nova região da	CodePipeline agora está disponível na América do Sul (São Paulo). Os tópicos Cotas em AWS CodePipeline e Regiões e endpoints foram atualizados.	7 de dezembro de 2016

Alteração	Descrição	Alterado em
Tópicos atualizados	Agora você pode adicionar AWS CodeBuild como uma ação de construção a qualquer estágio de um funil. AWS CodeBuild é um serviço de criação totalmente gerenciado na nuvem que compila seu código-fonte, executa testes de unidade e produz artefatos prontos para implantação. Você pode usar um projeto de compilação existente ou criar um no CodePipeline console. Em seguida, o resultado do projeto de build poderá ser implantado como parte de um pipeline. Os tópicos Integrações de produtos e serviços com CodePipelineCriar um pipeline, estágios e ações, Autenticação e Controle de Acesso, CodePipeline referência de estrutura de tubulação foram atualizados para refletir esse suporte para AWS CodeBuild. Agora você pode usar CodePipeline com CloudFormation o modelo de aplicativo AWS sem servidor para fornecer continuamente seus aplicativos sem servidor. O tópico Integrações de produtos e serviços com CodePipeline foi atualizado para refletir essa compatibilidade. Integrações de produtos e serviços com CodePipelinefoi reorganizado para ofertas de grupos AWS e parceiros por tipo de ação.	1° de dezembro de 2016
Nova região da	CodePipeline agora está disponível na Europa (Frankfurt). Os tópicos Cotas em AWS CodePipeline e Regiões e endpoints foram atualizados.	16 de novembro de 2016

Alteração	Descrição	Alterado em
Tópicos atualizados	AWS CloudFormation agora pode ser selecionado como provedor de implantação em pipelines, permitindo que você execute ações em CloudFormation pilhas e conjuntos de alterações como parte da execução de um pipeline. Os tópicos Integrações de produtos e serviços com CodePipelineCriar um pipeline, estágios e ações, Autenticação e Controle de Acesso, CodePipeline referência de estrutura de tubulação foram atualizados para refletir esse suporte para AWS CloudFormation.	3 de novembro de 2016
Nova região da	CodePipeline agora está disponível na região Ásia-Pacífico (Sydney). Os tópicos Cotas em AWS CodePipeline e Regiões e endpoints foram atualizados.	26 de outubro de 2016
Nova região da	CodePipeline agora está disponível na Ásia-Pacífico (Cingapura). Os tópicos Cotas em AWS CodePipeline e Regiões e endpoints foram atualizados.	20 de outubro de 2016
Nova região da	CodePipeline agora está disponível na região Leste dos EUA (Ohio). Os tópicos Cotas em AWS CodePipeline e Regiões e endpoints foram atualizados.	17 de outubro de 2016
Tópico atualizado	Criar um pipeline, estágios e ações foi atualizado para refletir a capacidade de exibir os identificadores de versão de ações personalizadas nas listas Provedor de origem e Provedor build.	22 de setembro de 2016
Tópico atualizado	A seção Adicionar uma ação de aprovação manual a uma fase foi atualizada para refletir um aprimoramento que permite que os revisores de ação de aprovação abram o formulário de aprovação ou rejeição da revisão diretamente de uma notificação por e-mail.	14 de setembro de 2016

Alteração	Descrição	Alterado em
Tópicos novos e atualizados	Um novo tópico que descreve como visualizar detalhes sobre as alterações de código que ocorrem no momento no pipeline de lançamento de software. O acesso rápido a essas informações pode ser útil ao analisar ações de aprovação manual ou falhas de solução de problemas no pipeline. Uma nova seção, Monitorar pipelines, fornece um local central para todos os tópicos referentes ao monitoramento do status e do progresso dos pipelines.	08 de setembro de 2016
Tópicos novos e atualizados	Uma nova seção, Adicionar uma ação de aprovação manual a uma fase, fornece informações sobre como configurar e usar ações de aprovação manual nos pipelines. Os tópicos nesta seção fornecem informações conceituais sobre o processo de aprovação; instruções sobre a configuração das permissões do IAM necessári as, a criação de ações de aprovação, e a aprovação ou rejeição de ações de aprovação; e amostras de dados JSON gerados quando uma ação de aprovação é acessada em um pipeline.	06 de julho de 2016
Nova região da	CodePipeline agora está disponível na região Europa (Irlanda). Os tópicos Cotas em AWS CodePipeline e Regiões e endpoints foram atualizados.	23 de junho de 2016
Novo tópico	Um novo tópico, , foi adicionado para descrever como repetir uma ação falha ou um grupo de ações falhas paralelas em estágio.	22 de junho de 2016

Alteração	Descrição	Alterado em
Tópicos atualizados	Vários tópicos, incluindo Criar um pipeline, estágios e ações Autenticação e Controle de Acesso, e CodePipeline referência de estrutura de tubulaçãoIntegrações de produtos e serviços com CodePipeline, foram atualizados para refletir o suporte à configuração de um pipeline para implantar código em conjunto com livros de receitas e aplicativos personalizados do Chef criados em. AWS OpsWorks CodePipeline Atualmente, o suporte para OpsWorks está disponível somente na região Leste dos EUA (Norte da Virgínia) (us-east-1).	2 de junho de 2016
Tópicos novos e atualizados	Um novo tópico, Tutorial: criar um pipeline simples (CodeCommit repositório), foi adicionado. Este tópico fornece um exemplo de apresentação que mostra como usar um CodeCommit repositório e uma ramificação como local de origem para uma ação de origem em um pipeline. Vários outros tópicos foram atualizados para refletir essa integração com CodeCommit, incluindo Autenticação e Controle de AcessoIntegrações de produtos e serviços com CodePipeline,Tutorial: Criar um pipeline de quatro estágios,, Solução de problemas CodePipeline e.	18 de abril de 2016
Novo tópico	Um novo tópico, Invocar um AWS Lambda função em um pipeline em CodePipeline, foi adicionado. Este tópico contém exemplos de AWS Lambda funções e etapas para adicionar funções Lambda aos pipelines.	27 de janeiro de 2016
Tópico atualizado	Uma nova seção foi adicionada às políticas de Autenticação e controle de acesso baseadas em recursos.	22 de janeiro de 2016

Alteração	Descrição	Alterado em
Novo tópico	Um novo tópico, Integrações de produtos e serviços com CodePipeline, foi adicionado. As informações sobre integrações com parceiros e outros Serviços da AWS foram transferidas para este tópico. Links para blogs e vídeos também foram adicionados.	17 de dezembro de 2015
Tópico atualizado	Detalhes de integração com o Solano CI foram adicionados ao título Integrações de produtos e serviços com CodePipeline.	17 de novembro de 2015
Tópico atualizado	O CodePipeline plug-in para Jenkins agora está disponível por meio do Jenkins Plugin Manager como parte da biblioteca de plug-ins para Jenkins. As etapas para instalar o plug-in foram atualizadas em Tutorial: Criar um pipeline de quatro estágios.	9 de novembro de 2015
Nova região da	CodePipeline agora está disponível na região Oeste dos EUA (Oregon). O tópico Cotas em AWS CodePipeline foi atualizado. Links foram adicionados a Regiões e endpoints .	22 de outubro de 2015
Novo tópico	Dois novos tópicos, Configure a criptografia do lado do servidor para artefatos armazenados no Amazon S3 para CodePipeline e Crie um pipeline CodePipeline que use recursos de outro AWS account, foram adicionados. Uma nova seção foi adicionada a Autenticação e controle de acesso.	25 de agosto de 2015
Tópico atualizado	O tópico Crie e adicione uma ação personalizada no CodePipeline foi atualizado para refletir as alterações na estrutura, inclusive `inputArtifactDetails` e `outputArtifactDetails` .	17 de agosto de 2015
Tópico atualizado	O tópico Solução de problemas CodePipeline foi atualizado com etapas revisadas sobre solução de problemas do perfil de serviço e o Elastic Beanstalk.	11 de agosto de 2015

Alteração	Descrição	Alterado em
Tópico atualizado	O tópico Autenticação e controle de acesso foi atualizado com as alterações mais recentes na função de serviço para CodePipeline.	6 de agosto de 2015
Novo tópico	Um tópico Solução de problemas CodePipeline foi adicionado. Etapas atualizadas foram adicionadas para perfis do IAM e Jenkins em Tutorial: Criar um pipeline de quatro estágios.	24 de julho de 2015
Atualização de tópico	Etapas atualizadas foram adicionadas para o download de arquivos de amostra em Tutorial: Criar um pipeline simples (bucket do S3) e Tutorial: Criar um pipeline de quatro estágios.	22 de julho de 2015
Atualização de tópico	Uma solução alternativa temporária para problemas de download com arquivos de amostra foi adicionada a Tutorial: Criar um pipeline simples (bucket do S3).	17 de julho de 2015
Atualização de tópico	Um link foi adicionado a Cotas em AWS CodePipeline para fornecer informações sobre os limites que podem ser alterados.	15 de julho de 2015
Atualização de tópico	A seção de políticas gerenciadas em Autenticação e controle de acesso foi atualizada.	10 de julho de 2015
Lançamento inicial público	Esta é a versão pública inicial do Guia CodePipeline do Usuário.	9 de julho de 2015

Referência de recursos do CodePipeline

Esta seção é uma referência detalhada para recursos na documentação do CodePipeline. Para obter uma visão geral conceitual da estrutura do pipeline, consulte CodePipeline referência de estrutura de tubulação.

Esta tabela é atualizada periodicamente com informações de referência dos recursos.

Data de atualização da tabela	Recurso	Guia do usuário do CodePipeline	Guia de API do CodePipeline	CloudFormationReferência do	Referência do AWS CDK: constructos L1
15 de outubro de 2024	Repetição automática em caso de falha de estágio	Configurar a repetição automática em caso de falha para um estágio	RetryConfiguration		
15 de outubro de 2024	Ignorar resultado para condições beforeEntry	Criar condições de entrada com resultado Ignorar e regra `VariableCheck` (console)	FailureConditions		
3 de outubro de 2024	Ação Commands com a nova categoria Compute	Referência da ação Commands	ActionDeclaration		

Data de atualização da tabela	Recurso	Guia do usuário do CodePipeline	Guia de API do CodePipeline	CloudFormationReferência do	Referência do AWS CDK: constructos L1
28 de agosto de 2024	Tipo beforeEntry para condições de estágio	Configurar condições para um estágio	BeforeEntryConditions	AWS::CodePipeline::Pipeline::BeforeEntryConditions	beforeEntry
28 de agosto de 2024	Tipo onSuccess para condições de estágio	Configurar condições para um estágio	SuccessConditions	AWS::CodePipeline::Pipeline::SuccessConditions	onSuccess
26 de abril de 2024	Reversões de estágio e tipo onFailure para condições de estágio	Configuração da reversão de estágio	RollbackStage	onFailure	onFailure
8 de fevereiro de 2024	Modos de execução PARALLEL e QUEUED	Definir ou alterar o modo de execução do pipeline	PipelineExecution	ExecutionMode	Modo de execução
17 de novembro de 2023	Substituições de origem	Iniciar um pipeline com uma substituição da revisão de origem	SourceRevisionOverride	N/D	N/D

Data de atualização da tabela	Recurso	Guia do usuário do CodePipeline	Guia de API do CodePipeline	CloudFormationReferência do	Referência do AWS CDK: constructos L1
24 de outubro de 2023	Tipos de pipeline	Que tipo de pipeline é ideal para mim?	PipelineDeclaration	PipelineType	enum PipelineType
24 de outubro de 2023	Variáveis em nível de pipeline	Tutorial: Usar variáveis no nível do pipeline	PipelineVariable	AWS::CodePipeline::Pipeline::PipelineVariableDeclaration	Variáveis ao nível do pipeline
24 de outubro de 2023	Gatilhos e filtragem de caminhos de arquivo/ramificações/solicitações pull	Automatizar a inicialização de pipelines usando gatilhos e filtragem Tutorial: Filtrar nomes de ramificação para solicitações pull a fim de iniciar o pipeline (tipo V2)	ActionDeclaration	AWS::CodePipeline::Pipeline::PipelineTriggerDeclaration	Acionador

Data de atualização da tabela	Recurso	Guia do usuário do CodePipeline	Guia de API do CodePipeline	CloudFormationReferência do	Referência do AWS CDK: constructos L1
17 de outubro de 2023	Repetir um estágio	Configurar a repetição do estágio para um estágio com falha ou ações com falha	ActionDeclaration	N/D	N/D

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.