Guide de l’utilisateur

AWS CodePipeline



Version de l'API 2015-07-09

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

AWS CodePipeline: Guide de l’utilisateur

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

Les marques et la présentation commerciale d’Amazon ne peuvent pas être utilisées en relation avec un produit ou un service qui n’appartient pas à Amazon, de toute manière susceptible de créer une confusion chez les clients ou de toute manière dénigrant ou discréditant Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce que c'est CodePipeline ? 1
Livraison continue et intégration continue 1
Qu'est-ce que je peux faire avec CodePipeline ? 2
Un rapide coup d'œil à CodePipeline 3
Comment puis-je commencer CodePipeline ? 4
Concepts 4
Pipelines 5
Exécutions de pipeline 7
Opérations scéniques 8
Exécutions d'actions 8
Types d'exécution 9
Types d'action 9
Artefacts 9
Révisions source 10
Triggers 10
Variables 11
Conditions 11
Rules 12
DevOps exemple de pipeline 12
Fonctionnement des exécutions de pipeline 14
Démarrage des exécutions de pipeline 15
Comment les révisions de source sont traitées dans les exécutions de pipelines 15
Fonctionnement des remplacements de source avec le transformateur EventBridge d'entrée 16
Arrêt des exécutions de pipeline 18
Comment les exécutions sont traitées en mode SUPERSEDED 21
Comment les exécutions sont traitées en mode QUEUED 22
Comment les exécutions sont traitées en mode PARALLÈLE 24
Gestion du débit des canalisations 24
Artefacts d'entrée et de sortie 27
Comment fonctionnent les conditions scéniques ? 30
Règles relatives aux conditions de stage 34
Types de canalisations 35
Quel type de pipeline me convient le mieux ? 35

Prise en main 41
Inscrivez-vous pour un Compte AWS 41
Appliquer une politique gérée pour l'accès administratif à CodePipeline 42
Installer la AWS CLI 43
Ouvrez la console pour CodePipeline 44
Étapes suivantes 45
Intégrations de produits et services 46
Intégrations avec les types CodePipeline d'action 46
Intégrations d'actions source 46
Intégrations d'actions de génération 54
Intégrations d'actions de test 56
Intégrations d'actions de déploiement 59
Intégration des actions d'approbation avec Amazon Simple Notification Service 65
Intégrations d'actions d'appel 66
Intégrations générales avec CodePipeline 67
Exemples issus de la communauté 71
Billets de blogs 71
Tutoriels 76
Tutoriel : Déploiement sur des instances Amazon EC2 avec CodePipeline 77
Conditions préalables 77
Étape 1 : Création d'instances Amazon EC2 Linux 78
Étape 2 : ajouter des autorisations de compartiment d'artefacts au rôle d'instance EC2 80
Étape 3 : ajouter un fichier de script à votre référentiel 81
Étape 4 : Création de votre pipeline 82
Étape 5 : Testez votre pipeline 86
Tutoriel : Créez et transférez une image Docker vers Amazon ECR avec CodePipeline (type V2) 87
Conditions préalables 88
Étape 1 : ajoutez un Dockerfile à votre référentiel source 89
Étape 2 : ajoutez un fichier imagedefinitions.json à votre référentiel source 90
Étape 3 : Création de votre pipeline 91
Étape 4 : Test de votre pipeline 97
Tutoriel : Déploiement sur Amazon EKS avec CodePipeline 98
Conditions préalables 98
Étape 1 : (Facultatif) Création d'un cluster dans Amazon EKS 99
Étape 2 : configurer votre cluster privé dans Amazon EKS 102

Étape 3 : mettre à jour la politique CodePipeline de rôle de service dans IAM 102
Étape 4 : Création d'une entrée d'accès pour le rôle CodePipeline de service 105
Étape 5 : Créez un référentiel source et ajoutez les fichiers de `configuration du graphique` Helm 106
Étape 6 : Création de votre pipeline 107
Tutoriel : Création d'un pipeline qui exécute des commandes avec compute (type V2) 109
Conditions préalables 110
Étape 1 : Créez des fichiers sources et transférez-les vers votre GitHub référentiel 110
Étape 2 : Créer votre pipeline 111
Étape 3 : Exécutez votre pipeline et vérifiez les commandes de construction 115
Tutoriel : Utiliser les balises Git pour démarrer votre pipeline 116
Conditions préalables 117
Étape 1 : Ouvrez CloudShell et clonez votre dépôt 117
Étape 2 : Création d'un pipeline à déclencher sur les balises Git 118
Étape 3 : Marquez vos commits en vue de leur publication 122
Étape 4 : publier les modifications et afficher les journaux 123
Tutoriel : Filtrer les noms de branches pour les pull requests afin de démarrer votre pipeline (type V2) 124
Conditions préalables 124
Étape 1 : Création d'un pipeline à démarrer lors d'une demande d'extraction pour des branches spécifiées 125
Étape 2 : Créez et fusionnez une pull request GitHub.com pour démarrer les exécutions de votre pipeline 127
Tutoriel : Utiliser des variables au niveau du pipeline 129
Conditions préalables 129
Étape 1 : Créez votre pipeline et créez un projet 129
Étape 2 : publier les modifications et afficher les journaux 133
Didacticiel : Création d'un pipeline simple (compartiment S3) 134
Création d'un compartiment S3 135
Créez des instances Amazon EC2 pour Windows Server et installez l'agent CodeDeploy 137
Créez une application dans CodeDeploy 140
Création de votre premier pipeline 142
Ajout d'une autre étape 144
Activation et désactivation des transitions entre les étapes 152
nettoyer des ressources ; 153
Tutoriel : Création d'un pipeline simple (CodeCommit référentiel) 154

Création d'un CodeCommit référentiel 155
Téléchargement, validation et envoi (push) de votre code 156
Créez une instance Amazon EC2 Linux et installez l'agent CodeDeploy 159
Créez une application dans CodeDeploy 161
Création de votre premier pipeline 163
Mettre à jour le code dans votre CodeCommit référentiel 166
nettoyer des ressources ; 168
Suggestions de lecture 168
Didacticiel : Création d'un pipeline à quatre étapes 169
Exécuter les opérations prérequises 170
Création d'un pipeline 175
Ajout d'étapes supplémentaires 177
nettoyer des ressources ; 180
Tutoriel : Configuration d'une règle d' CloudWatch événements pour recevoir des notifications par e-mail en cas de modification de l'état du pipeline 181
Configurer une notification par e-mail à l'aide d'Amazon SNS 182
Créer une règle de notification d' CloudWatch événements pour CodePipeline 183
nettoyer des ressources ; 185
Tutoriel : Créez et testez une application Android avec AWS Device Farm 185
Configurez CodePipeline pour utiliser vos tests Device Farm 187
Tutoriel : Testez une application iOS avec AWS Device Farm 192
Configurer CodePipeline pour utiliser vos tests Device Farm (exemple Amazon S3) 193
Tutoriel : Création d'un pipeline qui se déploie vers Service Catalog 198
Option 1 : déployer vers Service Catalog sans fichier de configuration 199
Option 2 : déployer vers Service Catalog à l'aide d'un fichier de configuration 204
Tutoriel : Création d'un pipeline avec AWS CloudFormation 209
Exemple 1 : créer un AWS CodeCommit pipeline avec AWS CloudFormation 210
Exemple 2 : créer un pipeline Amazon S3 avec AWS CloudFormation 212
Tutoriel : Création d'un pipeline qui utilise des variables issues des actions de AWS CloudFormation déploiement 216
Prérequis : créer un AWS CloudFormation rôle de service et CodeCommit référentiel 217
Étape 1 : Téléchargez, modifiez et chargez l'échantillon AWS CloudFormation modèle 218
Étape 2 : Créer votre pipeline 219
Étape 3 : Ajoutez un CloudFormation action de déploiement pour créer l'ensemble de modifications 222
Étape 4 : Ajouter une action d'approbation manuelle 223

Étape 5 : Ajouter une action CloudFormation de déploiement pour exécuter l'ensemble de modifications 223
Étape 6 : Ajouter une action CloudFormation de déploiement pour supprimer la pile 224
Tutoriel : Déploiement standard d'Amazon ECS avec CodePipeline 225
Conditions préalables 226
Étape 1 : Ajout d'un fichier de spécification de génération dans votre référentiel source 229
Étape 2 : Création de votre pipeline de déploiement continu 231
Étape 3 : ajouter des autorisations Amazon ECR au rôle CodeBuild 233
Étape 4 : Test de votre pipeline 233
Tutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR 234
Conditions préalables 236
Étape 1 : Création d'une image et transfert vers un référentiel Amazon ECR 236
Étape 2 : créer une définition de tâche et des fichiers AppSpec sources et les transférer vers un CodeCommit référentiel 238
Étape 3 : Créer votre équilibreur de charge d'application et vos groupes cibles 242
Étape 4 : Création de votre cluster et de votre service Amazon ECS 245
Étape 5 : Création de votre CodeDeploy application et de votre groupe de déploiement (plateforme de calcul ECS) 247
Étape 6 : Créer le pipeline 249
Étape 7 : Modifier le pipeline et vérifier le déploiement 253
Didacticiel : Création d'un pipeline qui déploie un kit Amazon Alexa Skill 253
Conditions préalables 254
Étape 1 : Créer un profil de sécurité LWA des services de développement Alexa 254
Étape 2 : créer des fichiers source de compétences Alexa et les transférer vers votre CodeCommit référentiel 255
Étape 3 : Utiliser les commandes CLI ASK pour créer un jeton d'actualisation 257
Étape 4 : Créer votre pipeline 257
Étape 5 : Modifier un fichier source et vérifier le déploiement 260
Tutoriel : Création d'un pipeline qui utilise Amazon S3 comme fournisseur de déploiement 261
Option 1 : déployer des fichiers de site Web statiques sur Amazon S3 262
Option 2 : déployer des fichiers d'archive créés sur Amazon S3 à partir d'un compartiment source S3 267
Tutoriel : Publier des applications sur AWS Serverless Application Repository 273
Avant de commencer 274
Étape 1 : Créer un fichier buildspec.yml 275

Étape 2 : Créer et configurer votre pipeline 275
Étape 3 : Déployer l'application de publication 278
Étape 4 : Créer l'action de publication 278
Tutoriel : Déploiements de fonctions Lambda avec CodePipeline 279
Conditions préalables 279
Étape 1 : Création de votre exemple de fonction Lambda 280
Étape 2 : Téléchargez le fichier de fonctions dans votre référentiel 281
Étape 3 : Création de votre pipeline 281
Étape 4 : Test de votre pipeline 284
En savoir plus 285
Tutoriel : Utilisation de variables avec des actions d'invocation Lambda 285
Conditions préalables 286
Étape 1 : créer une fonction Lambda 286
Étape 2 : ajoutez une action d'appel Lambda et une action d'approbation manuelle à votre pipeline 289
Tutoriel : Utiliser une action AWS Step Functions d'appel 291
Condition préalable : créer ou choisir un pipeline simple 292
Étape 1 : Créer l'exemple de machine d'état 292
Étape 2 : Ajouter une action d'appel Step Functions à votre pipeline 292
Tutoriel : Création d'un pipeline utilisé AppConfig comme fournisseur de déploiement 294
Conditions préalables 294
Étape 1 : Créez vos AWS AppConfig ressources 295
Étape 2 : télécharger des fichiers dans votre compartiment source S3 295
Étape 3 : Créer votre pipeline 296
Étape 4 : apporter une modification à n'importe quel fichier source et vérifier le déploiement 298
Tutoriel : Utiliser un clone complet avec une source de GitHub pipeline 298
Conditions préalables 299
Étape 1 : Création d'un fichier README 300
Étape 2 : Créez votre pipeline et créez votre projet 300
Étape 3 : mettre à jour la politique des rôles de CodeBuild service pour utiliser les connexions 304
Étape 4 : Afficher les commandes du référentiel dans la sortie de compilation 304
Tutoriel : Utiliser un clone complet avec une source de CodeCommit pipeline 305
Conditions préalables 306
Étape 1 : Création d'un fichier README 306

Étape 2 : Créez votre pipeline et créez un projet 306
Étape 3 : mettre à jour la politique CodeBuild de rôle de service pour cloner le référentiel 309
Étape 4 : Afficher les commandes du référentiel dans la sortie de compilation 310
Tutoriel : Création d'un pipeline avec des actions AWS CloudFormation StackSets de déploiement 310
Conditions préalables 311
Étape 1 : Téléchargez l'échantillon AWS CloudFormation modèle et fichier de paramètres .. 312
Étape 2 : Créer votre pipeline 314
Étape 3 : Afficher le déploiement initial 317
Étape 4 : ajouter une CloudFormationStackInstances action 317
Étape 5 : Afficher les ressources du stack set pour votre déploiement 318
Étape 6 : effectuez une mise à jour de votre stack set 319
Création d'une règle de vérification variable pour un pipeline en tant que condition d'entrée 320
Conditions préalables 321
Étape 1 : créer un exemple de fichier source et l'ajouter à votre GitHub référentiel 321
Étape 2 : Créer votre pipeline 322
Étape 2 : modifier l'étape de construction pour ajouter la condition et la règle 325
Étape 3 : Exécuter le pipeline et afficher les variables résolues 327
Bonnes pratiques et cas d'utilisation 330
Exemples d'utilisation CodePipeline 330
À utiliser CodePipeline avec Amazon S3, AWS CodeCommit, et AWS CodeDeploy 330
Utilisation CodePipeline avec des fournisseurs d'actions tiers (GitHub et Jenkins) 331
CodePipeline À utiliser pour compiler, construire et tester du code avec CodeBuild 332
CodePipeline À utiliser avec Amazon ECS pour la livraison continue d'applications basées sur des conteneurs vers le cloud 332
Utilisation CodePipeline avec Elastic Beanstalk pour la diffusion continue d'applications Web dans le cloud 332
À utiliser CodePipeline avec AWS Lambda pour la livraison continue Lambda-based et les applications sans serveur 332
À utiliser CodePipeline avec CloudFormation modèles pour une diffusion continue dans le cloud 333
Utilisation CodePipeline avec Amazon VPC 334
Disponibilité 334
Créez un point de terminaison VPC pour CodePipeline 335
Dépannage de la configuration de votre VPC 336
Définissez des CI/CD pipelines avec des étapes et des actions 337

Création d'un pipeline, d'étapes et d'actions 338
Création d'un pipeline personnalisé (console) 339
Création d'un pipeline (interface de ligne de commande) 353
Création d'un pipeline à partir de modèles statiques 359
Modification d'un pipeline 366
Modification d'un pipeline (console) 367
Modification d'un pipeline (AWS CLI) 371
Afficher les pipelines et les détails 375
Afficher les pipelines (console) 375
Afficher les détails des actions dans un pipeline (console) 380
Afficher l'ARN du pipeline et l'ARN du rôle de service (console) 383
Affichage des détails et de l'historique d'un pipeline (interface de ligne de commande) 384
Afficher les résultats des règles pour les conditions d'étape dans l'historique d'exécution 385
Supprime un pipeline. 388
Suppression d'un pipeline (console) 388
Suppression d'un pipeline (interface de ligne de commande) 388
Création d'un pipeline qui utilise des ressources d'un autre compte 390
Prérequis : créer un AWS KMS clé de chiffrement 392
Étape 1 : Configurer des stratégies de compte et des rôles 393
Étape 2 : Modifier le pipeline 400
Migrez les pipelines de sondage pour utiliser la détection des modifications basée sur les événements 403
Comment migrer les pipelines de sondages 404
Afficher les pipelines de sondages dans votre compte 405
Migrer les pipelines de sondage avec une CodeCommit source 411
Migrer les pipelines de sondage avec une source S3 activée pour les événements 433
Migrer les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3 453
Migrer les pipelines de sondage pour une action source GitHub (via l'application OAuth) vers les connexions 489
Migrer les pipelines de sondage pour une action source GitHub (via l'application OAuth) vers des webhooks 493
Création du rôle CodePipeline de service 511
Création du rôle CodePipeline de service (console) 511
Création du rôle CodePipeline de service (CLI) 512
Balisage de ressources 514
Balisage d'un pipeline 515

Balisage de pipelines (console) 516
Balisage de pipelines (interface de ligne de commande) 518
Création d'une règle de notification 520
..... 524
Connectez-vous à des fournisseurs de sources propriétaires à l'aide d'actions de source 527
actions relatives aux sources Amazon ECR et EventBridge 527
Créer une EventBridge règle pour une source Amazon ECR (console) 528
Création d'une EventBridge règle pour une source Amazon ECR (CLI) 531
Créez une EventBridge règle pour une source Amazon ECR (CloudFormation modèle) 534
Connexion à la source Amazon S3 : actions utilisant EventBridge des événements 540
Créez des pipelines avec une source S3 activée pour les événements (CLI) 541
Créez des pipelines avec une source S3 activée pour les événements (CloudFormation modèle) 545
Connexion à la source Amazon S3 : actions qui utilisent EventBridge et AWS CloudTrail 560
Création d'une EventBridge règle pour une source Amazon S3 (console) 561
Création d'une EventBridge règle pour une source Amazon S3 (CLI) 565
Créez une EventBridge règle pour une source Amazon S3 (CloudFormation modèle) 572
CodeCommit actions à la source et EventBridge 584
Création d'une EventBridge règle pour une CodeCommit source (console) 585
Création d'une EventBridge règle pour une CodeCommit source (CLI) 588
Créez une EventBridge règle pour une CodeCommit source (CloudFormation modèle) 593
Ajoutez des fournisseurs de sources tiers aux pipelines à l'aide de AWS CodeConnections 602
DevOps Connexions Azure 602
Création d'une connexion à Azure DevOps (console) 604
Création d'une connexion à Azure DevOps (CLI) 608
Connexions Bitbucket Cloud 610
Création d'une connexion à Bitbucket Cloud (console) 612
Création d'une connexion à Bitbucket Cloud (CLI) 615
GitHub connexions 617
Création d'une connexion à GitHub (console) 619
Création d'une connexion à GitHub (CLI) 622
GitHub Connexions aux serveurs d'entreprise 624
Création d'une connexion à GitHub Enterprise Server (console) 625
Création d'un hôte et d'une connexion à GitHub Enterprise Server (CLI) 629
GitLab.com connexions 632
Création d'une connexion à GitLab.com (console) 635

Création d'une connexion à GitLab.com (CLI) 639
Connexions pour l' GitLab autogestion 641
Création d'une connexion à l' GitLab autogestion (console) 643
Création d'un hôte et d'une connexion à une interface GitLab autogérée (CLI) 646
Utiliser une connexion partagée avec un autre compte 649
Automatisez le démarrage des pipelines en utilisant des déclencheurs et des filtres 650
Considérations relatives aux filtres déclencheurs 652
Événements de pull request pour les déclencheurs par fournisseur 653
Exemples de filtres déclencheurs 654
Ajouter un déclencheur lors de l'envoi de code sans filtre 663
Ajouter un déclencheur avec des types d'événements de type « code push » ou « pull request » 664
Ajouter des filtres pour les types d'événements de type push et pull request (console) 665
Ajouter des filtres pour les types d'événements de requêtes push et pull (CLI) 668
Ajoutez des filtres pour les types d'événements de requêtes push et pull (CloudFormation modèles) 671
Ajouter un déclencheur pour désactiver la détection des modifications 672
Démarrage et arrêt manuels des pipelines 674
Démarrez un pipeline dans CodePipeline 674
Lancement manuel d'un pipeline 676
Démarrez un pipeline selon un calendrier 678
Démarrer un pipeline avec un remplacement de révision de la source 681
Arrêt de l'exécution d'un pipeline 684
Arrêt de l'exécution d'un pipeline (console) 685
Arrêter une exécution entrante (console) 688
Arrêt de l'exécution d'un pipeline (interface de ligne de commande) 689
Arrêter une exécution entrante (CLI) 690
Afficher l'historique et définir le mode d'exécution des pipelines 692
Afficher les exécutions 692
Affichage de l'historique d'exécution du pipeline (console) 692
Affichage du statut d'exécution (console) 694
Afficher une exécution entrante (console) 696
Affichage des révisions de la source d'exécution du pipeline (console) 697
Affichage des exécutions des actions (console) 699
Affichage des artefacts d'action et des informations de stockage des artefacts (console) 700
Affichage des détails et de l'historique d'un pipeline (interface de ligne de commande) 700

Définir ou modifier le mode d'exécution du pipeline 713
Considérations relatives à l'affichage des modes d'exécution 713
Considérations relatives au passage d'un mode d'exécution à un autre 716
Définir ou modifier le mode d'exécution du pipeline (console) 717
Définissez le mode d'exécution du pipeline (CLI) 718
Annuler ou réessayer les étapes 722
Configuration d'une nouvelle tentative d'étape pour une étape ayant échoué ou des actions ayant échoué 722
Considérations relatives à une nouvelle tentative 723
Réessayer manuellement une étape ayant échoué 723
Configurer une étape pour une nouvelle tentative automatique en cas d'échec 728
Configuration de la restauration des étapes 734
Considérations relatives aux annulations 734
Faire reculer une étape manuellement 734
Configurer une étape pour la restauration automatique 740
Afficher l'état de restauration dans la liste des exécutions 744
Afficher les détails de l'état de restauration 747
Configuration des conditions pour une étape 752
Cas d'utilisation pour les conditions scéniques 753
Considérations relatives aux résultats configurés pour les conditions de l'étape 753
Considérations relatives aux règles configurées pour les conditions de l'étape 754
Création de conditions d'entrée 755
Création de conditions d'entrée - exemple de CloudWatchAlarm règle (console) 755
Création de conditions de saisie avec Ignorer le résultat et la `VariableCheck` règle (console) 756
Création de conditions d'entrée (CLI) 758
Création de conditions d'entrée (CFN) 760
Création de conditions en cas de défaillance 761
Création de conditions en cas de défaillance (console) 761
Exemple de création de conditions OnFailure avec un résultat de nouvelle tentative (console) 762
Création de conditions en cas de défaillance (CLI) 764
Création de conditions en cas de défaillance (CFN) 765
Créer des conditions de réussite 767
Création selon les conditions de réussite (console) 767
Création de conditions de réussite (CLI) 769

Création d'une condition de réussite (CFN) 770
Supprimer les conditions de l'étape 771
Conditions de phase dérogatoires 772
Utiliser des types d'actions, des actions personnalisées et des actions d'approbation 775
Utilisation des types d'actions 776
Demander un type d'action 777
Ajouter un type d'action disponible à un pipeline (console) 783
Afficher un type d'action 785
Mettre à jour un type d'action 787
Création d'une action personnalisée pour un pipeline 789
Création d'une action personnalisée 791
Création d'un exécutant de tâches pour l'action personnalisée 795
Ajout d'une action personnalisée à un pipeline 804
Marquer une action personnalisée dans CodePipeline 807
Ajout de balises à une action personnalisée 808
Affichage des balises pour une action personnalisée 808
Modification des balises d'une action personnalisée 809
Suppression de balises d'une action personnalisée 809
Invoquer une fonction Lambda dans un pipeline 810
Étape 1 : Créer un pipeline 812
Étape 2 : Création de la fonction Lambda 813
Étape 3 : ajouter la fonction Lambda à un pipeline dans la console CodePipeline 818
Étape 4 : Testez le pipeline à l'aide de la fonction Lambda 819
Étape 5 : étapes suivantes 819
Exemple d'événement JSON 820
Autres modèles de fonctions 822
Ajouter une action d'approbation manuelle à une étape 835
Options de configuration pour les actions d'approbation manuelle 836
Présentation de la configuration et du flux de travail des actions d'approbation 837
Accorder des autorisations d'approbation à un utilisateur IAM dans CodePipeline 838
Accorder des autorisations Amazon SNS à un rôle de service 841
Ajout d'une action d'approbation manuelle 843
Approbation ou rejet d'une action d'approbation 847
Format de données JSON pour les notifications d'approbation manuelle 851
Ajouter une action interrégionale à un pipeline 852
Gérer les actions interrégionales dans un pipeline (console) 854

Ajouter une action interrégionale à un pipeline (CLI) 857
Ajoutez une action interrégionale à un pipeline (CloudFormation) 863
Utilisation des variables 865
Configuration d'actions pour des variables 866
Affichage des variables de sortie 870
Exemple : Utiliser des variables dans les approbations manuelles 873
Exemple : utilisation d'une BranchName variable avec des variables d' CodeBuild environnement 874
Utilisation de transitions entre les étapes 876
Désactivation ou activation des transitions (console) 876
Désactivation ou activation des transitions (interface de ligne de commande) 879
Surveillance des pipelines 881
Surveillance des CodePipeline événements 882
Types de détails 884
Pipeline-level événements 886
Stage-level événements 895
Action-level événements 899
Création d'une règle qui envoie une notification concernant un événement de pipeline 907
Référence des compartiments d'espace réservé pour les événements 911
Noms des compartiments d'espace réservé pour les événements par région 913
Journalisation des appels d'API AWS CloudTrail avec 916
CodePipeline informations dans CloudTrail 916
Comprendre les entrées des fichiers CodePipeline journaux 917
CodePipeline CloudWatch métriques 919
PipelineDuration 920
FailedPipelineExecutions 920
Résolution des problèmes 921
Erreur de pipeline : un pipeline configuré avec AWS Elastic Beanstalk renvoie un message d'erreur : « Le déploiement a échoué. Le rôle fourni ne dispose pas d'autorisations suffisantes : Service:AmazonElasticLoadBalancing » 922
Erreur de déploiement : un pipeline configuré avec un AWS Elastic Beanstalk l'action de déploiement se bloque au lieu d'échouer si l'autorisation « DescribeEvents » est manquante ... 923
Erreur de pipeline : une action source renvoie le message d'autorisations insuffisantes : « Impossible d'accéder au nom du CodeCommit référentiel du `référentiel`. Assurez-vous que le rôle IAM du pipeline dispose des autorisations suffisantes pour accéder au référentiel. » 923

Erreur de pipeline : Une action Jenkins de génération ou de test s'exécute pendant une longue durée puis échoue, en raison d'informations d'identification ou d'autorisations insuffisantes 924
Erreur de pipeline : un pipeline créé en un AWS Région utilisant un bucket créé dans un autre AWS La région renvoie un « InternalError » avec le code « JobFailed » 925
Erreur de déploiement : un fichier ZIP contenant un fichier WAR a été déployé avec succès sur AWS Elastic Beanstalk, mais l'URL de l'application indique une erreur 404 introuvable 923
Les noms des dossiers d'artefact du pipeline semblent tronqués 926
Ajoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com 926
Ajouter CodeBuild GitClone des autorisations pour les actions CodeCommit source 928
Erreur de pipeline : un déploiement avec l' CodeDeployToECS action renvoie un message d'erreur : « Exception lors de la tentative de lecture du fichier d'artefact de définition de tâche depuis : nom de l'artefact < source » > 929
GitHub (via l'application OAuth) action source : la liste des référentiels montre les différents référentiels 930
GitHub Action source (via GitHub l'application) : impossible de terminer la connexion pour un référentiel 930
Erreur Amazon S3 : l'accès au compartiment S3 > est refusé à l'<ARN du rôle de CodePipeline service ARN < BucketName > 931
Les pipelines dotés d'un Amazon S3, d'Amazon ECR ou d' CodeCommit une source ne démarrent plus automatiquement 933
Erreur de connexion lors de la connexion à GitHub : « Un problème est survenu, assurez-vous que les cookies sont activés dans votre navigateur » ou « Le propriétaire d'une organisation doit installer l' GitHub application » 935
Les pipelines dont le mode d'exécution est passé en mode QUEUED ou PARALLEL échouent lorsque la limite d'exécution est atteinte 935
Les pipelines en mode PARALLÈLE ont une définition de pipeline obsolète s'ils sont modifiés lors du passage en mode QUEUED ou SUPERSEDED 936
Les pipelines passés du mode PARALLÈLE afficheront un mode d'exécution précédent 936
Les pipelines avec des connexions qui utilisent le filtrage des déclencheurs par chemin de fichier peuvent ne pas démarrer lors de la création de la branche 937
Les pipelines dont les connexions utilisent le filtrage des déclencheurs par chemin de fichier risquent de ne pas démarrer lorsque la limite de fichiers est atteinte 937
CodeCommit ou les révisions de source S3 en mode PARALLÈLE peuvent ne pas correspondre à EventBridge l'événement 938

L'action de déploiement EC2 échoue avec un message d'erreur `Aucun fichier de ce type` 938
L'action EKS Deploy échoue avec un message d'erreur indiquant que le `cluster est inaccessible` 939
Le pipeline ne se déclenche pas pour toutes les branches lorsque plusieurs actions de source font référence au même référentiel 940
Besoin d'aide pour résoudre un autre problème ? 940
Sécurité 942
Protection des données 943
Confidentialité du trafic inter-réseau 944
Chiffrement au repos 945
Chiffrement en transit 945
Gestion des clés de chiffrement 945
Configurer le chiffrement côté serveur pour les artefacts stockés dans Amazon S3 pour CodePipeline 945
Utilisation AWS Secrets Manager pour suivre les mots de passe de base de données ou les clés d'API tierces 949
Gestion des identités et des accès 949
Public ciblé 950
Authentification par des identités 950
Gestion de l'accès à l'aide de politiques 951
Comment ? AWS CodePipeline fonctionne avec IAM 953
Identity-based exemples de politiques 959
Resource-based exemples de politiques 988
Résolution des problèmes 990
référence des autorisations 992
Gérer le rôle CodePipeline de service 1003
Intervention en cas d'incidents 1010
Validation de conformité 1010
Résilience 1011
Sécurité de l'infrastructure 1011
Bonnes pratiques de sécurité 1012
Référence sur la structure des pipelines 1013
Déclaration sur le pipeline 1017
`name` 1019
`roleArn` 1020

ArtifactStore OU ArtifactStores 1020
étapes 1021
version 1022
executionMode 1022
Type de pipeline 1023
variables 1023
déclencheurs 1023
metadata 1028
Déclaration d'étape 1029
name 1031
actions 1031
conditions 1032
rules 1032
Déclaration d'action 1032
name 1036
region 1037
roleArn 1037
espace de nom 1037
action TypeId 1038
InputArtifacts 1039
Artefacts de sortie 1040
configuration (par fournisseur d'actions) 1041
Ordre d'exécution 1043
Fournisseurs d'actions valides dans CodePipeline 1043
Réglages valides pour le PollForSourceChanges paramètre 1049
Artefacts d'entrée et de sortie valides pour chaque type d'action 1051
Paramètres de configuration valides pour chaque type de fournisseur 1053
Référence sur la structure des actions 1057
Référence des actions Amazon EC2 1058
Type d'action 1059
Paramètres de configuration 1059
Artefacts d'entrée 1063
Artefacts de sortie 1064
Autorisations relatives à la politique de rôle de service pour l'action de déploiement d'EC2 1064
Référence du fichier de spécifications du déploiement 1067
Déclaration d'action 1068

Déclaration d'action avec exemple de spécification Deploy 1070
Consultez aussi 1071
Référence des actions source Amazon ECR 1071
Type d'action 1072
Paramètres de configuration 1072
Artefacts d'entrée 1073
Artefacts de sortie 1073
Variables de sortie 1073
Autorisations relatives aux rôles de service : action Amazon ECR 1074
Déclaration d'action (exemple Amazon ECR) 1074
Consultez aussi 1076
`ECRBuildAndPublish`créer une référence d'action 1076
Type d'action 1077
Paramètres de configuration 1077
Artefacts d'entrée 1078
Artefacts de sortie 1078
Variables de sortie 1078
Autorisations relatives aux rôles de service : `ECRBuildAndPublish` action 1079
Déclaration d'action 1081
Consultez aussi 1082
Référence des actions de déploiement Amazon ECS et CodeDeploy bleu-vert 1082
Type d'action 1083
Paramètres de configuration 1083
Artefacts d'entrée 1085
Artefacts de sortie 1086
Autorisations relatives aux rôles de service : `CodeDeployToECS` action 1086
Déclaration d'action 1088
Consultez aussi 1090
Référence des actions de déploiement d'Amazon Elastic Container Service 1091
Type d'action 1092
Paramètres de configuration 1092
Artefacts d'entrée 1093
Artefacts de sortie 1093
Autorisations relatives aux rôles de service : action standard Amazon ECS 1093
Déclaration d'action 1095
Consultez aussi 1096

Référence des actions de déploiement d'Amazon Elastic Kubernetes Service EKS 1097
Type d'action 1098
Paramètres de configuration 1098
Artefacts d'entrée 1100
Artefacts de sortie 1100
Variables d'environnement 1100
Variables de sortie 1100
Autorisations relatives aux politiques de rôle de service 1101
Déclaration d'action 1103
Consultez aussi 1104
AWS Lambda référence des actions de déploiement 1104
Type d'action 1105
Paramètres de configuration 1105
Artefacts d'entrée 1109
Artefacts de sortie 1109
Variables de sortie 1109
Autorisations relatives à la politique de rôle de service pour l'action de déploiement Lambda 1109
Déclaration d'action 1111
Consultez aussi 1112
Référence des actions de déploiement d'Amazon S3 1112
Type d'action 1113
Paramètres de configuration 1113
Artefacts d'entrée 1115
Artefacts de sortie 1115
Autorisations relatives aux rôles de service : action de déploiement S3 1115
Exemple de configuration d'action 1116
Consultez aussi 1119
Référence des actions source d'Amazon S3 1119
Type d'action 1120
Paramètres de configuration 1120
Artefacts d'entrée 1122
Artefacts de sortie 1123
Variables de sortie 1123
Autorisations relatives aux rôles de service : action source S3 1123
Déclaration d'action 1124

Consultez aussi 1126
AWS AppConfig référence d'action de déploiement 1126
Type d'action 1126
Paramètres de configuration 1126
Artefacts d'entrée 1127
Artefacts de sortie 1127
Autorisations relatives aux rôles de service : AppConfig action 1127
Exemple de configuration d'action 1128
Consultez aussi 1129
CloudFormation référence d'action de déploiement 1130
Type d'action 1131
Paramètres de configuration 1131
Artefacts d'entrée 1136
Artefacts de sortie 1136
Variables de sortie 1136
Autorisations relatives aux rôles de service : CloudFormation action 1137
Déclaration d'action 1138
Consultez aussi 1140
CloudFormation StackSets 1140
Comment ? CloudFormation StackSets les actions fonctionnent 1141
Comment structurer StackSets les actions dans un pipeline 1143
L'action CloudFormationStackSet 1145
L' CloudFormationStackInstances action 1159
Autorisations relatives aux rôles de service : CloudFormationStackSet action 1169
Autorisations relatives aux rôles de service : CloudFormationStackInstances action 1170
Modèles d'autorisations pour les opérations relatives aux ensembles de piles 1170
Types de données des paramètres du modèle 1171
Consultez aussi 1140
AWS CodeBuild créer et tester une référence d'action 1173
Type d'action 1174
Paramètres de configuration 1174
Artefacts d'entrée 1177
Artefacts de sortie 1178
Variables de sortie 1179
Autorisations relatives aux rôles de service : CodeBuild action 1179
Déclaration d'action (CodeBuild exemple) 1180

Consultez aussi 1182
AWS CodePipeline invoquer la référence de l'action 1182
Type d'action 1183
Paramètres de configuration 1183
Artefacts d'entrée 1185
Artefacts de sortie 1186
Autorisations relatives à la politique de rôle de service pour l'action d' CodePipeline invocation 1186
Déclaration d'action 1186
Consultez aussi 1187
AWS CodeCommit référence de l'action source 1187
Type d'action 1189
Paramètres de configuration 1189
Artefacts d'entrée 1190
Artefacts de sortie 1191
Variables de sortie 1191
Autorisations relatives aux rôles de service : CodeCommit action 1192
Exemple de configuration d'action 1193
Consultez aussi 1195
AWS CodeDeploy référence d'action de déploiement 1195
Type d'action 1196
Paramètres de configuration 1196
Artefacts d'entrée 1196
Artefacts de sortie 1197
Autorisations relatives aux rôles de service : AWS CodeDeploy action 1197
Déclaration d'action 1198
Consultez aussi 1199
CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées 1200
Type d'action 1204
Paramètres de configuration 1204
Artefacts d'entrée 1205
Artefacts de sortie 1206
Variables de sortie 1206
Autorisations relatives aux rôles de service : AWS CodeConnections action 1207
Déclaration d'action 1208

Installation de l'application d'installation et création d'une connexion 1209
Consultez aussi 1209
Référence des actions des commandes 1210
Considérations relatives à l'action des commandes 1211
Autorisations relatives aux politiques de rôle de service 1212
Type d'action 1213
Paramètres de configuration 1214
Artefacts d'entrée 1219
Artefacts de sortie 1219
Variables d'environnement 1219
Autorisations relatives aux rôles de service : action des commandes 1221
Déclaration d'action (exemple) 1221
Consultez aussi 1223
AWS Device Farm référence de l'action de test 1223
Type d'action 1224
Paramètres de configuration 1224
Artefacts d'entrée 1228
Artefacts de sortie 1228
Autorisations relatives aux rôles de service : AWS Device Farm action 1228
Déclaration d'action 1229
Consultez aussi 1230
Référence des actions de déploiement d'Elastic Beanstalk 1231
Type d'action 1231
Paramètres de configuration 1231
Artefacts d'entrée 1232
Artefacts de sortie 1232
Autorisations relatives aux rôles de service : action de `ElasticBeanstalk` déploiement 1232
Déclaration d'action 1233
Consultez aussi 1234
Amazon Inspector `InspectorScan` invoque une référence d'action 1235
ID du type d'action 1235
Paramètres de configuration 1236
Artefacts d'entrée 1239
Artefacts de sortie 1239
Variables de sortie 1239
Autorisations relatives aux rôles de service : `InspectorScan` action 1239

Déclaration d'action 1240
Consultez aussi 1241
AWS Lambda invoquer la référence de l'action 1242
Type d'action 1242
Paramètres de configuration 1242
Artefacts d'entrée 1243
Artefacts de sortie 1243
Variables de sortie 1243
Exemple de configuration d'action 1243
Exemple d'événement JSON 1244
Consultez aussi 1247
AWS OpsWorks référence des actions de déploiement 1247
Type d'action 1247
Paramètres de configuration 1248
Artefacts d'entrée 1248
Artefacts de sortie 1248
Autorisations relatives aux rôles de service : AWS OpsWorks action 1248
Exemple de configuration d'action 1249
Consultez aussi 1250
AWS Service Catalog référence des actions de déploiement 1250
Type d'action 1250
Paramètres de configuration 1250
Artefacts d'entrée 1251
Artefacts de sortie 1251
Autorisations relatives aux rôles de service : action Service Catalog 1251
Exemples de configurations d'actions par type de fichier de configuration 1252
Consultez aussi 1253
AWS Step Functions 1254
Type d'action 1247
Paramètres de configuration 1254
Artefacts d'entrée 1256
Artefacts de sortie 1256
Variables de sortie 1256
Autorisations relatives aux rôles de service : `StepFunctions` action 1257
Exemple de configuration d'action 1257
Comportement 1260

Consultez aussi 1129
Référence de structure de règles 1263
CloudWatchAlarm 1263
Type de règle 1264
Paramètres de configuration 1264
Exemple de configuration de règles 1265
Consultez aussi 1266
CodeBuild règle 1266
Autorisations de politique des rôles de service 1266
Type de règle 1267
Paramètres de configuration 1267
Exemple de configuration de règles 1269
Consultez aussi 1270
Commandes 1270
Considérations relatives à la règle des commandes 1271
Autorisations de politique des rôles de service 1266
Type de règle 1267
Paramètres de configuration 1267
Exemple de configuration de règles 1269
Consultez aussi 1270
DeploymentWindow 1276
Type de règle 1276
Paramètres de configuration 1276
Exemple de configuration de règles 1279
Consultez aussi 1279
LambdaInvoke 1280
Type de règle 1264
Paramètres de configuration 1280
Exemple de configuration de règles 1281
Consultez aussi 1282
VariableCheck 1282
Type de règle 1282
Paramètres de configuration 1282
Exemple de configuration de règles 1285
Consultez aussi 1286
Référence du modèle d'intégration 1287

Comment les types d'actions tiers fonctionnent avec l'intégrateur 1287
Concepts 1288
Modèles d'intégration pris en charge 1290
Modèle d'intégration Lambda 1291
Mettez à jour votre fonction Lambda pour gérer les entrées de CodePipeline 1292
Renvoie les résultats de votre fonction Lambda à CodePipeline 1296
Utilisez des jetons de continuation pour attendre les résultats d'un processus asynchrone 1298
Fournissez CodePipeline les autorisations nécessaires pour invoquer la fonction Lambda de l'intégrateur lors de l'exécution 1299
Modèle d'intégration des travailleurs 1299
Choix et configuration d'une stratégie de gestion des autorisations pour votre exécutant de tâches 1300
Référence pour les fichiers de définitions d'image 1302
fichier imagedefinitions.json pour les actions de déploiement standard d'Amazon ECS 1302
Detail.json fichier image pour les actions de blue/green déploiement d'Amazon ECS 1305
Référence des variables 1310
Concepts 1311
Variables 1311
Espaces de noms 1312
Cas d'utilisation des variables 1313
Configuration des variables 1314
Configuration des variables au niveau du pipeline 1314
Configuration des variables au niveau de l'action 1315
Résolution des variables 1317
Règles pour les variables 1318
Variables disponibles pour les actions de pipeline 1319
Actions avec touches variables définies 1319
Actions à l'aide de touches variables configurées par l'utilisateur 1325
Utilisation de modèles globulaires dans la syntaxe 1328
Mise à jour de pipelines d'interrogation vers la méthode recommandée de détection des modifications 1330
Mettre à jour une action source GitHub (via l'application OAuth) en une action source GitHub (via GitHub l'application) 1331
Étape 1 : remplacez votre action (via l'application OAuth) GitHub 1333
Étape 2 : créer une connexion avec GitHub 1333
Étape 3 : Enregistrez votre action GitHub source 1334

Quotas 1336
Annexe A : actions relatives aux sources GitHub (via l'application OAuth) 1353
Ajouter une action GitHub source (via l'application OAuth) 1354
GitHub (via l'application OAuth) référence de l'action source 1355
Type d'action 1356
Paramètres de configuration 1356
Artefacts d'entrée 1358
Artefacts de sortie 1358
Variables de sortie 1359
Déclaration d'action (GitHub exemple) 1359
Connexion à GitHub (OAuth) 1361
Consultez aussi 1361
Historique de la documentation 1362
Mises à jour antérieures 1401
CodePipeline référence de fonctionnalité 1413
..... mcdxvii

Qu'est-ce que AWS CodePipeline?

AWS CodePipeline est un service de diffusion continue que vous pouvez utiliser pour modéliser, visualiser et automatiser les étapes nécessaires à la publication de votre logiciel. Vous pouvez rapidement modéliser et configurer les différentes étapes du processus de publication d'un logiciel. CodePipeline automatise les étapes nécessaires à la publication continue des modifications apportées à votre logiciel. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.

Rubriques

- Livraison continue et intégration continue
- Qu'est-ce que je peux faire avec CodePipeline ?
- Un rapide coup d'œil à CodePipeline
- Comment puis-je commencer CodePipeline ?
- CodePipeline concepts
- DevOps exemple de pipeline
- Fonctionnement des exécutions de pipeline
- Artefacts d'entrée et de sortie
- Comment fonctionnent les conditions scéniques ?
- Types de canalisations
- Quel type de pipeline me convient le mieux ?

Livraison continue et intégration continue

CodePipeline est un service de livraison continue qui automatise la création, les tests et le déploiement de vos logiciels en production.

La livraison continue est une méthode de développement logiciel où le processus de publication est automatisé. Chaque modification logicielle est automatiquement créée, testée et déployée dans la production. Avant la publication finale vers la production, un utilisateur, un test automatisé ou une règle commerciale détermine quand cette opération doit avoir lieu. Chaque modification logicielle réussie peut être immédiatement publiée dans la production avec une livraison continue, toutefois les modifications n'ont pas toutes besoin d'être publiées immédiatement.

L'intégration continue est une pratique de développement logiciel dans laquelle les membres d'une équipe utilisent un système de contrôle de version et intègrent fréquemment leur travail au même endroit, comme une succursale principale. Chaque modification est créée et vérifiée afin de repérer les erreurs d'intégration aussi vite que possible. L'intégration continue est axée sur le développement automatique et les tests de code, alors que la livraison continue automatise l'intégralité du processus de publication logicielle jusqu'à la production.

Pour plus d'informations, consultez Pratiquer l'intégration et la livraison continues sur AWS : Accélérer la livraison logicielle avec DevOps.

Vous pouvez utiliser la CodePipeline console, le AWS Command Line Interface (AWS CLI), les AWS SDK ou toute combinaison de ces éléments pour créer et gérer vos pipelines.

Qu'est-ce que je peux faire avec CodePipeline ?

Vous pouvez l'utiliser CodePipeline pour créer, tester et déployer automatiquement vos applications dans le cloud. Plus précisément, vous pouvez :

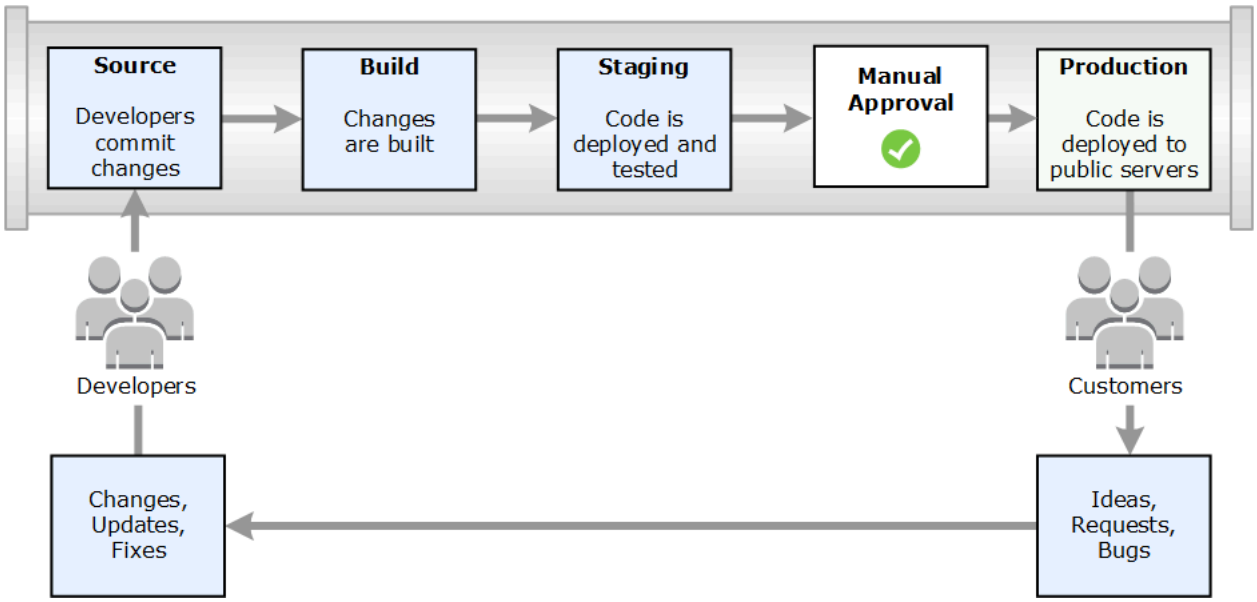
- Automatisez vos processus de publication : automatise CodePipeline entièrement votre processus de publication de bout en bout, en commençant par votre référentiel source, en passant par la génération, les tests et le déploiement. Vous pouvez éviter que des modifications ne transite à travers un pipeline en incluant une action d'approbation manuelle dans n'importe quelle étape, à l'exception de l'étape Source. Vous pouvez publier quand vous le souhaitez, de la manière dont vous le souhaitez, sur les systèmes de votre choix, sur une instance ou plusieurs.
- Établissez un processus de publication cohérent : définissez un ensemble cohérent d'étapes pour chaque modification de code. CodePipeline gère chaque étape de votre sortie en fonction de vos critères.
- Accélérer la livraison tout en améliorant la qualité : vous pouvez automatiser votre processus de publication pour permettre à vos développeurs de tester et de publier un code progressivement, puis d'accélérer le lancement de nouvelles fonctionnalités pour vos clients.
- Utiliser vos outils favoris : vous pouvez incorporer vos outils source, de génération et de déploiement existants à votre pipeline. Pour obtenir la liste complète Services AWS des outils tiers actuellement pris en charge par CodePipeline, consultezIntégrations de produits et de services avec CodePipeline.
- Visualisez la progression en un coup d'œil : vous pouvez consulter l'état en temps réel de vos pipelines, vérifier les détails des alertes, réessayer les étapes ou les actions qui ont échoué,

consulter les détails des révisions de source utilisées lors de la dernière exécution du pipeline à chaque étape et relancer manuellement n'importe quel pipeline.

- Afficher les détails de l'historique du pipeline : vous pouvez consulter les détails des exécutions d'un pipeline, y compris les heures de début et de fin, la durée d'exécution et les ID d'exécution.

Un rapide coup d'œil à CodePipeline

Le schéma suivant montre un exemple de processus de publication à l'aide d' CodePipeline.



Dans cet exemple, lorsque les développeurs valident des modifications apportées à un référentiel source, CodePipeline détecte automatiquement les modifications. Ces modifications sont générées, et si des tests sont configurés, ces derniers sont exécutés. Une fois les tests terminés, le code ainsi généré est déployé sur les serveurs intermédiaires, afin de le tester. À partir du serveur intermédiaire, CodePipeline exécute d'autres tests, tels que des tests d'intégration ou de charge. Une fois ces tests terminés avec succès et après l'approbation d'une action d'approbation manuelle ajoutée au pipeline, CodePipeline déploie le code testé et approuvé sur les instances de production.

CodePipeline peut déployer des applications sur des instances EC2 à l'aide de CodeDeploy AWS Elastic Beanstalk, ou AWS OpsWorks Stacks. CodePipeline peut également déployer des applications basées sur des conteneurs vers des services à l'aide d'Amazon ECS. Les développeurs peuvent également utiliser les points d'intégration fournis CodePipeline pour connecter d'autres outils ou services, notamment des services de création, des fournisseurs de test ou d'autres cibles ou systèmes de déploiement.

Un pipeline peut être très simple ou très complexe, en fonction des besoins de votre processus de publication.

Comment puis-je commencer CodePipeline ?

Pour commencer CodePipeline :

1. Découvrez comment CodePipeline fonctionne en lisant la CodePipeline concepts section.
2. Préparez-vous à l'utiliser CodePipeline en suivant les étapes décrites dansCommencer avec CodePipeline.
3. Faites des essais CodePipeline en suivant les étapes des CodePipeline tutoriels didacticiels.
4. CodePipeline À utiliser pour vos projets nouveaux ou existants en suivant les étapes décrites dansCréation d'un pipeline, d'étapes et d'actions.

CodePipeline concepts

Il est plus facile de modéliser et de configurer votre processus de publication automatisé si vous comprenez les concepts et les termes utilisés dans AWS CodePipeline. Voici quelques concepts à connaître lorsque vous utilisez CodePipeline.

Pour un exemple de DevOps pipeline, voirDevOps exemple de pipeline.

Les termes suivants sont utilisés dans CodePipeline :

Rubriques

- Pipelines
- Exécutions de pipeline
- Opérations scéniques
- Exécutions d'actions
- Types d'exécution
- Types d'action
- Artefacts
- Révisions source
- Triggers
- Variables

- Conditions
- Rules

Pipelines

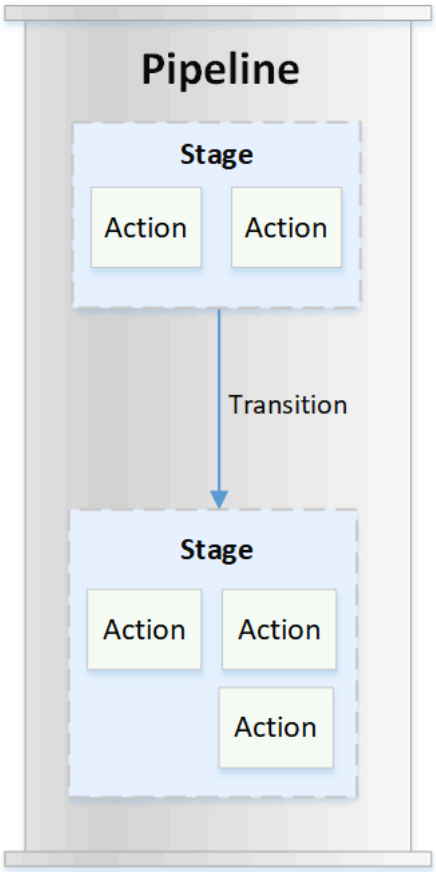
Un pipeline est une structure de workflow qui décrit la progression des modifications logicielles d'un processus de publication. Chaque pipeline est constitué d'une série d'étapes.

Stages

Une étape est une unité logique que vous pouvez utiliser pour isoler un environnement et limiter le nombre de modifications simultanées dans cet environnement. Chaque étape contient des actions effectuées sur les artefactsde l'application. Votre code source est un exemple d'artefact. Une étape peut être une étape de construction, au cours de laquelle le code source est généré et les tests sont exécutés. Il peut également s'agir d'une étape de déploiement, où le code est déployé dans des environnements d'exécution. Chaque étape est constituée d'une série d'actions en série ou en parallèle.

Transitions

Une transition est le point où l'exécution d'un pipeline passe à l'étape suivante du pipeline. Vous pouvez désactiver la transition en entrée d'une étape pour empêcher des exécutions d'entrer dans cette étape, puis activer la transition pour permettre la poursuite des exécutions. Lorsque plusieurs exécutions arrivent à une transition désactivée, seule l'exécution la plus ancienne passe à l'étape suivante lorsque la transition est activée. Cela signifie que les exécutions plus récentes continuent de remplacer les exécutions en attente pendant que la transition est désactivée. Une fois la transition activée, l'exécution qui continue est l'exécution de remplacement.



Actions

Une action est un ensemble d'opérations effectuées sur le code d'application et configurées de manière à ce que les actions s'exécutent dans le pipeline à un point spécifié. Cela peut inclure des éléments tels qu'une action source à partir d'une modification de code, une action pour déployer l'application sur des instances, etc. Par exemple, une étape de déploiement peut contenir une action de déploiement qui déploie du code vers un service de calcul tel qu'Amazon EC2 ou. AWS Lambda

Les types CodePipeline d'action valides sont `source` `buildtest`,`deploy`,`approval`, et`invoke`. Pour voir la liste des fournisseurs d'actions, consultez Fournisseurs d'actions valides dans CodePipeline.

Les actions peuvent être exécutées en série ou en parallèle. Pour plus d'informations sur les actions en série et en parallèle au cours d'une étape, consultez les `runOrder` informations relatives aux exigences relatives à la structure des actions.

Exécutions de pipeline

Une exécution est un ensemble de modifications publiées par un pipeline. Chaque exécution de pipeline est unique et possède son propre ID. Une exécution correspond à un ensemble de modifications, telles qu'une validation fusionnée ou une version manuelle de la dernière validation. Deux exécutions peuvent publier le même ensemble de modifications à des moments différents.

Un pipeline peut traiter plusieurs exécutions en même temps, alors qu'une étape de pipeline ne traite qu'une seule exécution à la fois. Pour ce faire, une étape est verrouillée pendant qu'elle traite une exécution. Deux exécutions de pipeline ne peuvent pas occuper la même étape en même temps. L'exécution en attente d'entrer dans la phase occupée est une exécution entrante. Une exécution entrante peut toujours échouer, être remplacée ou arrêtée manuellement. Pour plus d'informations sur le fonctionnement des exécutions entrantes, consultezComment fonctionnent les exécutions entrantes.

Les exécutions de pipeline parcourent les étapes du pipeline dans l'ordre. Les statuts valides pour les pipelines sont `InProgress`, `Stopping`, `Stopped`, `Succeeded`, `Superseded` et `Failed`.

Pour de plus amples informations, veuillez consulter PipelineExecution.

Exécutions arrêtées

L'exécution du pipeline peut être arrêtée manuellement afin que l'exécution du pipeline en cours ne se poursuive pas à travers le pipeline. Si elle est arrêtée manuellement, une exécution de pipeline affiche un statut `Stopping` jusqu'à ce qu'elle soit complètement arrêtée. Ensuite, elle affiche un statut `Stopped`. Un pipeline au statut `Stopped` peut faire l'objet d'une nouvelle tentative d'exécution.

Il existe deux manières d'arrêter l'exécution d'un pipeline :

- Arrêter et attendre
- Arrêter et abandonner

Pour plus d'informations sur les cas d'utilisation pour arrêter une exécution et les détails de séquence pour ces options, veuillez consulter Arrêt des exécutions de pipeline.

Exécutions ayant échoué

Si une exécution échoue, elle s'arrête et ne parcourt pas complètement le pipeline. Son statut passe à la scène `FAILED` et la CodePipeline déverrouille. Une exécution plus récente peut récupérer et entrer dans l'étape déverrouillée, puis la verrouiller. Vous pouvez faire une nouvelle tentative pour

l'exécution ayant échoué, sauf si celle-ci a été remplacée ou ne peut pas faire l'objet d'une nouvelle tentative. Vous pouvez revenir à une précédente exécution réussie d'une étape ayant échoué.

Modes d'exécution

Pour fournir l'ensemble de modifications le plus récent via un pipeline, les exécutions plus récentes passent et remplacent les exécutions moins récentes déjà exécutées via le pipeline. Lorsque cela se produit, l'exécution la moins récente est remplacée par la plus récente. Une exécution peut être remplacée par une exécution plus récente à un certain point. Ce point se situe entre les étapes. Le mode d'exécution par défaut est SUPERSEDED.

En mode SUPERSEDED, si une exécution attend d'entrer dans une phase verrouillée, une exécution plus récente peut la rattraper et la remplacer. L'exécution la plus récente attend alors que l'étape se déverrouille, et l'exécution remplacée s'arrête avec un statut `SUPERSEDED`. Lorsqu'une exécution de pipeline est remplacée, l'exécution est arrêtée et ne parcourt pas complètement le pipeline. À ce stade, vous ne pouvez plus relancer l'exécution remplacée. Les autres modes d'exécution disponibles sont le mode PARALLÈLE ou le mode QUEUED.

Pour plus d'informations sur les modes d'exécution et les étapes verrouillées, consultezComment les exécutions sont traitées en mode SUPERSEDED.

Opérations scéniques

Lorsqu'une exécution de pipeline passe par une étape, celle-ci est en train de terminer toutes les actions qu'elle contient. Pour plus d'informations sur le fonctionnement des étapes et des informations sur les étapes verrouillées, consultezComment les exécutions sont traitées en mode SUPERSEDED.

Les statuts valides pour les étapes sont `InProgress`Stopping, `Stopped`Succeeded, et`Failed`. Vous pouvez réessayer une étape qui a échoué, sauf si la phase qui a échoué ne peut pas être réessayée. Pour de plus amples informations, veuillez consulter StageExecution. Vous pouvez revenir en arrière d'une étape à une exécution réussie précédente spécifiée. Une étape peut être configurée pour revenir automatiquement en arrière en cas de panne, comme indiqué dansConfiguration de la restauration des étapes. Pour de plus amples informations, veuillez consulter RollbackStage.

Exécutions d'actions

Une exécution d'action est le processus d'exécution d'une action configurée qui fonctionne sur des artefacts désignés. Il peut s'agir d'artefacts en entrée, d'artefacts en sortie ou des deux. Par exemple,

une action de build peut exécuter des commandes de build sur un artefact en entrée, comme la compilation du code source de l'application. Les détails d'une exécution d'action incluent un ID d'exécution d'action, le déclencheur source d'exécution du pipeline associé et les artefacts d'entrée et de sortie de l'action.

Les statuts valides pour les actions sont`InProgress`, `Abandoned``Succeeded`, ou`Failed`. Pour de plus amples informations, veuillez consulter ActionExecution.

Types d'exécution

L'exécution d'un pipeline ou d'une étape peut être une exécution standard ou une exécution annulée.

Pour les types standard, l'exécution possède un identifiant unique et constitue une exécution complète du pipeline. Un rollback de pipeline comporte une étape à annuler et une exécution réussie pour cette étape en tant qu'exécution cible vers laquelle revenir en arrière. L'exécution du pipeline cible est utilisée pour récupérer les révisions des sources et les variables pour la phase à réexécuter.

Types d'action

Les types d'actions sont des actions préconfigurées qui peuvent être sélectionnées dans CodePipeline. Le type d'action est défini par son propriétaire, son fournisseur, sa version et sa catégorie. Le type d'action fournit des paramètres personnalisés qui sont utilisés pour effectuer les tâches d'action dans un pipeline.

Pour plus d'informations sur les Services AWS produits et services tiers que vous pouvez intégrer à votre pipeline en fonction du type d'action, consultezIntégrations avec les types CodePipeline d'action.

Pour plus d'informations sur les modèles d'intégration pris en charge pour les types d'action dans CodePipeline, consultezRéférence du modèle d'intégration.

Pour plus d'informations sur la manière dont les fournisseurs tiers peuvent configurer et gérer les types d'actions dans CodePipeline, consultezUtilisation des types d'actions.

Artefacts

Les artefacts font référence à l'ensemble des données (code source de l'application, applications créées, dépendances, fichiers de définitions, modèles, etc.) utilisées par les actions de pipeline. Les artefacts sont produits par certaines actions et consommés par d'autres. Dans un pipeline, les artefacts peuvent être l'ensemble de fichiers utilisés par une action (artefacts d'entrée) ou la sortie mise à jour d'une action terminée (artefacts de sortie).

Les actions transmettent la sortie à une autre action pour un traitement ultérieur à l'aide du compartiment d'artefacts du pipeline. CodePipeline copie les artefacts dans la boutique d'artefacts, où l'action les récupère. Pour plus d'informations sur les artefacts, consultez Artefacts d'entrée et de sortie.

Révisions source

Lorsque vous modifiez le code source, une nouvelle version est créée. Une révision source est la version d'une modification de la source qui déclenche l'exécution d'un pipeline. Une exécution traite les révisions des sources. Pour les CodeCommit dépôts GitHub et les dépôts, il s'agit du commit. Pour les actions ou les compartiments S3, il s'agit de la version de l'objet.

Vous pouvez démarrer l'exécution d'un pipeline avec une révision de source, telle qu'un commit, que vous spécifiez. L'exécution traitera la révision spécifiée et remplacera la révision qui aurait été utilisée pour l'exécution. Pour de plus amples informations, veuillez consulter Démarrer un pipeline avec un remplacement de révision de la source.

Triggers

Les déclencheurs sont des événements qui démarrent votre pipeline. Certains déclencheurs, tels que le démarrage manuel d'un pipeline, sont disponibles pour tous les fournisseurs d'actions source d'un pipeline. Certains déclencheurs dépendent du fournisseur source d'un pipeline. Par exemple, les CloudWatch événements doivent être configurés avec des ressources événementielles d'Amazon CloudWatch dont l'ARN du pipeline a été ajouté en tant que cible dans la règle d'événement. Amazon CloudWatch Events est le déclencheur recommandé pour la détection automatique des modifications pour les pipelines avec une action source CodeCommit ou S3. Les webhooks sont un type de déclencheur configuré pour les événements du référentiel tiers. Par exemple, WebHookV2 est un type de déclencheur qui permet d'utiliser des balises Git pour démarrer des pipelines avec des fournisseurs de sources tiers tels que GitHub.com GitHub Enterprise Server GitLab.com, GitLab self-managed ou Bitbucket Cloud. Dans la configuration du pipeline, vous pouvez spécifier un filtre pour les déclencheurs, tels que les demandes push ou pull. Vous pouvez filtrer les événements de code push sur les balises Git, les branches ou les chemins de fichiers. Vous pouvez filtrer les événements de demande d'extraction en fonction de l'événement (ouvert, mis à jour, fermé), des branches ou des chemins de fichiers.

Pour plus d'informations sur les déclencheurs, consultez Démarrez un pipeline dans CodePipeline. Pour un didacticiel qui explique comment utiliser les balises Git comme déclencheurs pour votre pipeline, consultezTutoriel : Utiliser les balises Git pour démarrer votre pipeline.

> **Important**
>
> Les sondages seront désactivés pour les pipelines qui sont inactifs pendant plus de 30 jours. Pour plus d'informations, consultez la section « sondage » DisabledAt dans la référence sur la structure du pipeline. Pour connaître les étapes à suivre pour migrer votre pipeline de la détection des modifications par sondage vers la détection des modifications basée sur les événements, consultez la section Méthodes de détection des modifications.

Variables

Une variable est une valeur qui peut être utilisée pour configurer dynamiquement des actions dans votre pipeline. Les variables peuvent être déclarées au niveau du pipeline ou émises par des actions dans le pipeline. Les valeurs des variables sont résolues au moment de l'exécution du pipeline et peuvent être consultées dans l'historique des exécutions. Pour les variables déclarées au niveau du pipeline, vous pouvez soit définir des valeurs par défaut dans la configuration du pipeline, soit les remplacer pour une exécution donnée. Pour les variables émises par une action, la valeur est disponible une fois l'action exécutée avec succès. Pour de plus amples informations, veuillez consulter Référence des variables.

Conditions

Une condition contient un ensemble de règles qui sont évaluées. Si toutes les règles d'une condition réussissent, la condition est remplie. Vous pouvez configurer les conditions de sorte que lorsque les critères ne sont pas remplis, le résultat spécifié, tel que l'échec de l'étape, soit activé. Les conditions sont également appelées portes car elles vous permettent de spécifier quand une exécution entrera et traversera une étape ou quittera la phase après l'avoir parcourue. Cela revient à autoriser une ligne de circulation sur une route à se rassembler à une barrière fermée, puis à spécifier l'ouverture de la barrière pour permettre la circulation dans une zone. Les résultats pour les types de conditions incluent l'échec de l'étape ou l'annulation de la phase. Les conditions vous aident à spécifier à quel moment ces actions se produisent au cours d'une étape du pipeline. Vous pouvez modifier les conditions lors de l'exécution.

Il existe trois types de conditions. Les conditions d'entrée répondent à la question « Si les règles de la condition sont remplies, passez à l'étape ». L'étape est verrouillée lorsque l'exécution entre dans la phase, puis les règles sont exécutées. Pour les conditions On Failure, la règle s'active lorsqu'une étape échoue, ce qui entraîne l'annulation d'une étape échouée. Pour les conditions On Success,

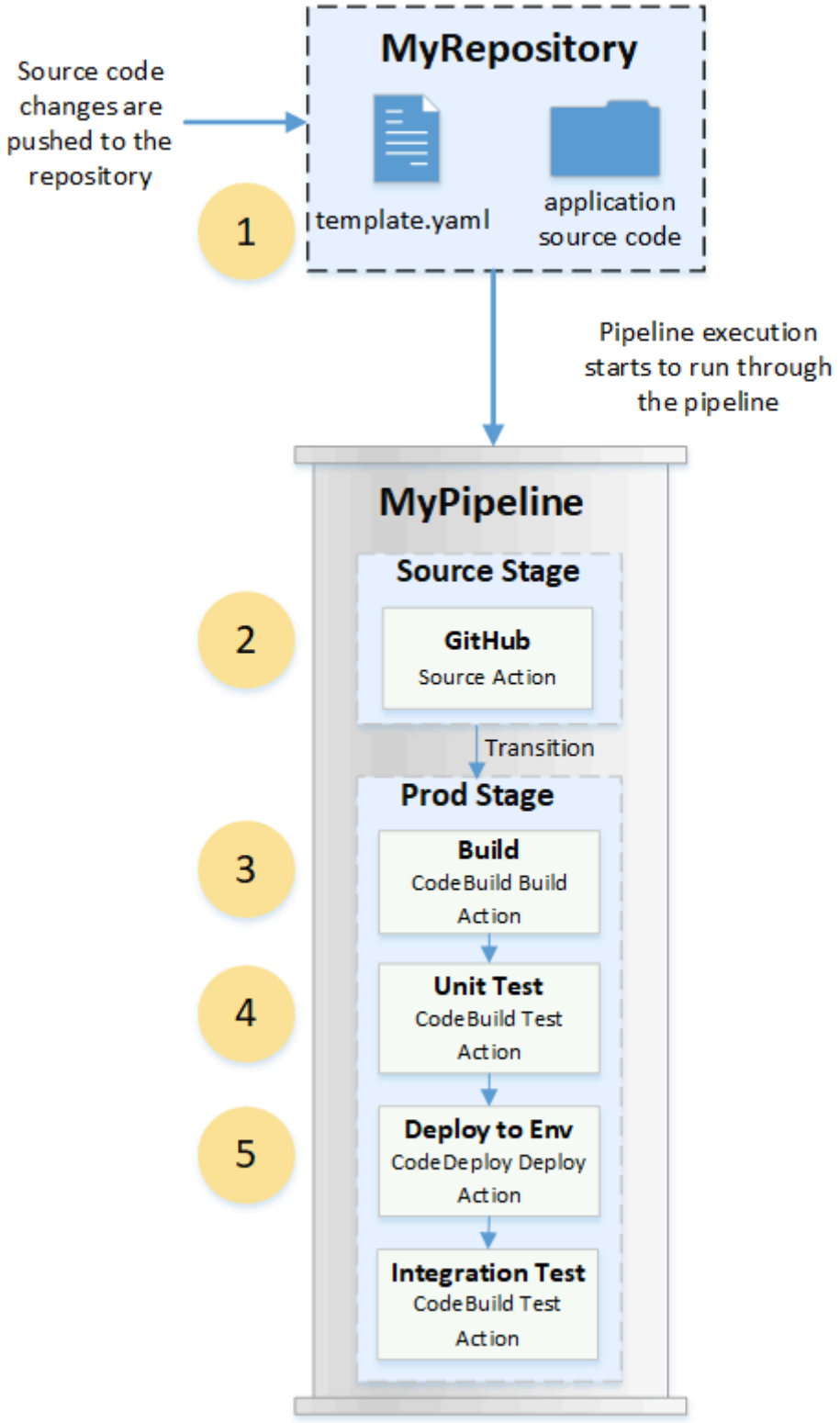
la règle s'active lorsqu'une étape est réussie, par exemple en vérifiant la réussite d'une exécution pour détecter les alarmes avant de continuer. Par exemple, une condition En cas de réussite entraîne l'annulation d'une étape réussie si la CloudWatchAlarm règle détecte la présence d'alarmes dans l'environnement de déploiement. Pour de plus amples informations, veuillez consulter Comment fonctionnent les conditions scéniques ?.

Rules

Les conditions utilisent une ou plusieurs règles préconfigurées qui s'exécutent et effectuent des vérifications qui activeront ensuite le résultat configuré lorsque la condition n'est pas remplie. Par exemple, si vous respectez toutes les règles relatives à une règle de condition d'entrée qui vérifie l'état de l'alarme et les délais de déploiement, une étape sera déployée avec succès une fois toutes les vérifications réussies. Pour de plus amples informations, veuillez consulter Comment fonctionnent les conditions scéniques ?.

DevOps exemple de pipeline

À titre d'exemple de DevOps pipeline, un pipeline à deux étages peut avoir un étage source appelé Source et un deuxième étage appelé Prod. Dans cet exemple, le pipeline met à jour l'application avec les dernières modifications et déploie en continu le résultat le plus récent. Avant de déployer l'application la plus récente, le pipeline génère et teste l'application Web. Dans cet exemple, un groupe de développeurs a configuré un modèle d'infrastructure et le code source d'une application Web dans un GitHub référentiel appelé MyRepository.



Par exemple, un développeur transmet un correctif à la page d'index de l'application Web, ce qui entraîne l'action suivante :

1. Le code source de l'application est conservé dans un référentiel configuré en tant qu'action GitHub source dans le pipeline. Lorsque les développeurs envoient des validations au référentiel, CodePipeline détectent la modification poussée et que l'exécution d'un pipeline démarre à partir de la phase source.
2. L'action GitHub source s'exécute correctement (c'est-à-dire que les dernières modifications ont été téléchargées et stockées dans le compartiment d'artefacts propre à cette exécution). Les artefacts de sortie produits par l'action GitHub source, qui sont les fichiers d'application du référentiel, sont ensuite utilisés comme artefacts d'entrée sur lesquels les actions devront travailler à l'étape suivante.
3. L'exécution du pipeline passe de l'étape source à l'étape de production. La première action de la phase de production exécute un projet de génération créé CodeBuild et configuré en tant qu'action de génération dans le pipeline. La tâche de build extrait une image d'environnement de build et génère l'application Web dans un conteneur virtuel.
4. L'action suivante de la phase de production est un projet de test unitaire créé CodeBuild et configuré en tant qu'action de test dans le pipeline.
5. Le code testé est ensuite travaillé par une action de déploiement dans l'étape de production qui déploie l'application dans un environnement de production. Une fois l'action de déploiement terminée avec succès, l'action finale de l'étape est un projet de test d'intégration créé CodeBuild et configuré en tant qu'action de test dans le pipeline. L'action de test appelle les scripts shell qui installent et exécutent un outil de test, tel qu'un vérificateur de liens, sur l'application Web. Une fois ces étapes terminées, la sortie est une application Web générée et un ensemble de résultats de test.

Les développeurs peuvent ajouter au pipeline des actions qui déploient ou testent davantage l'application après sa génération et son test pour chaque modification.

Pour plus d'informations, consultez Fonctionnement des exécutions de pipeline.

Fonctionnement des exécutions de pipeline

Cette section fournit une vue d'ensemble de la manière dont CodePipeline traite un ensemble de modifications. CodePipelinesuit chaque exécution de pipeline qui démarre lorsqu'un pipeline est démarré manuellement ou qu'une modification est apportée au code source. CodePipeline utilise les modes d'exécution suivants pour gérer la progression de chaque exécution dans le pipeline. Pour plus d'informations, consultez Définir ou modifier le mode d'exécution du pipeline.

- Mode REMPLACÉ : une exécution plus récente peut remplacer une exécution plus ancienne. Il s'agit de l'option par défaut.
- Mode FILE D'ATTENTE : les exécutions sont traitées une par une dans l'ordre dans lequel elles sont mises en file d'attente. Cela nécessite le type de pipeline V2.
- Mode PARALLÈLE : En mode PARALLÈLE, les exécutions s'exécutent simultanément et indépendamment les unes des autres. Les exécutions n'attendent pas la fin des autres exécutions pour démarrer ou terminer. Cela nécessite le type de pipeline V2.

> **Important**
>
> Pour les pipelines en mode PARALLÈLE, la restauration des étapes n'est pas disponible. De même, les conditions de défaillance avec un type de résultat d'annulation ne peuvent pas être ajoutées à un pipeline en mode PARALLÈLE.

Démarrage des exécutions de pipeline

Vous pouvez démarrer une exécution lorsque vous modifiez votre code source ou démarrer manuellement le pipeline. Vous pouvez également déclencher une exécution via une règle Amazon CloudWatch Events que vous planifiez. Par exemple, lorsqu'une modification de code source est transmise à un référentiel configuré en tant qu'action source du pipeline, le pipeline détecte la modification et lance une exécution.

> **Note**
>
> Si un pipeline contient plusieurs actions source, elles sont toutes réexécutées même si une modification est détectée pour une seule action source.

Comment les révisions de source sont traitées dans les exécutions de pipelines

Pour chaque exécution de pipeline qui commence par des modifications du code source (révisions de source), les révisions de source sont déterminées comme suit.

- Pour les pipelines ayant une CodeCommit source, le HEAD est cloné CodePipeline au moment où le commit est poussé. Par exemple, un commit est poussé, ce qui démarre le pipeline pour

l'exécution 1. Au moment où un deuxième commit est poussé, cela démarre le pipeline pour l'exécution 2.

> **Note**
>
> Pour les pipelines en mode PARALLÈLE avec une CodeCommit source, quel que soit le commit qui a déclenché l'exécution du pipeline, l'action source clonera toujours le HEAD au moment de son démarrage. Pour plus d'informations, consultez CodeCommit ou les révisions de source S3 en mode PARALLÈLE peuvent ne pas correspondre à EventBridge l'événement.

- Pour les pipelines dotés d'une source S3, l' EventBridge événement de mise à jour du compartiment S3 est utilisé. Par exemple, l'événement est généré lorsqu'un fichier est mis à jour dans le bucket source, ce qui démarre le pipeline pour l'exécution 1. Au moment où l'événement pour une deuxième mise à jour du bucket est créé, le pipeline pour l'exécution 2 démarre.

  > **Note**
  >
  > Pour les pipelines en mode PARALLÈLE avec une source S3, quelle que soit la balise d'image qui a déclenché l'exécution, l'action source commencera toujours par la dernière balise d'image. Pour plus d'informations, consultez CodeCommit ou les révisions de source S3 en mode PARALLÈLE peuvent ne pas correspondre à EventBridge l'événement.

- Pour les pipelines ayant une source de connexions, comme vers Bitbucket, le HEAD est cloné CodePipeline au moment où le commit est poussé. Par exemple, pour un pipeline en mode PARALLÈLE, un commit est poussé, ce qui démarre le pipeline pour l'exécution 1, et la seconde exécution du pipeline utilise le second commit.

Fonctionnement des remplacements de source avec le transformateur EventBridge d'entrée

Vous pouvez utiliser des remplacements pour démarrer un pipeline avec un ID de révision source spécifique que vous fournissez pour l'exécution du pipeline. Par exemple, si vous souhaitez démarrer un pipeline qui traitera un ID de validation spécifique à partir de votre CodeCommit source, vous pouvez ajouter l'ID de validation en tant que remplacement lorsque vous démarrez votre pipeline.

Il existe quatre types de révision des sources pour `revisionType` :

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

> **Note**
>
> Pour les révisions de source `COMMIT_ID` et les `IMAGE_DIGEST` types de révisions, l'ID de révision source s'applique à tout le contenu du référentiel, dans toutes les branches.

> **Note**
>
> Pour les `S3_OBJECT_VERSION_ID` et `S3_OBJECT_KEY` types de révisions de source, l'un ou l'autre des types peut être utilisé indépendamment, ou ils peuvent être utilisés ensemble pour remplacer la source par un identifiant de version spécifique ObjectKey . Pour`S3_OBJECT_KEY`, le paramètre de configuration `AllowOverrideForS3ObjectKey` doit être défini sur`true`. Pour plus d'informations sur les paramètres de configuration des sources S3, consultezParamètres de configuration.

Vous pouvez spécifier des remplacements de source à l'aide du transformateur d'entrée. EventBridge Utilisez le transformateur d'entrée pour transmettre les données de l'une des manières suivantes :

- Vous pouvez utiliser le transformateur d'entrée pour transmettre les données sous forme de paramètres JSON.
- Vous pouvez utiliser le transformateur d'entrée pour transmettre les variables du pipeline.

Pour des exemples de transmission de données sous forme de paramètres JSONActions et ressources relatives aux sources Amazon ECR EventBridge, consultez, Connexion à la source Amazon S3 : actions qui utilisent EventBridge et AWS CloudTrail pour S3 et CodeCommit actions à la source et EventBridge pour CodeCommit.

Arrêt des exécutions de pipeline

Pour utiliser la console pour arrêter l'exécution d'un pipeline, vous pouvez choisir Stop execution (Arrêter l'exécution) sur la page de visualisation du pipeline, sur la page de l'historique de l'exécution ou sur la page de l'historique détaillé. Pour utiliser l'interface de ligne de commande pour arrêter l'exécution d'un pipeline, utilisez la commande `stop-pipeline-execution`. Pour plus d'informations, consultez Arrêtez l'exécution d'un pipeline dans CodePipeline.

Il existe deux manières d'arrêter l'exécution d'un pipeline :

- Arrêter et attendre : toutes les exécutions d'action en cours sont autorisées à se terminer et les actions suivantes ne sont pas démarrées. Les étapes suivantes du pipeline ne sont pas exécutées. Vous ne pouvez pas utiliser cette option sur une exécution déjà à l'état `Stopping`.
- Arrêter et abandonner : toutes les exécutions d'action en cours sont abandonnées et ne sont pas terminées, et les actions suivantes ne sont pas lancées. Les étapes suivantes du pipeline ne sont pas exécutées. Vous pouvez utiliser cette option sur une exécution déjà à l'état `Stopping`.

> **Note**
>
> Cette option peut entraîner des tâches défaillantes ou hors séquence.

Chaque option aboutit à une séquence différente de phases d'exécution de pipeline et d'action, comme suit.

Option 1 : Arrêter et attendre

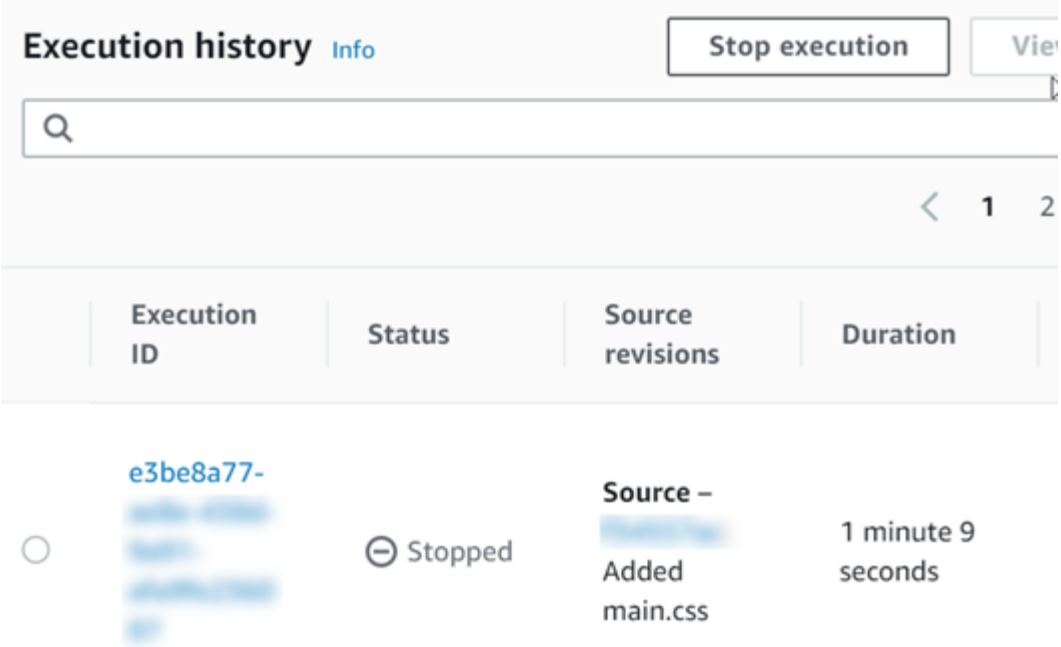
Lorsque vous choisissez d'arrêter et d'attendre, l'exécution sélectionnée se poursuit jusqu'à ce que les actions en cours soient terminées. Par exemple, l'exécution du pipeline suivant a été arrêtée pendant que l'action de création était en cours.

1. Dans la vue du pipeline, la bannière de message de réussite s'affiche et l'action de création se poursuit jusqu'à ce qu'elle soit terminée. L'état d'exécution du pipeline est Stopping.

   Dans la vue de l'historique, l'état des actions en cours, telles que l'action de création, est In progress jusqu'à ce que l'action de création soit terminée. Pendant que les actions sont en cours, l'état de l'exécution du pipeline est Stopping.

2. L'exécution s'arrête lorsque le processus d'arrêt est terminé. Si l'action de création se termine avec succès, son état passe à Succeeded et l'exécution du pipeline affiche l'état Stopped. Les actions suivantes ne démarrent pas. Le bouton Retry (Réessayer) est activé.

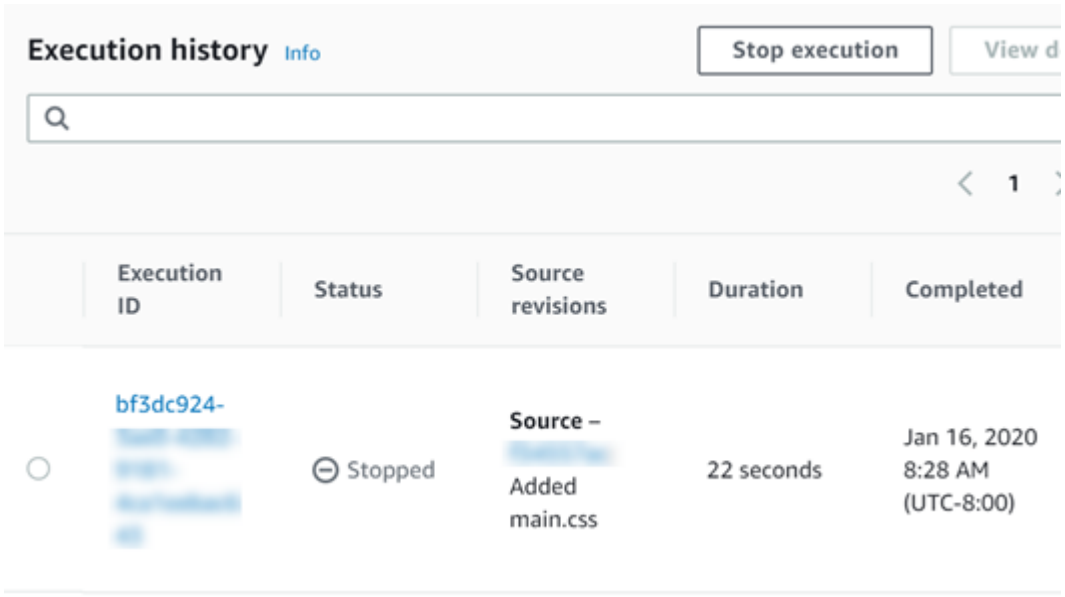
   Dans la vue de l'historique, l'état de l'exécution est Stopped une fois que l'action en cours est terminée.



Option 2 : Arrêter et abandonner

Lorsque vous choisissez d'arrêter et d'abandonner, l'exécution sélectionnée n'attend pas la fin des actions en cours. Les actions sont abandonnées. Par exemple, l'exécution du pipeline suivant a été arrêtée et abandonnée pendant que l'action de création était en cours.

1. Dans la vue du pipeline, le message de bannière de réussite s'affiche, l'action de création affiche l'état In progress (En cours) et l'exécution du pipeline affiche l'état Stopping (En cours d'arrêt).
2. Après l'arrêt de l'exécution du pipeline, l'action de création affiche l'état Abandoned et l'exécution du pipeline affiche l'état Stopped. Les actions suivantes ne démarrent pas. Le bouton Retry (Réessayer) est activé.
3. Dans la vue de l'historique, l'état d'exécution est Stopped.



Cas d'utilisation pour arrêter l'exécution d'un pipeline

Nous vous recommandons d'utiliser l'option d'arrêt et d'attente pour arrêter l'exécution d'un pipeline. Cette option est plus sûre car elle évite d'éventuelles tâches défaillantes ou hors séquence dans votre pipeline. Lorsqu'une action est abandonnée dans CodePipeline, le fournisseur de l'action poursuit toutes les tâches liées à l'action. Dans le cas d'une CloudFormation action, l'action de déploiement dans le pipeline est abandonnée, mais la mise à jour de la pile peut se poursuivre et entraîner un échec de mise à jour.

À titre d'exemple d'actions abandonnées qui peuvent entraîner des tâches hors séquence, si vous déployez un fichier volumineux (1 Go) par le biais d'une action de déploiement S3 et que vous choisissez d'arrêter et d'abandonner l'action alors que le déploiement est déjà en cours, l'action est abandonnée dans Amazon S3 CodePipeline, mais se poursuit dans Amazon S3. Amazon S3 ne reçoit aucune instruction pour annuler le téléchargement. Ensuite, si vous démarrez une nouvelle exécution de pipeline avec un très petit fichier, deux déploiements seront en cours. Étant donné la petite taille du fichier de la nouvelle exécution, le nouveau déploiement se termine alors que l'ancien déploiement est toujours en cours de téléchargement. Lorsque l'ancien déploiement se termine, le nouveau fichier est remplacé par l'ancien fichier.

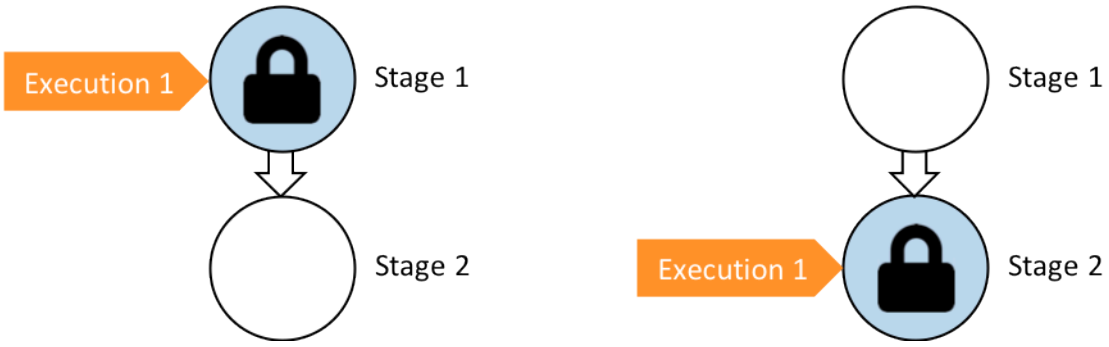
Vous pouvez utiliser l'option d'arrêt et d'abandon dans le cas d'une action personnalisée. Par exemple, vous pouvez abandonner une action personnalisée dont le travail n'a pas besoin d'être terminé avant de commencer une nouvelle exécution pour corriger un bogue.

Comment les exécutions sont traitées en mode SUPERSEDED

Le mode par défaut pour le traitement des exécutions est le mode SUPERSEDED. Une exécution consiste en un ensemble de modifications collectées et traitées par l'exécution. Les pipelines peuvent traiter plusieurs exécutions en même temps. Chaque exécution est exécutée à travers le pipeline séparément. Le pipeline traite chaque exécution dans l'ordre et peut remplacer une exécution antérieure par une exécution plus récente. Les règles suivantes sont utilisées pour traiter les exécutions dans un pipeline pour le mode SUPERSEDED.

Règle 1 : les phases sont verrouillées lorsqu'une exécution est en cours de traitement

Une étape ne pouvant traiter qu'une seule exécution à la fois, elle est verrouillée lorsqu'elle est en cours. Lorsque l'exécution termine une étape, elle passe à l'étape suivante du pipeline.

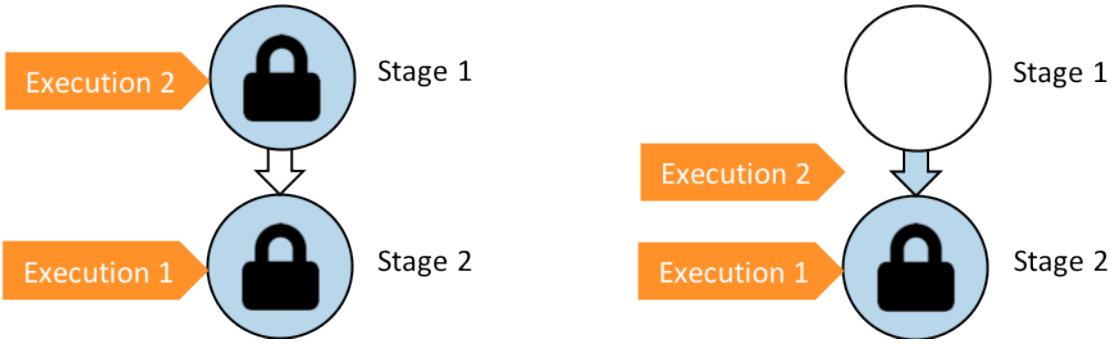


Avant : Stage 1 is locked as Execution 1 enters. Après : Stage 2 is locked as Execution 1 enters.

Règle 2 : Les exécutions suivantes attendent le déverrouillage de l'étape

Lorsqu'une étape est verrouillée, les exécutions en attente sont conservées devant l'étape verrouillée. Toutes les actions configurées pour une étape doivent réussir avant que l'étape ne soit considérée comme étant achevée. Une défaillance libère le verrou sur l'étape. Lorsqu'une exécution est arrêtée, l'exécution ne se poursuit pas et la phase est déverrouillée.

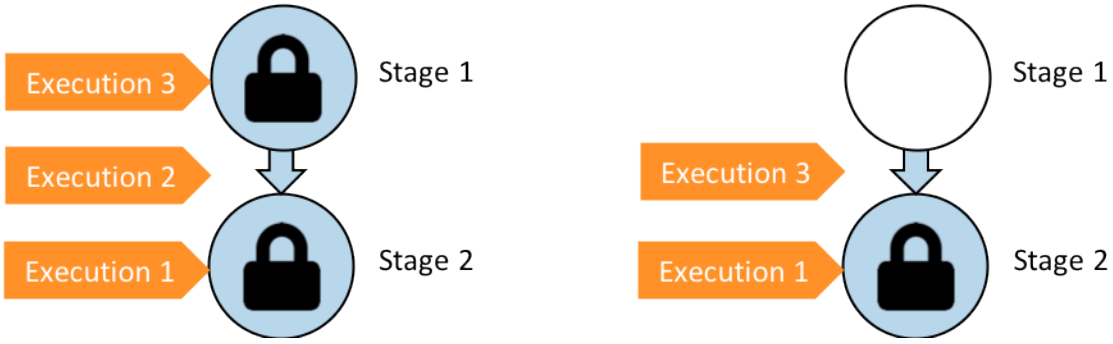
> **Note**
>
> Avant d'arrêter une exécution, nous vous recommandons de désactiver la transition au début de la phase. De cette façon, lorsque la phase est déverrouillée en raison de l'arrêt de l'exécution, la phase n'accepte aucune exécution de pipeline ultérieure.



Avant : Stage 2 is locked as Execution 1 enters. Après : Execution 2 exits Stage 1 and waits between stages.

Règle 3 : Les exécutions en attente sont remplacées par des exécutions plus récentes

Les exécutions ne sont remplacées qu'entre les étapes. Une étape verrouillée conserve une exécution à l'avant de l'étape en attendant la fin de l'étape. Une exécution plus récente dépasse une exécution en attente et passe à l'étape suivante dès que l'étape est déverrouillée. L'exécution remplacée ne se poursuit pas. Dans cet exemple, l'exécution 2 a été remplacée par l'exécution 3 pendant l'attente de l'étape verrouillée. L'exécution 3 passe à l'étape suivante.



Avant : l'exécution 2 attend entre les étapes tandis que l'exécution 3 entre dans l'étape 1. Après : l'exécution 3 quitte l'étape 1 et l'exécution 2 est remplacée par l'exécution 3.

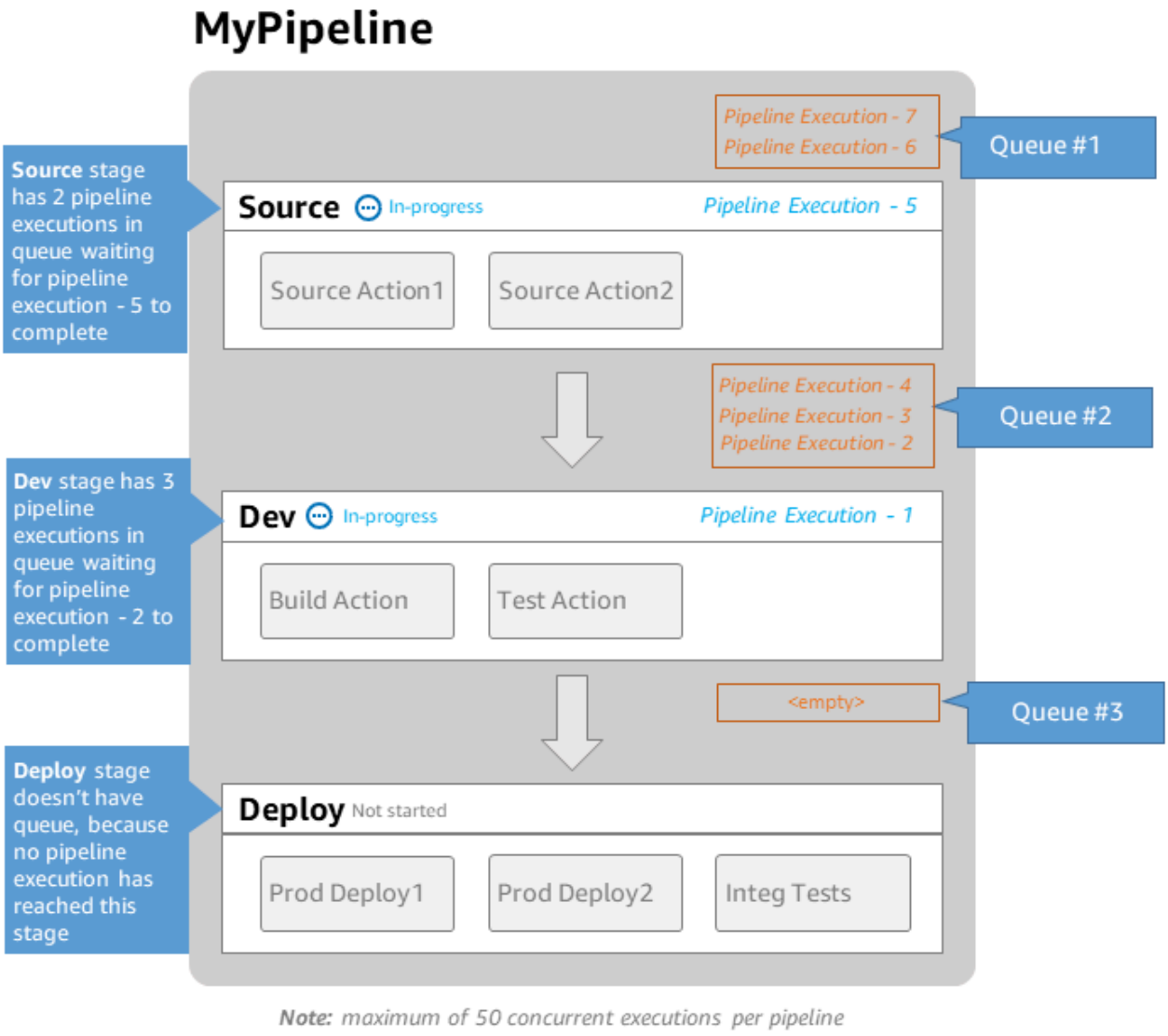
Pour plus d'informations sur les considérations relatives à l'affichage et à la commutation entre les modes d'exécution, consultezDéfinir ou modifier le mode d'exécution du pipeline. Pour plus d'informations sur les quotas avec modes d'exécution, consultezQuotas dans AWS CodePipeline.

Comment les exécutions sont traitées en mode QUEUED

Pour les pipelines en mode QUEUED, les étapes sont verrouillées lorsqu'une exécution est en cours de traitement ; toutefois, les exécutions en attente ne dépassent pas les exécutions qui ont déjà commencé.

Les exécutions en attente se rassemblent aux points d'entrée des étapes verrouillées dans l'ordre dans lequel elles atteignent l'étape, formant une file d'exécutions en attente. Avec le mode QUEUED, vous pouvez avoir plusieurs files d'attente dans le même pipeline. Lorsqu'une exécution en file d'attente entre dans une phase, celle-ci est verrouillée et aucune autre exécution ne peut entrer. Ce comportement reste le même que celui du mode SUPERSEDED. Lorsque l'exécution termine l'étape, celle-ci est déverrouillée et prête pour la prochaine exécution.

Le schéma suivant montre comment les étapes d'un pipeline en mode QUEUED sont exécutées. Par exemple, alors que le stage Source traite l'exécution 5, les exécutions des 6 et 7 forment la file d'attente #1 et attendent au point d'entrée de l'étape. La prochaine exécution dans la file d'attente sera traitée après le déverrouillage de l'étape.



Pour plus d'informations sur les considérations relatives à l'affichage et à la commutation entre les modes d'exécution, consultezDéfinir ou modifier le mode d'exécution du pipeline. Pour plus d'informations sur les quotas avec modes d'exécution, consultezQuotas dans AWS CodePipeline.

Comment les exécutions sont traitées en mode PARALLÈLE

Pour les pipelines en mode PARALLÈLE, les exécutions sont indépendantes les unes des autres et n'attendez pas que les autres exécutions soient terminées pour démarrer. Il n'y a pas de files d'attente. Pour visualiser les exécutions parallèles dans le pipeline, utilisez la vue historique des exécutions.

Utilisez le mode PARALLÈLE dans les environnements de développement où chaque fonctionnalité possède sa propre branche de fonctionnalités et se déploie sur des cibles qui ne sont pas partagées par d'autres utilisateurs.

Pour plus d'informations sur les considérations relatives à l'affichage et à la commutation entre les modes d'exécution, consultezDéfinir ou modifier le mode d'exécution du pipeline. Pour plus d'informations sur les quotas avec modes d'exécution, consultezQuotas dans AWS CodePipeline.

Gestion du débit des canalisations

Le flux des exécutions de pipeline peut être contrôlé par :

- Une transition, qui contrôle le flux des exécutions dans l'étape. Les transitions peuvent être activées ou désactivées. Lorsqu'une transition est désactivée, les exécutions du pipeline ne peuvent pas entrer dans la phase. L'exécution du pipeline en attente d'entrer dans une phase où la transition est désactivée est appelée exécution entrante. Une fois que vous avez activé la transition, une exécution entrante passe à la phase et la verrouille.

  Comme pour les exécutions en attente d'une étape verrouillée, lorsqu'une transition est désactivée, l'exécution en attente d'entrer dans l'étape peut toujours être remplacée par une nouvelle exécution. Lorsqu'une transition désactivée est réactivée, l'exécution la plus récente, y compris toute exécution ayant remplacé des anciennes exécutions pendant que la transition était désactivée, entre dans l'étape.
- Action d'approbation qui empêche un pipeline de passer à l'action suivante tant que l'autorisation n'est pas accordée (par exemple, par le biais d'une approbation manuelle à partir d'une identité autorisée). Vous pouvez utiliser une action d'approbation pour contrôler l'heure à laquelle un pipeline passe à une étape de production finale, par exemple.

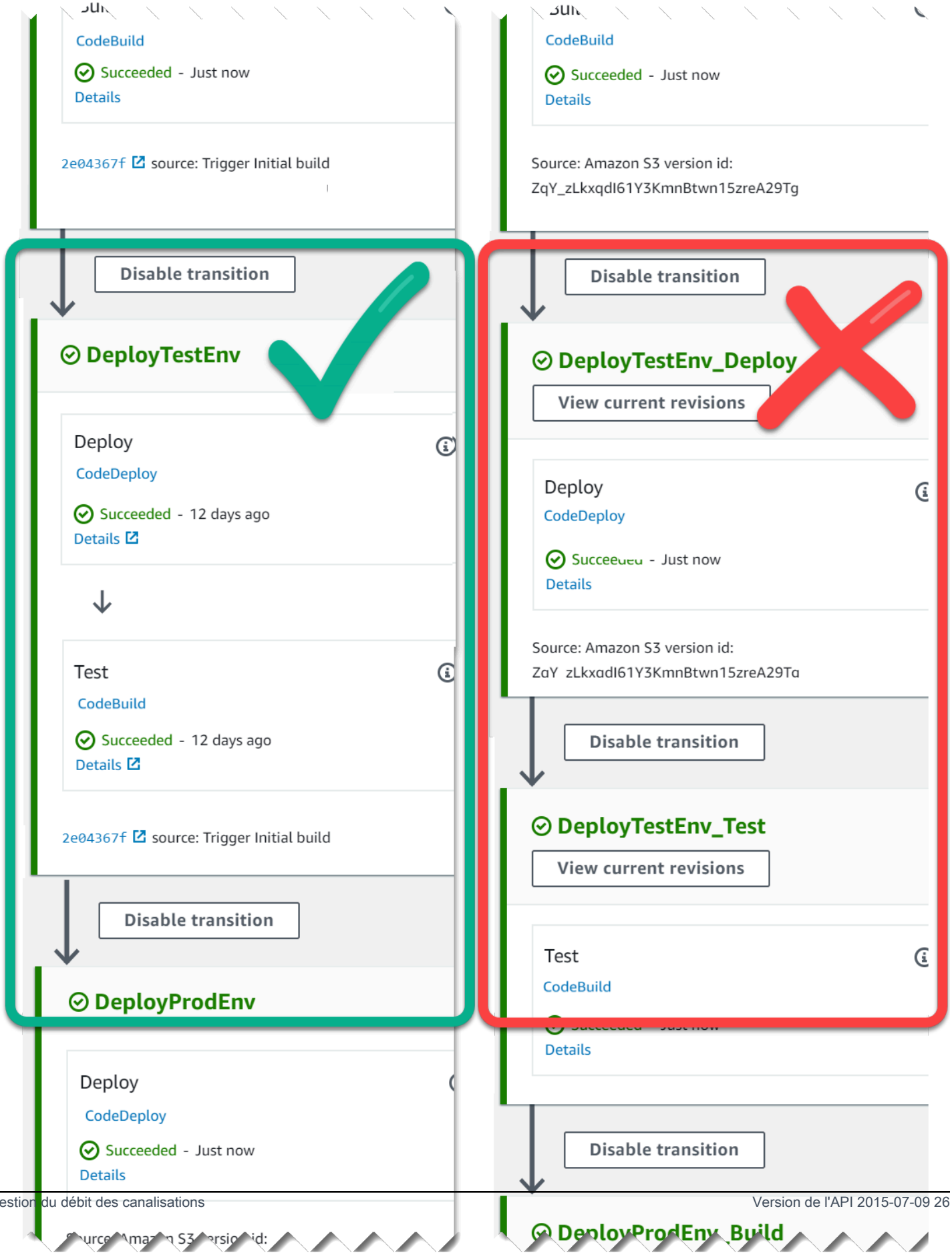
> **Note**
>
> Une étape avec une action d'approbation est verrouillée jusqu'à ce que l'action d'approbation soit approuvée ou rejetée, ou qu'elle ait expiré. Une action d'approbation ayant expiré est traitée de la même manière qu'une action ayant échoué.

- Un échec, lorsqu'une action d'une étape n'aboutit pas. La révision ne passe pas à l'action suivante de l'étape ou à la prochaine étape du pipeline. Les événements suivants peuvent se produire :
  - Vous relancez manuellement l'étape qui contient les actions ayant échoué. Cela reprend l'exécution (il réessaie les actions qui ont échoué et, en cas de succès, continue dans le stage/pipeline).
  - Une autre exécution entre dans l'étape ayant échoué et remplace l'exécution ayant échoué. À ce stade, l'exécution ayant échoué ne peut pas faire l'objet d'une nouvelle tentative.

Structure de pipeline recommandée

Lorsque vous décidez de la manière dont une modification de code doit circuler dans votre pipeline, il est préférable de regrouper les actions connexes au sein d'une étape de sorte que, lorsque l'étape se verrouille, les actions traitent toutes la même exécution. Vous pouvez créer un stage pour chaque environnement d'application Région AWS, chaque zone de disponibilité, etc. Un pipeline avec trop d'étapes (c'est-à-dire, trop granulaire) peut autoriser un nombre de modifications simultanées trop important, tandis qu'un pipeline avec de nombreuses actions dans une étape volumineuse (trop « grossière ») peut prendre trop de temps pour publier une modification.

Par exemple, une action de test après une action de déploiement dans la même phase permet de s'assure que le test portera sur la modification qui a été déployée. Dans cet exemple, une modification est déployée dans un environnement de test, puis testée ; ensuite, la dernière modification de l'environnement de test est déployée dans un environnement de production. Dans l'exemple recommandé, l'environnement de test et l'environnement de production constituent des étapes distinctes.


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

Gauche : Actions associées au test, au déploiement et à l'approbation regroupées (recommandé). Droite : Actions associées dans des étapes distinctes (non recommandé).

Comment fonctionnent les exécutions entrantes

Une exécution entrante est une exécution qui attend qu'une étape, une transition ou une action non disponible soit disponible avant de poursuivre. L'étape, la transition ou l'action suivante peut ne pas être disponible pour les raisons suivantes :

- Une autre exécution est déjà entrée dans la phase suivante et l'a bloquée.
- La transition pour passer à l'étape suivante est désactivée.

Vous pouvez désactiver une transition pour suspendre une exécution entrante si vous souhaitez contrôler si une exécution en cours a le temps de se terminer lors des étapes suivantes, ou si vous souhaitez arrêter toutes les actions à un moment donné. Pour déterminer s'il s'agit d'une exécution entrante, vous pouvez consulter le pipeline dans la console ou consulter le résultat de la get-pipeline-state commande.

Les exécutions entrantes prennent en compte les considérations suivantes :

- Dès que l'étape d'action, de transition ou de verrouillage devient disponible, l'exécution entrante en cours entre dans la phase et se poursuit dans le pipeline.
- Pendant que l'exécution entrante est en attente, elle peut être arrêtée manuellement. Une exécution entrante peut avoir un `Failed` état `InProgressStopped`, ou.
- Lorsqu'une exécution entrante a été arrêtée ou a échoué, elle ne peut pas être réessayée car aucune action n'a échoué à recommencer. Lorsqu'une exécution entrante a été arrêtée et que la transition est activée, l'exécution entrante arrêtée ne se poursuit pas dans la phase.

Vous pouvez consulter ou arrêter une exécution entrante.

Artefacts d'entrée et de sortie

CodePipeline s'intègre aux outils de développement pour vérifier les modifications apportées au code, puis créer et déployer à toutes les étapes du processus de livraison continue. Les artefacts sont les fichiers qui sont traités par des actions du pipeline, tels que des fichiers ou des dossiers contenant du code d'application, des fichiers de page d'index, des scripts, etc. Par exemple, l'artefact

d'action source Amazon S3 est un nom de fichier (ou chemin de fichier) dans lequel les fichiers de code source de l'application sont fournis pour l'action source du pipeline. Les fichiers sont fournis sous forme de fichier ZIP, comme l'exemple de nom d'artefact suivant :`SampleApp_Windows.zip`. L'artefact de sortie de l'action source, les fichiers de code source de l'application, est l'artefact de sortie de cette action et l'artefact d'entrée pour l'action suivante, telle qu'une action de génération. Autre exemple, une action de génération peut exécuter des commandes de génération qui compilent le code source de l'application pour un artefact d'entrée, à savoir les fichiers de code source de l'application. Consultez la page de référence de configuration des actions pour une action spécifique pour plus de détails sur les paramètres de l'artefact, par exemple AWS CodeBuild créer et tester une référence d'action pour l' CodeBuild action.

Les actions utilisent des artefacts d'entrée et de sortie qui sont stockés dans le compartiment d'artefacts Amazon S3 que vous avez choisi lors de la création du pipeline. CodePipeline compresse et transfère les fichiers pour les artefacts d'entrée ou de sortie, en fonction du type d'action de la scène.

> **Note**
>
> Le compartiment d'artefacts n'est pas le même que le compartiment utilisé comme emplacement du fichier source pour un pipeline où l'action source choisie est S3.

Par exemple :

1. CodePipeline déclenche l'exécution de votre pipeline lorsqu'il y a une validation dans le référentiel source, fournissant l'artefact de sortie (tous les fichiers à créer) depuis le stage Source.
2. L'artefact de sortie (tout fichier devant être généré) de l'étape précédente est ingéré en tant qu'artefact d'entrée à l'étape Build. Un artefact de sortie (application générée) issu de l'étape Build peut correspondre à une application mise à jour ou à une image Docker mise à jour et intégrée à un conteneur.
3. L'artefact de sortie de l'étape précédente (l'application créée) est ingéré en tant qu'artefact d'entrée dans la phase de déploiement, comme les environnements de préparation ou de production dans le. AWS Cloud Vous pouvez déployer des applications sur une flotte de déploiement, ou déployer des applications basées sur conteneur sur des tâches exécutés dans des clusters ECS.

Lorsque vous créez ou modifiez une action, vous définissez l'artefact ou les artefacts d'entrée et de sortie pour l'action. Par exemple, pour un pipeline en deux étapes avec une étape source et une

étape de déploiement, dans Modifier l'action, vous choisissez le nom de l'artefact de l'action source pour l'artefact en entrée pour l'action de déploiement.

- Lorsque vous utilisez la console pour créer votre premier pipeline, CodePipeline créez un compartiment Amazon S3 dans celui-ci Compte AWS et Région AWS pour stocker des éléments pour tous les pipelines. Chaque fois que vous utilisez la console pour créer un autre pipeline dans cette région, CodePipeline créez un dossier pour ce pipeline dans le bucket. Ce dossier est utilisé pour stocker les artéfacts pour votre pipeline pendant l'exécution du processus de publication automatisé. Ce compartiment est nommé codepipeline- *region* -*12345EXAMPLE-1a2b-3c4d-67890EXAMPLE*, où *region* est la AWS région dans laquelle vous avez créé le pipeline. *12345EXAMPLE-1a2b-3c4d-67890EXAMPLE* Il s'agit d'une valeur aléatoire basée sur un identifiant unique universel (UUID) qui garantit que le nom du compartiment est unique.

  > **Note**
  >
  > Si vous avez déjà un compartiment commençant par codepipeline-region - dans la région où vous créez le pipeline, CodePipeline utilisez-le comme compartiment par défaut. Il suit également l'ordre lexicographique ; par exemple, codepipeline- region-abcexample est choisi avant codepipeline- region-defexample.

  CodePipeline tronque les noms des artefacts, ce qui peut donner à certains noms de compartiments une apparence similaire. Même si le nom de l'artefact semble tronqué, il est CodePipeline mappé au compartiment d'artefacts d'une manière qui n'est pas affectée par les artefacts dont le nom est tronqué. Le pipeline peut fonctionner normalement. Ce n'est pas un problème avec le dossier ou les artefacts. Les noms de pipeline sont limités à 100 caractères. Bien que le nom du dossier de l'artefact puisse apparaître raccourci, il est toujours unique à votre pipeline.

  Lorsque vous créez ou modifiez un pipeline, vous devez disposer d'un compartiment d'artefacts dans le pipeline Compte AWS et Région AWS vous devez disposer d'un compartiment d'artefacts par région dans laquelle vous prévoyez d'exécuter une action. Si vous utilisez la console pour créer un pipeline ou des actions inter-régions, les compartiments d'artefact par défaut sont configurés par CodePipeline dans les régions où vous avez des actions.

  Si vous utilisez le AWS CLI pour créer un pipeline, vous pouvez stocker les artefacts de ce pipeline dans n'importe quel compartiment Amazon S3 tant que ce compartiment se trouve dans le même

Compte AWS compartiment Région AWS que le pipeline. Vous pouvez le faire si vous craignez de dépasser les limites des compartiments Amazon S3 autorisés pour votre compte. Si vous utilisez le AWS CLI pour créer ou modifier un pipeline et que vous ajoutez une action interrégionale (une action avec un AWS fournisseur dans une région différente de votre pipeline), vous devez fournir un compartiment d'artefacts pour chaque région supplémentaire dans laquelle vous prévoyez d'exécuter une action.

- Chaque action dispose d'un type. Selon le type, l'action peut avoir l'un des éléments suivants, ou les deux :
  - Un artéfact d'entrée, artéfact utilisé pendant l'exécution de l'action.
  - Une artéfact de sortie, qui est le résultat de l'action.

  Chaque artéfact de sortie dans le pipeline doit avoir un nom unique. Chaque artéfact d'entrée d'une action doit correspondre à l'artéfact de sortie d'une action précédente dans le pipeline, que cette action précède immédiatement l'action d'une étape ou s'exécute dans plusieurs étapes précédentes.

  Un artéfact peut s'appliquer à plusieurs actions.

Comment fonctionnent les conditions scéniques ?

Pour chaque condition qui spécifie une règle, celle-ci est exécutée. Si la condition échoue, l'étape exécute le résultat spécifié. Facultativement, dans le cadre de la règle, vous pouvez également spécifier les ressources CodePipeline à utiliser dans certains cas. Par exemple, la `CloudWatchAlarm` règle utilisera une ressource CloudWatch d'alarme pour vérifier la condition.

Une condition peut correspondre à plusieurs règles, et chaque règle peut spécifier l'un des trois fournisseurs.

Le flux de haut niveau pour créer des conditions est le suivant :

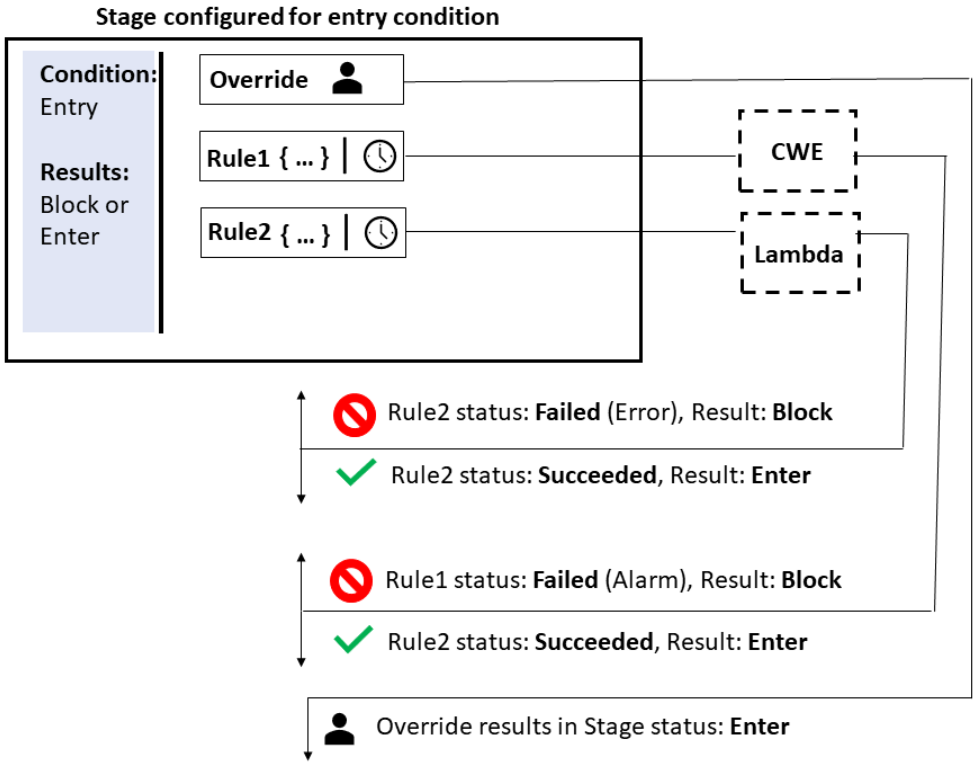
1. Choisissez le type de condition parmi les types de conditions disponibles dans CodePipeline. Par exemple, utilisez un type de condition On Success pour configurer une étape de sorte qu'une fois celle-ci réussie, un ensemble de règles puisse être utilisé pour effectuer des vérifications avant de continuer.

2. Choisissez la règle . Par exemple, la `CloudWatchAlarm` règle vérifie les alarmes et utilise EB pour vérifier un seuil d'alarme préconfiguré. Si la vérification est réussie et que l'alarme est inférieure au seuil, l'étape peut se poursuivre.
3. Configurez le résultat, tel qu'un rollback qui serait utilisé en cas d'échec de la règle.

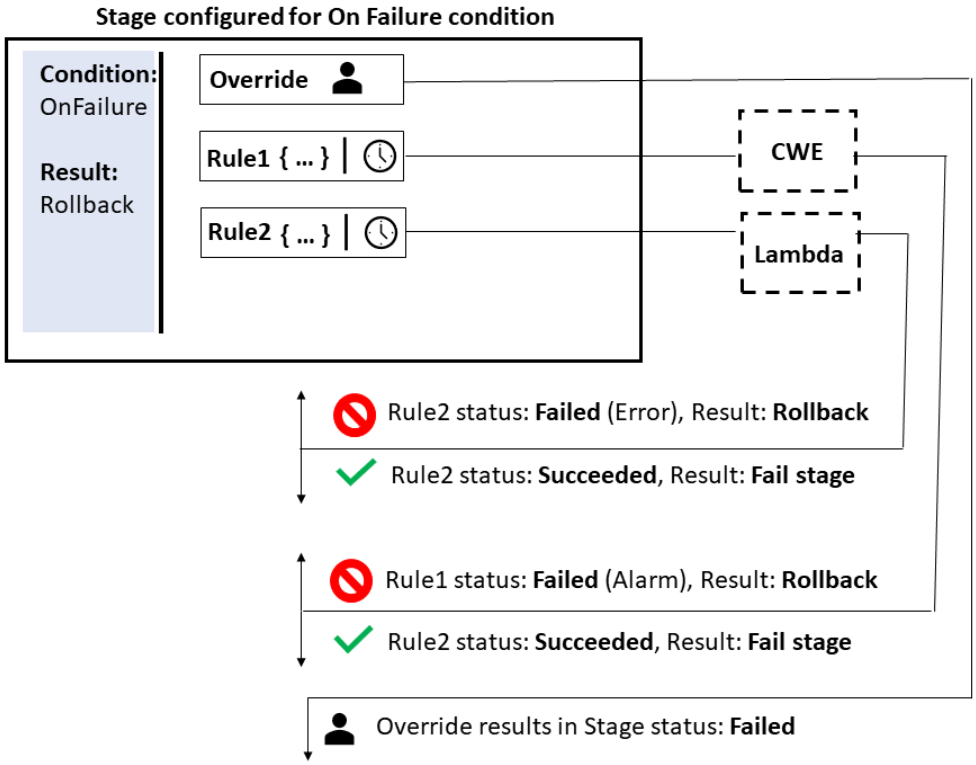
Les conditions sont utilisées pour des types d'expressions spécifiques et chacune possède des options spécifiques pour les résultats disponibles comme suit :

- Entrée - Les conditions pour effectuer des contrôles qui, si elles sont respectées, permettent d'accéder à une étape. Les règles sont activées avec les options de résultat suivantes : Echec ou Ignorer
- En cas d'échec - Les conditions pour effectuer des vérifications pour l'étape en cas d'échec. Les règles sont activées avec l'option de résultat suivante : Rollback
- En cas de succès - Les conditions pour effectuer des vérifications pour l'étape où elle réussit. Les règles sont activées avec les options de résultat suivantes : Rollback ou Fail

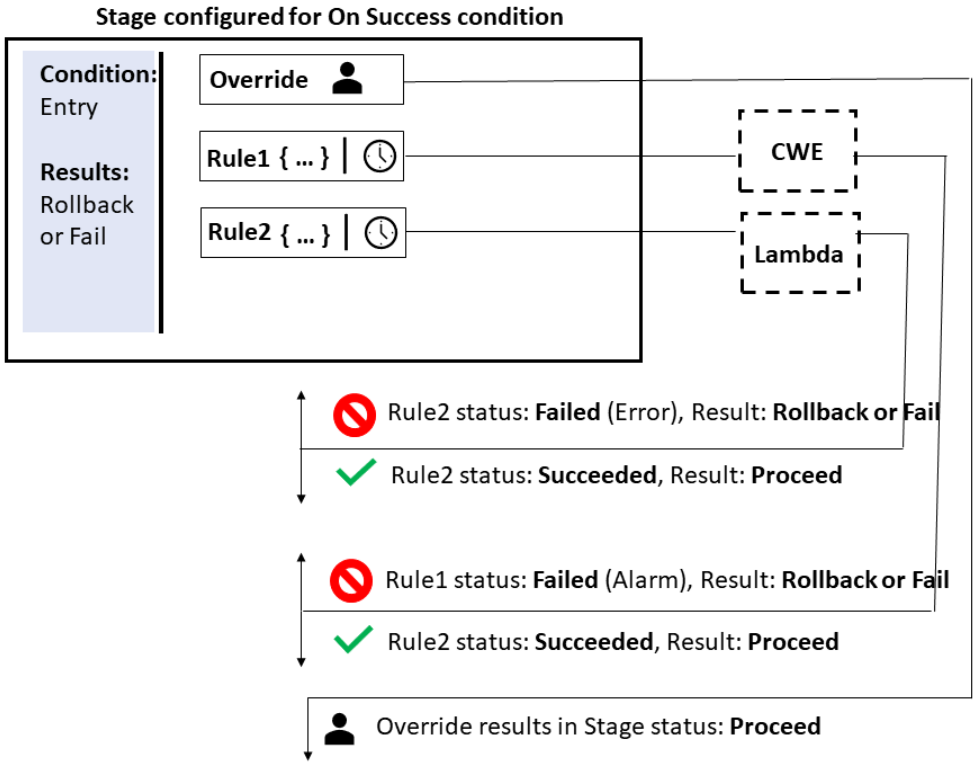
Le schéma suivant montre un exemple de flux pour le type de condition d'entrée dans CodePipeline. Les conditions répondent à la question suivante : que doit-il se passer si la condition n'est pas remplie, ce qui signifie qu'une règle échoue ? Dans le flux suivant, une condition d'entrée est configurée avec une LambdaInvoke règle et une `CloudWatchAlarm` règle. Si la règle échoue, le résultat configuré, tel que Fail, est activé.



Le schéma suivant montre un exemple de flux pour la condition En cas de défaillance de type in CodePipeline. Les conditions répondent à la question suivante : que doit-il se passer si la condition est remplie, ce qui signifie que les règles réussissent toutes leurs vérifications ? Dans le flux suivant, une condition En cas de défaillance est configurée avec une LambdaInvoke règle et une `CloudWatchAlarm` règle. Si la règle fonctionne, le résultat configuré, tel que Fail, est activé.



Le schéma suivant montre un exemple de flux pour la condition En cas de réussite, tapez dans CodePipeline. Les conditions répondent à la question suivante : que doit-il se passer si la condition est remplie, ce qui signifie que les règles réussissent toutes leurs vérifications ? Dans le flux suivant, une condition En cas de réussite est configurée avec une `LambdaInvoke` règle et une `CloudWatchAlarm` règle. Si la règle fonctionne, le résultat configuré, tel que Fail, est activé.



Règles relatives aux conditions de stage

Lorsque vous configurez les conditions de l'étape, vous effectuez une sélection parmi des règles prédéfinies et vous spécifiez les résultats de la règle. L'état d'une condition sera Echec si l'une des règles de la condition échoue et Réussi toutes les règles réussissent. La manière dont les critères sont remplis pour les conditions relatives à On Failure et On Success dépend du type de règle.

Vous trouverez ci-dessous des règles gérées que vous pouvez ajouter aux conditions d'étape.

- Les conditions peuvent utiliser la règle Commandes pour spécifier des commandes répondant aux critères de la règle en matière de conditions. Pour plus d'informations sur cette règle, consultezCommandes.
- Les conditions peuvent utiliser la AWS DeploymentWindow règle pour spécifier les délais de déploiement approuvés afin d'autoriser un déploiement. Les critères de la règle seront mesurés à l'aide d'une expression cron fournie pour une fenêtre de déploiement. La règle fonctionne lorsque la date et l'heure de la fenêtre de déploiement répondent aux critères de l'expression cron de la règle. Pour plus d'informations sur cette règle, consultezDeploymentWindow.
- Les conditions peuvent utiliser la règle AWS Lambda pour vérifier les états d'erreur renvoyés par les fonctions Lambda configurées. La règle est respectée lorsque la vérification reçoit le résultat

de la fonction Lambda. Une erreur provenant de la fonction Lambda répond aux critères des conditions On Failure. Pour plus d'informations sur cette règle, consultezLambdaInvoke.

- Les conditions peuvent utiliser la AWS CloudWatchAlarm règle pour vérifier les alarmes configurées à partir d' CloudWatch événements. La règle est respectée lorsque la vérification renvoie l'état d'alarme OK, ALARM ou INSUFFISANT_DATA. Pour les conditions On Success, OK et INSUFFISANT_DATA répondent aux critères. ALARM répond aux critères relatifs aux conditions de défaillance. Pour plus d'informations sur cette règle, consultezCloudWatchAlarm.
- Les conditions peuvent utiliser la VariableCheck règle pour créer une condition dans laquelle la variable de sortie est vérifiée par rapport à une expression fournie. La règle passe le contrôle lorsque la valeur de la variable répond aux critères de la règle, par exemple si la valeur est égale ou supérieure à une variable de sortie spécifiée. Pour plus d'informations sur cette règle, consultezVariableCheck.

Types de canalisations

CodePipeline fournit les types de pipelines suivants, dont les caractéristiques et les prix diffèrent, afin que vous puissiez adapter les caractéristiques et les coûts de votre pipeline aux besoins de vos applications.

- Les pipelines de type V1 ont une structure JSON qui contient des paramètres standard au niveau du pipeline, de l'étape et de l'action.
- Les pipelines de type V2 ont la même structure que les pipelines de type V1, ainsi que des paramètres supplémentaires pour la sécurité du déclenchement et la configuration des déclencheurs.

Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.

Consultez la CodePipeline référence de la structure du pipeline page pour plus de détails sur les paramètres de chaque type de pipeline. Pour plus d'informations sur le type de pipeline à choisir, consultezQuel type de pipeline me convient le mieux ?.

Quel type de pipeline me convient le mieux ?

Le type de pipeline est déterminé par l'ensemble de caractéristiques et de fonctionnalités prises en charge par chaque version de pipeline.

Voici un résumé des cas d'utilisation et des caractéristiques disponibles pour chaque type de pipeline.

	Type V1	Type V2
Caractéristiques		
Cas d'utilisation	• Déploiements standard	• Déploiements avec configuration à partir de variables transmises au niveau du pipeline lors de l'exécution • Déploiements dans lesquels les pipelines sont configurés pour démarrer sur des balises Git
Action-levelvariables	Pris en charge	Pris en charge
Mode d'exécution PARALLÈLE	Non pris en charge	Pris en charge
Pipeline-level variables	Non pris en charge	Pris en charge
Mode d'exécution QUEUED	Non pris en charge	Pris en charge
Annulation des étapes du pipeline	Non pris en charge	Pris en charge
Dérogations à la révision de la source	Non pris en charge	Pris en charge
Conditions de l'étape	Non pris en charge	Pris en charge
Déclenche et filtre les balises Git, les pull requests, les branches ou les chemins de fichiers	Non pris en charge	Pris en charge

	Type V1	Type V2
Caractéristiques		
L'action Commandes	Non pris en charge	Pris en charge
Création de conditions de saisie avec Ignorer le résultat	Non pris en charge	Pris en charge
Configurer une étape pour une nouvelle tentative automatique en cas d'échec	Non pris en charge	Pris en charge

Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.

Vous pouvez créer et exécuter un script Python pour vous aider à analyser le coût potentiel du déplacement d'un pipeline de type V1 vers un pipeline de type V2.

> **Note**
>
> L'exemple de script ci-dessous est uniquement destiné à des fins de démonstration et d'évaluation. Il ne s'agit pas d'un outil de devis et ne garantit pas le coût de votre utilisation réelle d'un pipeline de type V2, et il n'inclut pas les taxes qui pourraient s'appliquer. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.

Pour créer et exécuter un script qui vous aidera à évaluer le coût du déplacement d'un pipeline de type V1 vers un pipeline de type V2

1. Téléchargez et installez Python.
2. Ouvrez une fenêtre du terminal. Exécutez la commande suivante pour créer un nouveau script Python nommé PipelineCostAnalyzer.py.

   ```
   vi PipelineCostAnalyzer.py
   ```

3. Copiez et collez le code suivant dans le 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. À partir du terminal ou de l'invite de commande, modifiez les répertoires dans lesquels vous avez créé le script d'analyse.

   À partir de ce répertoire, exécutez la commande suivante, où region est l' Région AWS endroit où vous avez créé les pipelines V1 que vous souhaitez analyser. Vous pouvez également évaluer éventuellement un pipeline spécifique en indiquant son nom :

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

   Par exemple, exécutez la commande suivante pour exécuter le script python nommé PipelineCostAnalyzer.py. Dans cet exemple, la région est`us-west-2`.

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

> **Note**
>
> Ce script analysera tous les pipelines V1 de la zone spécifiée, Région AWS sauf si vous spécifiez un nom de pipeline spécifique.

5. Dans l'exemple de sortie du script suivant, nous pouvons voir la liste des exécutions d'actions, la liste des exécutions d'actions éligibles à la facturation, le temps d'exécution total de ces exécutions d'actions et le coût estimé de ces actions telles qu'elles sont effectuées dans un 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
```

Dans cet exemple, la valeur négative de la dernière ligne représente le montant estimé qui pourrait être économisé en passant à des pipelines de type V2.

> **Note**
>
> La sortie du script et les exemples associés qui montrent les coûts et d'autres informations ne sont que des estimations. Ils sont destinés à des fins de démonstration et d'évaluation uniquement et ne garantissent aucune économie réelle. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.

Commencer avec CodePipeline

Si vous êtes nouveau dans ce CodePipeline domaine, vous pouvez suivre les didacticiels de ce guide après avoir suivi les étapes de ce chapitre pour vous installer.

La CodePipeline console inclut des informations utiles dans un panneau pliable que vous pouvez ouvrir à partir de l'icône d'informations ou de n'importe quel lien d'information sur la page.

().

Vous pouvez fermer ce panneau à tout moment.

La CodePipeline console permet également de rechercher rapidement vos ressources, telles que des référentiels, des projets de création, des applications de déploiement et des pipelines. Choisissez Go to ressource (Accéder aux ressources) ou appuyez sur la touche /, puis saisissez le nom de la ressource. Toutes les correspondances s'affichent dans la liste. Les recherches ne sont pas sensibles à la casse. Vous pouvez uniquement consulter les ressources que vous êtes autorisé à afficher. Pour plus d'informations, consultez Affichage des ressources dans la console.

Avant de pouvoir l'utiliser AWS CodePipeline pour la première fois, vous devez créer votre Compte AWS et créer votre premier utilisateur administratif.

Rubriques

- Inscrivez-vous pour un Compte AWS
- Appliquer une politique gérée pour l'accès administratif à CodePipeline
- Installer la AWS CLI
- Ouvrez la console pour CodePipeline
- Étapes suivantes

Inscrivez-vous pour un Compte AWS

Pour commencer AWS, vous avez besoin d'un Compte AWS. Il existe deux façons de s'inscrire à AWS. Si vous souhaitez contrôler totalement la configuration de votre compte et l'emplacement de stockage de votre contenu, ou si vous avez des charges de travail réglementées telles que HIPAA ou FedRAMP, utilisez Sign up for (advanced). AWS Si vous êtes un nouvel utilisateur et que vous souhaitez utiliser un identifiant que vous possédez déjà pour vous connecter AWS et créer un AWS

environnement avec des paramètres par défaut préconfigurés dans Select Régions AWS, utilisez S'inscrire pour. AWS

Appliquer une politique gérée pour l'accès administratif à CodePipeline

Vous devez accorder des autorisations pour interagir avec CodePipeline. La méthode la plus rapide consiste à appliquer la politique `AWSCodePipeline_FullAccess` gérée à l'utilisateur administratif.

> **Note**
>
> La `AWSCodePipeline_FullAccess` politique inclut des autorisations qui permettent à l'utilisateur de la console de transmettre un rôle IAM CodePipeline ou à un autre Services AWS. Cela autorise le service à assumer le rôle et à effectuer des actions en votre nom. Lorsque vous attachez la stratégie à un utilisateur, un rôle ou un groupe, les autorisations `iam:PassRole` sont appliquées. Assurez-vous que la stratégie est uniquement appliquée à des utilisateurs de confiance. Lorsque les utilisateurs disposant de ces autorisations utilisent la console pour créer ou modifier un pipeline, les options suivantes sont disponibles :
>
> - Créez un rôle CodePipeline de service ou choisissez-en un existant et transmettez-le à CodePipeline
> - Vous pourriez choisir de créer une règle d' CloudWatch événements pour détecter les modifications et de transmettre le rôle de service d' CloudWatch CloudWatch événements à Events
>
> Pour plus d'informations, consultez la section Octroi à un utilisateur des autorisations lui permettant de transmettre un rôle à un Service AWS.

> **Note**
>
> La `AWSCodePipeline_FullAccess` politique donne accès à toutes les CodePipeline actions et ressources auxquelles l'utilisateur IAM a accès, ainsi qu'à toutes les actions possibles lors de la création d'étapes dans un pipeline, telles que la création d'étapes incluant CodeDeploy Elastic Beanstalk ou Amazon S3. La bonne pratique consiste à accorder aux personnes autorisées uniquement les autorisations dont elles ont besoin pour réaliser leur travail. Pour plus d'informations sur la façon de restreindre les utilisateurs IAM à un ensemble

limité d' CodePipeline actions et de ressources, consultezSupprimer les autorisations associées au rôle CodePipeline de service.

Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

  Créez un jeu d'autorisations. Suivez les instructions de la rubrique Création d'un jeu d'autorisations du Guide de l'utilisateur AWS IAM Identity Center .

- Utilisateurs gérés dans IAM par un fournisseur d'identité :

  Créez un rôle pour la fédération d'identité. Suivez les instructions de la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM.

- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM dans le Guide de l'utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

Installer la AWS CLI

Pour appeler CodePipeline des commandes depuis la AWS CLI sur une machine de développement locale, vous devez installer la AWS CLI. Cette étape est facultative si vous souhaitez commencer à suivre uniquement les étapes de ce guide pour la CodePipeline console.

Pour installer et configurer AWS CLI

1. Sur votre ordinateur local, téléchargez et installez le AWS CLI. Cela vous permettra d'interagir avec vous CodePipeline depuis la ligne de commande. Pour plus d'informations, voir Configuration avec l'interface de ligne de AWS commande.

   > **Note**
   >
   > CodePipeline fonctionne uniquement avec AWS CLI les versions 1.7.38 et ultérieures. Pour déterminer la version AWS CLI que vous avez peut-être installée, exécutez la

commandeaws --version. Pour mettre à niveau une ancienne version du AWS CLI vers la dernière version, suivez les instructions de la section Désinstallation du AWS CLI, puis suivez les instructions de la section Installation du AWS Command Line Interface.

2. Configurez le AWS CLI avec la configure commande, comme suit :

```
aws configure
```

Lorsque vous y êtes invité, spécifiez la clé AWS d'accès et la clé d'accès AWS secrète de l'utilisateur IAM que vous utiliserez avec CodePipeline. Lorsque vous êtes invité à saisir le nom de région par défaut, spécifiez la région dans laquelle vous allez créer le pipeline, par exemple`us-east-2`. Lorsque vous êtes invité à saisir le format de sortie par défaut, entrez `json`. Par exemple :

```
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**
>
> Pour plus d'informations sur l'IAM, les clés d'accès et les clés secrètes, consultez les sections Gestion des clés d'accès pour les utilisateurs IAM et Comment obtenir des informations d'identification ? .
> Pour plus d'informations sur les régions et les points de terminaison disponibles CodePipeline, consultez la section AWS CodePipeline Points de terminaison et quotas.

Ouvrez la console pour CodePipeline

- Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

Étapes suivantes

Vous réunissez toutes les conditions prérequises. Vous pouvez commencer à utiliser CodePipeline. Pour commencer à utiliser CodePipeline consultez le CodePipeline tutoriels.

Intégrations de produits et de services avec CodePipeline

Par défaut, AWS CodePipeline est intégré à un certain nombre de Services AWS produits et services partenaires. Utilisez les informations contenues dans les sections suivantes pour vous aider CodePipeline à configurer et à intégrer les produits et services que vous utilisez.

Les ressources connexes suivantes peuvent s'avérer utiles lors de l'utilisation de ce service.

Rubriques

- Intégrations avec les types CodePipeline d'action
- Intégrations générales avec CodePipeline
- Exemples issus de la communauté

Intégrations avec les types CodePipeline d'action

Les informations sur les intégrations présentées dans cette rubrique sont organisées par type CodePipeline d'action.

Rubriques

- Intégrations d'actions source
- Intégrations d'actions de génération
- Intégrations d'actions de test
- Intégrations d'actions de déploiement
- Intégration des actions d'approbation avec Amazon Simple Notification Service
- Intégrations d'actions d'appel

Intégrations d'actions source

Les informations suivantes sont organisées par type CodePipeline d'action et peuvent vous aider CodePipeline à configurer l'intégration avec les fournisseurs d'actions source suivants.

Rubriques

- Actions relatives aux sources Amazon ECR
- Actions relatives aux sources Amazon S3

- Connexions au cloud Bitbucket GitHub (via une GitHub application), au serveur GitHub d'entreprise et GitLab à l' GitLab.comautogestion
- CodeCommit actions à la source
- GitHub (via l'application OAuth) actions source

Actions relatives aux sources Amazon ECR

Amazon ECR est un service de référentiel d'images AWS Docker. Vous pouvez utiliser les commandes Docker de transfert et d'extraction pour charger des images Docker dans votre référentiel. Une URI et une image du référentiel Amazon ECR sont utilisées dans les définitions de tâches Amazon ECS pour référencer les informations d'image source.

En savoir plus :

- Pour consulter les paramètres de configuration et un exemple d' JSON/YAML extrait de code, voir Référence des actions source Amazon ECR
- Création d'un pipeline, d'étapes et d'actions
- Tutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR

Actions relatives aux sources Amazon S3

Amazon S3 est un espace de stockage pour Internet. Vous pouvez utiliser Amazon S3 pour stocker et récupérer n'importe quelle quantité de données, n'importe quand et depuis n'importe quel emplacement sur le Web. Vous pouvez configurer CodePipeline pour utiliser un compartiment Amazon S3 versionné comme action source pour votre code.

> **Note**
>
> Amazon S3 peut également être inclus dans un pipeline en tant qu'action de déploiement.

En savoir plus :

- Pour consulter les paramètres de configuration et un exemple d' JSON/YAML extrait de code, voir Référence des actions source d'Amazon S3
- Étape 1 : créer un compartiment source S3 pour votre application

- Création d'un pipeline (interface de ligne de commande)
- CodePipeline utilise Amazon EventBridge (anciennement Amazon CloudWatch Events) pour détecter les modifications apportées à votre compartiment source Amazon S3. Consultez Intégrations générales avec CodePipeline.

Connexions au cloud Bitbucket GitHub (via une GitHub application), au serveur GitHub d'entreprise et GitLab à l' GitLab.comautogestion

Les connexions (`CodeStarSourceConnection`actions) sont utilisées pour accéder à votre Bitbucket Cloud tiers GitHub, à votre serveur GitHub d'entreprise ou à votre référentiel GitLab autogéré. GitLab.com

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans les régions Asie-Pacifique (Hong Kong), Asie-Pacifique (Hyderabad), Asie-Pacifique (Jakarta), Asie-Pacifique (Melbourne), Asie-Pacifique (Osaka), Afrique (Le Cap), Moyen-Orient (Bahreïn), Moyen-Orient (Émirats arabes unis), Europe (Espagne), Europe (Zurich), Israël (Tel Aviv) ou (). AWS GovCloud US-West Pour faire référence aux autres actions disponibles, voirIntégrations de produits et de services avec CodePipeline. Pour les considérations relatives à cette action dans la région Europe (Milan), voir la note dansCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

Nuage Bitbucket	Vous pouvez configurer CodePipeline pour utiliser un référentiel Bitbucket Cloud comme source de votre code. Vous devez avoir préalablement créé un compte Bitbucket et au moins un dépôt Bitbucket Cloud. Vous pouvez ajouter une action source pour votre dépôt Bitbucket Cloud en créant un pipeline ou en modifiant un pipeline existant. **Note** Vous pouvez créer des connexions à un référentiel Bitbucket Cloud. Les types de fournisseurs Bitbucket installés, tels que Bitbucket Server, ne sont pas pris en charge.

	Vous pouvez configurer des ressources appelées connexions pour permettre à vos pipelines d'accéder à des référentiels de code tiers. Lorsque vous créez une connexion, vous installez l'application Connector dans votre référentiel de code tiers, puis vous l'associez à votre connexion. Pour Bitbucket Cloud, utilisez l'option Bitbucket dans la console ou l'`CodestarSourceConnection` action dans la CLI. Consultez Connexions Bitbucket Cloud. Vous pouvez utiliser l'option Clonage complet pour cette action afin de référencer les métadonnées Git du référentiel afin que les actions en aval puissent exécuter directement les commandes Git. Cette option ne peut être utilisée que par des actions CodeBuild en aval. En savoir plus : • Pour consulter les paramètres de configuration et un exemple d' JSON/YAML extrait de code, consultez. CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées • Pour consulter un didacticiel de démarrage qui crée un pipeline avec une source Bitbucket Cloud, consultez Getting started with connections.
GitHub ou GitHub Enterprise Cloud	Vous pouvez configurer CodePipeline pour utiliser un GitHub référentiel comme source de votre code. Vous devez avoir préalablement créé un GitHub compte et au moins un GitHub dépôt. Vous pouvez ajouter une action source pour votre GitHub référentiel en créant un pipeline ou en modifiant un pipeline existant. Vous pouvez configurer des ressources appelées connexions pour permettre à vos pipelines d'accéder à des référentiels de code tiers. Lorsque vous créez une connexion, vous installez l'application Connector dans votre référentiel de code tiers, puis vous l'associez à votre connexion. Utilisez l'option fournisseur GitHub (via GitHub App) dans la console ou l'`CodestarSourceConnection` action dans la CLI. Consultez GitHub connexions.

	Vous pouvez utiliser l'option Clonage complet pour cette action afin de référencer les métadonnées Git du référentiel afin que les actions en aval puissent exécuter directement les commandes Git. Cette option ne peut être utilisée que par des actions CodeBuild en aval. En savoir plus : • Pour consulter les paramètres de configuration et un exemple d' JSON/YAML extrait de code, voir CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées • Pour consulter un didacticiel expliquant comment se connecter à un GitHub dépôt et utiliser l'option de clonage complet, voirTutoriel : Utiliser un clone complet avec une source de GitHub pipeline. • L'action en cours GitHub (via GitHub l'application) est l'action source de la version 2 pour GitHub. L'action GitHub (via l'application OAuth) est l'action de version 1 GitHub gérée avec l'authentification par jeton OAuth. Bien que nous ne recommandions pas d'utiliser l'action GitHub (via l'applica tion OAuth), les pipelines existants dotés de l'action GitHub (via l'applica tion OAuth) continueront de fonctionner sans aucun impact. Vous pouvez désormais utiliser une action CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées source dans votre pipeline qui gère votre action GitHub source avec GitHub les applications. Si vous avez un pipeline qui utilise l'action GitHub (via l'application OAuth), consultez les étapes pour le mettre à jour afin d'utiliser une action GitHub (via GitHub l'application) dans. Mettre à jour une action source GitHub (via l'application OAuth) en une action source GitHub (via GitHub l'application)
GitHub Serveur d'entreprise	Vous pouvez configurer CodePipeline pour utiliser un référentiel GitHub Enterprise Server comme source de votre code. Vous devez avoir préalablement créé un GitHub compte et au moins un GitHub dépôt. Vous pouvez ajouter une action source pour votre référentiel GitHub Enterprise Server en créant un pipeline ou en modifiant un pipeline existant.

	Vous pouvez configurer des ressources appelées connexions pour permettre à vos pipelines d'accéder à des référentiels de code tiers. Lorsque vous créez une connexion, vous installez l'application Connector dans votre référentiel de code tiers, puis vous l'associez à votre connexion. Utilisez l'option GitHub Enterprise Server provider dans la console ou l'`CodestarSourceConnection` action dans la CLI. Consultez GitHub Connexions aux serveurs d'entreprise. Vous pouvez utiliser l'option Clonage complet pour cette action afin de référencer les métadonnées Git du référentiel afin que les actions en aval puissent exécuter directement les commandes Git. Cette option ne peut être utilisée que par des actions CodeBuild en aval. En savoir plus : • Pour consulter les paramètres de configuration et un exemple d' JSON/YAML extrait de code, voir CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées • Pour consulter un didacticiel expliquant comment se connecter à un GitHub dépôt et utiliser l'option de clonage complet, voirTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.
GitLab.com	Vous pouvez configurer CodePipeline pour utiliser un GitLab.com référenti el comme source de votre code. Vous devez avoir préalablement créé un GitLab.com compte et au moins un GitLab.com dépôt. Vous pouvez ajouter une action source pour votre GitLab.com référentiel en créant un pipeline ou en modifiant un pipeline existant. Utilisez l'option GitLabfournisseur dans la console ou l'`CodestarS ourceConnection` action avec le `GitLab` fournisseur dans la CLI. Consultez GitLab.com connexions. En savoir plus :

	• Pour consulter les paramètres de configuration et un exemple d' JSON/YAML extrait de code, voir CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées
GitLab autogéré	Vous pouvez configurer CodePipeline pour utiliser une installation GitLab autogérée comme source de votre code. Vous devez avoir préalablement créé un GitLab compte et disposer d'un abonnement autogéré GitLab (Enterprise Edition ou Community Edition). Vous pouvez ajouter une action source pour votre référentiel GitLab autogéré en créant un pipeline ou en modifiant un pipeline existant. Vous pouvez configurer des ressources appelées connexions pour permettre à vos pipelines d'accéder à des référentiels de code tiers. Lorsque vous créez une connexion, vous installez l'application Connector dans votre référentiel de code tiers, puis vous l'associez à votre connexion. Utilisez l'option fournisseur GitLab autogéré dans la console ou l'`CodestarSourceConnection` action dans la CLI. Consultez Connexions pour l' GitLab autogestion. Vous pouvez utiliser l'option Clonage complet pour cette action afin de référencer les métadonnées Git du référentiel afin que les actions en aval puissent exécuter directement les commandes Git. Cette option ne peut être utilisée que par des actions CodeBuild en aval. En savoir plus : • Pour consulter les paramètres de configuration et un exemple d' JSON/YAML extrait de code, voir CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées • Pour connaître les étapes à suivre pour créer une connexion avec ce type de fournisseur, consultezConnexions pour l' GitLab autogestion.

CodeCommit actions à la source

CodeCommit est un service de contrôle des versions que vous pouvez utiliser pour stocker et gérer les ressources de façon privée (par exemple, les documents, le code source et les fichiers binaires) dans le cloud. Vous pouvez configurer CodePipeline pour utiliser une branche d'un CodeCommit référentiel comme source de votre code. Créez le référentiel et associez-le à un répertoire de travail sur votre machine locale. Ensuite, vous pouvez créer un pipeline qui utilise la branche dans le cadre d'une action source dans une étape. Vous pouvez vous connecter au CodeCommit référentiel en créant un pipeline ou en modifiant un pipeline existant.

Vous pouvez utiliser l'option Clonage complet pour cette action afin de référencer les métadonnées Git du référentiel afin que les actions en aval puissent exécuter directement les commandes Git. Cette option ne peut être utilisée que par des actions CodeBuild en aval.

En savoir plus :

- Pour consulter les paramètres de configuration et un exemple d' JSON/YAML extrait de code, consultez. CodeCommit référence de l'action source
- Tutoriel : Création d'un pipeline simple (CodeCommit référentiel)
- CodePipeline utilise Amazon CloudWatch Events pour détecter les modifications apportées CodeCommit aux référentiels utilisés comme source pour un pipeline. Chaque action source correspond à une règle d'événement. Cette règle d'événement lance votre pipeline lorsqu'une modification survient dans le référentiel. Consultez Intégrations générales avec CodePipeline.

GitHub (via l'application OAuth) actions source

L'action GitHub (via l'application OAuth) est l'action de version 1 GitHub gérée avec les applications OAuth. Dans les régions disponibles, vous pouvez également utiliser une action CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées source dans votre pipeline qui gère votre action GitHub source avec GitHub Apps. Si vous avez un pipeline qui utilise l'action GitHub (via l'application OAuth), consultez les étapes pour le mettre à jour afin d'utiliser une action GitHub (via GitHub l'application) dans. Mettre à jour une action source GitHub (via l'application OAuth) en une action source GitHub (via GitHub l'application)

> **Note**
>
> Bien que nous ne recommandions pas d'utiliser l'action GitHub (via l'application OAuth), les pipelines existants dotés de l'action GitHub (via l'application OAuth) continueront de fonctionner sans aucun impact.

En savoir plus :

- Pour plus d'informations sur l'accès OAuth-based GitHub (via l'application OAuth) par rapport à l' GitHub accès basé sur l'application, voir. https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps
- Pour consulter une annexe contenant les détails de l'action GitHub (via l'application OAuth), consultez. Annexe A : actions relatives aux sources GitHub (via l'application OAuth)

Intégrations d'actions de génération

Les informations suivantes sont organisées par type CodePipeline d'action et peuvent vous aider CodePipeline à configurer l'intégration avec les fournisseurs d'actions de build suivants.

Rubriques

- CodeBuild créer des actions
- CloudBees créer des actions
- Actions de création et de publication Amazon ECR
- Les actions de génération Jenkins
- TeamCity créer des actions

CodeBuild créer des actions

CodeBuild est un service de génération entièrement géré qui compile votre code source, exécute des tests unitaires et produit des artefacts prêts à déployer.

Vous pouvez l'ajouter CodeBuild en tant qu'action de construction à la phase de construction d'un pipeline. Pour plus d'informations, consultez la référence de configuration des CodePipeline actions pourAWS CodeBuild créer et tester une référence d'action.

> **Note**
>
> CodeBuild peut également être inclus dans un pipeline en tant qu'action de test, avec ou sans sortie de compilation.

En savoir plus :

- Pour consulter les paramètres de configuration et un exemple d' JSON/YAML extrait de code, consultez. AWS CodeBuild créer et tester une référence d'action
- Qu'est-ce que c'est CodeBuild ?
- CodeBuild— Service de construction entièrement géré

CloudBees créer des actions

Vous pouvez le configurer CodePipeline pour CloudBeescréer ou tester votre code dans le cadre d'une ou de plusieurs actions d'un pipeline.

En savoir plus :

- re:Invent 2017 : une première dans le cloud avec AWS

Actions de création et de publication Amazon ECR

Amazon ECR est un service de référentiel d'images AWS Docker. Vous pouvez utiliser les commandes Docker de transfert et d'extraction pour charger des images Docker dans votre référentiel.

Vous pouvez ajouter l'`ECRBuildAndPublish`action à votre pipeline pour automatiser la création et la diffusion d'une image. Pour plus d'informations, consultez la référence de configuration des CodePipeline actions pour`ECRBuildAndPublish`créer une référence d'action.

Les actions de génération Jenkins

Vous pouvez configurer l'utilisation CodePipeline de Jenkins CI pour créer ou tester votre code dans le cadre d'une ou de plusieurs actions d'un pipeline. Vous devez avoir préalablement créé un projet Jenkins et installé et configuré le CodePipeline plugin pour Jenkins pour ce projet. Vous pouvez vous connecter au projet Jenkins en créant un nouveau pipeline ou en modifiant un pipeline existant.

L'accès à Jenkins est configuré pour chaque projet. Vous devez installer le CodePipeline plugin pour Jenkins sur chaque instance Jenkins que vous souhaitez utiliser. CodePipeline Vous devez également configurer CodePipeline l'accès au projet Jenkins. Sécurisez votre projet Jenkins en le configurant pour accepter uniquement HTTPS/SSL les connexions. Si votre projet Jenkins est installé sur une instance Amazon EC2, pensez à fournir AWS vos informations d'identification en AWS CLI les installant sur chaque instance. Configurez ensuite un AWS profil sur ces instances avec les informations d'identification que vous souhaitez utiliser pour les connexions. Ceci est une alternative à l'ajout et au stockage via l'interface web de Jenkins.

En savoir plus :

- Accès à Jenkins
- Didacticiel : Création d'un pipeline à quatre étapes

TeamCity créer des actions

Vous pouvez le configurer CodePipeline pour TeamCitycréer et tester votre code dans le cadre d'une ou de plusieurs actions d'un pipeline.

En savoir plus :

- TeamCity Plugin pour CodePipeline

Intégrations d'actions de test

Les informations suivantes sont organisées par type CodePipeline d'action et peuvent vous aider CodePipeline à configurer l'intégration avec les fournisseurs d'actions de test suivants.

Rubriques

- CodeBuild actions de test
- AWS Device Farm actions de test
- Actions de test de Ghost Inspector
- OpenText LoadRunner Actions de test dans le cloud
- Refléter l'automatisation des tests

CodeBuild actions de test

CodeBuildest un service de création entièrement géré dans le cloud. CodeBuild compile votre code source, exécute des tests unitaires et produit des artefacts prêts à être déployés.

Vous pouvez l'ajouter CodeBuild à un pipeline en tant qu'action de test. Pour plus d'informations, consultez la référence de configuration d'action CodePipeline pour AWS CodeBuild créer et tester une référence d'action.

> **Note**
>
> CodeBuild peut également être incluse dans un pipeline en tant qu'action de construction, avec un artefact de sortie de construction obligatoire.

En savoir plus :

- Pour consulter les paramètres de configuration et un exemple d' JSON/YAML extrait de code, consultez. AWS CodeBuild créer et tester une référence d'action
- Qu'est-ce que c'est CodeBuild ?

AWS Device Farm actions de test

AWS Device Farm est un service de test d'applications que vous pouvez utiliser pour tester et interagir avec vos applications Android, iOS et web sur de véritables téléphones et tablettes physiques. Vous pouvez le configurer CodePipeline AWS Device Farm pour tester votre code dans le cadre d'une ou de plusieurs actions d'un pipeline. AWS Device Farm vous permet de télécharger vos propres tests ou d'utiliser des tests de compatibilité intégrés et sans script. Étant donné que les tests sont exécutés en parallèle, les tests sur plusieurs appareils commencent en quelques minutes. Un rapport de test contenant des résultats de haut niveau, des journaux de bas niveau, des captures d'écran pixel par pixel et des données de performance est mis à jour au fur et à mesure que les tests sont terminés. AWS Device Farm prend en charge les tests d'applications Android, iOS et Fire OS natives et hybrides, y compris celles créées avec Titanium PhoneGap, Xamarin, Unity et d'autres frameworks. Il prend en charge l'accès à distance aux applications Android, ce qui vous permet d'interagir directement avec les appareils de test.

En savoir plus :

- Pour consulter les paramètres de configuration et un exemple d' JSON/YAML extrait de code, consultez. AWS Device Farm référence de l'action de test
- Qu'est-ce que c'est AWS Device Farm ?
- Utilisation AWS Device Farm lors d'une phase CodePipeline de test

Actions de test de Ghost Inspector

Vous pouvez CodePipeline configurer Ghost Inspector pour tester votre code lors d'une ou de plusieurs actions dans un pipeline.

En savoir plus :

- Documentation de Ghost Inspector pour l'intégration des services avec CodePipeline

OpenText LoadRunner Actions de test dans le cloud

Vous pouvez configurer CodePipeline pour utiliser OpenText LoadRunner Cloud dans une ou plusieurs actions d'un pipeline.

En savoir plus :

- LoadRunner Documentation sur le cloud pour l'intégration avec CodePipeline

Refléter l'automatisation des tests

Reflect est la solution d'automatisation des AI-powered tests qui vous permet de simplifier les tests et de surmonter les défis liés aux processus manuels. Grâce à l'automatisation des tests sans code, Reflect rationalise la création, l'exécution et la maintenance des tests, ce qui vous permet de créer des tests robustes et reproductibles sans avoir besoin de connaissances techniques. En éliminant la complexité et en minimisant les perturbations de vos flux de travail, il vous aide à accélérer les tests et à fournir des applications de haute qualité en toute confiance à chaque fois.

En savoir plus :

- AWS CodePipeline tester l'intégration avec Reflect

Intégrations d'actions de déploiement

Les informations suivantes sont organisées par type CodePipeline d'action et peuvent vous aider CodePipeline à configurer l'intégration avec les fournisseurs d'actions de déploiement suivants.

Rubriques

- Actions de déploiement d'Amazon EC2
- Actions de déploiement d'Amazon Elastic Kubernetes Service EKS
- Actions de déploiement d'Amazon S3
- AWS AppConfig actions de déploiement
- CloudFormation actions de déploiement
- CloudFormation StackSets actions de déploiement
- Actions de déploiement d'Amazon ECS
- Actions de déploiement d'Elastic Beanstalk
- OpsWorks actions de déploiement
- Actions de déploiement du Service Catalog
- Actions de déploiement d'Amazon Alexa
- CodeDeploy actions de déploiement
- XebiaLabs actions de déploiement

Actions de déploiement d'Amazon EC2

Amazon EC2 vous permet de créer et de gérer le calcul dans le cloud. Vous pouvez ajouter une action à un pipeline qui utilise Amazon EC2 comme fournisseur de déploiement pour déployer votre application sur vos instances.

En savoir plus :

- Consultez la page de référence des actions à l'adresseRéférence des actions Amazon EC2.
- Pour obtenir un didacticiel, consultez Tutoriel : Déploiement sur des instances Amazon EC2 avec CodePipeline.

Actions de déploiement d'Amazon Elastic Kubernetes Service EKS

Amazon EKS vous permet de créer et de gérer des clusters Kubernetes. Vous pouvez ajouter une action à un pipeline qui utilise Amazon EKS comme fournisseur de déploiement pour déployer votre image sur votre cluster. Vous pouvez utiliser des modèles Helm ou des fichiers manifestes Kubernetes.

En savoir plus :

- Consultez la page de référence des actions à l'adresseRéférence des actions de déploiement d'Amazon Elastic Kubernetes Service EKS.
- Pour obtenir un didacticiel, consultez Tutoriel : Déploiement sur Amazon EKS avec CodePipeline.

Actions de déploiement d'Amazon S3

Amazon S3 est un espace de stockage pour Internet. Vous pouvez utiliser Amazon S3 pour stocker et récupérer n'importe quelle quantité de données, n'importe quand et depuis n'importe quel emplacement sur le Web. Vous pouvez ajouter une action à un pipeline qui utilise Amazon S3 comme fournisseur de déploiement.

> **Note**
>
> Amazon S3 peut également être inclus dans un pipeline en tant qu'action source.

En savoir plus :

- Création d'un pipeline, d'étapes et d'actions
- Tutoriel : Création d'un pipeline qui utilise Amazon S3 comme fournisseur de déploiement

AWS AppConfig actions de déploiement

AWS AppConfig est une fonctionnalité permettant de AWS Systems Manager créer, de gérer et de déployer rapidement des configurations d'applications. Vous pouvez l'utiliser AppConfig avec des applications hébergées sur des instances EC2 AWS Lambda, des conteneurs, des applications mobiles ou des appareils IoT.

En savoir plus :

- CodePipeline Référence de configuration d'action pour AWS AppConfig référence d'action de déploiement
- Tutoriel : Création d'un pipeline utilisé AWS AppConfig comme fournisseur de déploiement

CloudFormation actions de déploiement

CloudFormationpermet aux développeurs et aux administrateurs système de créer et de gérer facilement un ensemble de AWS ressources connexes, en utilisant des modèles pour fournir et mettre à jour ces ressources. Vous pouvez utiliser les exemples de modèles du service ou créer les vôtres. Les modèles décrivent les AWS ressources et les dépendances ou paramètres d'exécution nécessaires pour exécuter votre application.

Le modèle d'application AWS sans serveur (AWS SAM) s'étend CloudFormation pour fournir un moyen simplifié de définir et de déployer des applications sans serveur. AWS SAM prend en charge les API Amazon API Gateway, les fonctions AWS Lambda et les tables Amazon DynamoDB. Vous pouvez utiliser CodePipeline with CloudFormation et le AWS SAM pour fournir en continu vos applications sans serveur.

Vous pouvez ajouter une action à un pipeline utilisé AWS CloudFormation comme fournisseur de déploiement. Lorsque vous l'utilisez AWS CloudFormation en tant que fournisseur de déploiement, vous pouvez agir sur les CloudFormation piles et modifier les ensembles de modifications dans le cadre de l'exécution d'un pipeline. CloudFormation peut créer, mettre à jour, remplacer et supprimer des piles et modifier des ensembles lors de l'exécution d'un pipeline. Par conséquent, AWS les ressources personnalisées peuvent être créées, provisionnées, mises à jour ou supprimées lors de l'exécution d'un pipeline conformément aux spécifications que vous fournissez dans les CloudFormation modèles et les définitions de paramètres.

En savoir plus :

- CodePipeline Référence de configuration d'action pour CloudFormation référence d'action de déploiement
- Livraison continue avec CodePipeline — Découvrez comment l'utiliser pour CodePipeline créer un flux de travail de livraison continue pour CloudFormation.
- Automatisation du déploiement des Lambda-based applications — Apprenez à utiliser le modèle d'application AWS sans serveur et CloudFormation à créer un flux de travail de livraison continue pour votre Lambda-based application.

CloudFormation StackSets actions de déploiement

CloudFormationvous permet de déployer des ressources sur plusieurs comptes et AWS régions.

Vous pouvez utiliser CodePipeline with CloudFormation pour mettre à jour la définition de votre stack set et déployer des mises à jour sur vos instances.

Vous pouvez ajouter les actions suivantes à un pipeline pour l'utiliser AWS CloudFormation StackSets en tant que fournisseur de déploiement.

- CloudFormationStackSet
- CloudFormationStackInstances

En savoir plus :

- CodePipeline Référence de configuration d'action pour CloudFormation StackSets référence des actions de déploiement
- Tutoriel : Création d'un pipeline avec AWS CloudFormation StackSets actions de déploiement

Actions de déploiement d'Amazon ECS

Amazon ECS est un service de gestion de conteneurs hautement évolutif et performant qui vous permet d'exécuter des applications basées sur des conteneurs dans le. AWS Cloud Lorsque vous créez un pipeline, vous pouvez sélectionner Amazon ECS comme fournisseur de déploiement. Une modification du code dans votre référentiel de contrôle de source incite votre pipeline à créer une nouvelle image Docker, à la transférer dans votre registre de conteneurs, puis à déployer l'image mise à jour sur Amazon ECS. Vous pouvez également utiliser l'action du fournisseur ECS (Blue/Green) CodePipeline pour acheminer et déployer le trafic vers Amazon ECS avec CodeDeploy.

En savoir plus :

- Qu'est-ce qu'Amazon ECS ?
- Tutoriel : Déploiement continu avec CodePipeline
- Création d'un pipeline, d'étapes et d'actions
- Tutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR

Actions de déploiement d'Elastic Beanstalk

Elastic Beanstalk est un service de déploiement et de mise à l'échelle d'applications et de services Web développés avec Java, .NET, PHP, Python, Ruby Node.js, Go et Docker sur des serveurs courants tels qu'Apache, Nginx, Passenger et IIS. Vous pouvez configurer CodePipeline pour utiliser Elastic Beanstalk pour déployer votre code. Vous pouvez créer l'application et l'environnement Elastic Beanstalk à utiliser dans le cadre d'une action de déploiement au cours d'une étape, soit avant de créer le pipeline, soit lorsque vous utilisez l'assistant de création de pipeline.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans les régions Asie-Pacifique (Hyderabad), Asie-Pacifique (Melbourne), Moyen-Orient (Émirats arabes unis), Europe (Espagne) ou Europe (Zurich). Pour faire référence aux autres actions disponibles, voirIntégrations de produits et de services avec CodePipeline.

En savoir plus :

- Commencer à utiliser Elastic Beanstalk
- Création d'un pipeline, d'étapes et d'actions

OpsWorks actions de déploiement

OpsWorks est un service de gestion de configuration qui vous aide à configurer et à exploiter des applications de toutes formes et tailles à l'aide de Chef. Vous pouvez ainsi définir l'architecture de l'application et les spécifications de chaque composant, y compris l'installation du package, la configuration logicielle et les ressources telles que le stockage. AWS OpsWorks Stacks Vous pouvez le configurer CodePipeline pour AWS OpsWorks Stacks déployer votre code conjointement avec des livres de recettes et des applications Chef personnalisés dans OpsWorks.

- Livres de cuisine Chef personnalisés : OpsWorks utilise les livres de cuisine Chef pour gérer des tâches telles que l'installation et la configuration de packages et le déploiement d'applications.
- Applications : une OpsWorks application est composée de code que vous souhaitez exécuter sur un serveur d'applications. Le code de l'application est stocké dans un référentiel, tel qu'un compartiment Amazon S3.

Avant de créer le pipeline, vous devez créer la OpsWorks pile et la couche. Vous pouvez créer l' OpsWorks application à utiliser dans une action de déploiement au cours d'une étape, soit avant de créer le pipeline, soit lorsque vous utilisez l'assistant de création de pipeline.

CodePipeline le support pour OpsWorks n'est actuellement disponible que dans la région USA Est (Virginie du Nord) (us-east-1).

En savoir plus :

- Utilisation CodePipeline avec AWS OpsWorks Stacks
- Livres de recettes et recettes
- AWS OpsWorks Applications

Actions de déploiement du Service Catalog

Service Catalog permet aux entreprises de créer et de gérer des catalogues de produits dont l'utilisation est approuvée sur AWS.

Vous pouvez configurer CodePipeline pour déployer les mises à jour et les versions de vos modèles de produits sur Service Catalog. Vous pouvez créer le produit Service Catalog à utiliser dans le cadre d'une action de déploiement, puis utiliser l'assistant Create Pipeline pour créer le pipeline.

En savoir plus :

- Tutoriel : Création d'un pipeline qui se déploie vers Service Catalog
- Création d'un pipeline, d'étapes et d'actions

Actions de déploiement d'Amazon Alexa

Amazon Alexa Skills Kit vous permet de créer et de distribuer des compétences basées sur le cloud aux utilisateurs d' Alexa-enabled appareils.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans la région Asie-Pacifique (Hong Kong) ou en Europe (Milan). Pour utiliser les autres actions de déploiement disponibles dans cette région, consultezIntégrations d'actions de déploiement.

Vous pouvez ajouter une action à un pipeline qui utilise le kit Alexa Skills en tant que fournisseur de déploiement. Les modifications de source sont détectées par votre pipeline, qui déploie ensuite les mises à jour de votre compétence Alexa dans le service Alexa.

En savoir plus :

- Didacticiel : Création d'un pipeline qui déploie un kit Amazon Alexa Skill

CodeDeploy actions de déploiement

CodeDeploycoordonne les déploiements d'applications sur les EC2/on-premises instances Amazon, les plateformes de calcul Amazon Elastic Container Service et les plateformes de AWS Lambda calcul sans serveur. Vous pouvez le configurer CodePipeline pour l'utiliser CodeDeploy pour déployer votre code. Vous pouvez créer l'application, le déploiement et le groupe de déploiement CodeDeploy à utiliser dans une action de déploiement d'une étape avant la création du pipeline ou lorsque vous utilisez l'assistant Créer un pipeline.

En savoir plus :

- Étape 3 : Créez une application dans CodeDeploy
- Tutoriel : Création d'un pipeline simple (CodeCommit référentiel)

XebiaLabs actions de déploiement

Vous pouvez le configurer CodePipeline pour l'utiliser XebiaLabspour déployer votre code dans le cadre d'une ou de plusieurs actions d'un pipeline.

En savoir plus :

- Utilisation de XL Deploy avec CodePipeline

Intégration des actions d'approbation avec Amazon Simple Notification Service

Amazon SNS est un service de notification push rapide, flexible et entièrement géré qui vous permet d'envoyer des messages individuels ou de les distribuer à un grand nombre de destinataires. Amazon SNS permet d'envoyer des notifications push aux utilisateurs d'appareils mobiles, aux destinataires

d'e-mails de manière simple et rentable, ou même d'envoyer des messages à d'autres services distribués.

Lorsque vous créez une demande d'approbation manuelle dans CodePipeline, vous pouvez éventuellement publier sur une rubrique d'Amazon SNS afin que tous les utilisateurs IAM abonnés soient informés que l'action d'approbation est prête à être révisée.

En savoir plus :

- Qu'est-ce qu'Amazon SNS ?
- Accorder des autorisations Amazon SNS à un rôle CodePipeline de service

Intégrations d'actions d'appel

Les informations suivantes sont organisées par type CodePipeline d'action et peuvent vous aider CodePipeline à configurer l'intégration avec les fournisseurs d'actions d'appel suivants.

Rubriques

- Amazon Inspector appelle des actions
- Actions d'appel Lambda
- Step Functions invoque des actions

Amazon Inspector appelle des actions

Amazon Inspector est un service de gestion des vulnérabilités qui découvre automatiquement les charges de travail et les analyse en permanence pour détecter les vulnérabilités logicielles et les risques d'exposition involontaire au réseau. Amazon Inspector prend en charge plusieurs formats d'archive, notamment tar et war, et Amazon Inspector prend en charge les fichiers binaires tels que les binaires Rust et Go.

Vous pouvez configurer l' CodePipeline `InspectorScan`action pour automatiser l'analyse de votre code source ou du référentiel d'images Amazon ECR pour détecter les vulnérabilités.

En savoir plus :

- CodePipeline référence de configuration d'action pour Amazon Inspector `InspectorScan` invoque une référence d'action

Actions d'appel Lambda

Lambda vous permet d'exécuter du code sans provisionner ni gérer de serveurs. Vous pouvez configurer CodePipeline pour utiliser les fonctions Lambda afin d'ajouter de la flexibilité et des fonctionnalités à vos pipelines. Vous pouvez créer la fonction Lambda à ajouter en tant qu'action dans une étape soit avant de créer le pipeline, soit lorsque vous utilisez l'assistant de création de pipeline.

En savoir plus :

- CodePipeline Référence de configuration d'action pour AWS Lambda invoquer la référence de l'action
- Invoquer un AWS Lambda fonction dans un pipeline dans CodePipeline

Step Functions invoque des actions

Step Functions vous permet de créer et de configurer des machines d'état. Vous pouvez configurer CodePipeline pour utiliser les actions d'appel Step Functions afin de déclencher des exécutions par des machines à états.

En savoir plus :

- CodePipeline Référence de configuration d'action pour AWS Step Functions invoquer une référence d'action
- Tutoriel : Utiliser une AWS Step Functions action d'appel dans un pipeline

Intégrations générales avec CodePipeline

Les Service AWS intégrations suivantes ne sont pas basées sur des types CodePipeline d'action.

Amazon CloudWatch	Amazon CloudWatch surveille vos AWS ressources. En savoir plus : • Qu'est-ce qu'Amazon CloudWatch ?
Amazon EventBridge	Amazon EventBridge est un service Web qui détecte les modifications apportées à votre compte sur la Services AWS base de règles que

	vous définissez et qui invoque une action dans une ou plusieurs options spécifiées Services AWS lorsqu'une modification se produit. • Lancez automatiquement l'exécution d'un pipeline lorsque quelque chose change. Vous pouvez CodePipeline le configurer en tant que cible dans les règles définies dans Amazon EventBridge. Cette option permet de configurer le lancement automatique des pipelines lorsqu'un autre service change. En savoir plus : • Qu'est-ce qu'Amazon EventBridge ? • Démarrez un pipeline dans CodePipeline. • CodeCommit actions à la source et EventBridge • Recevoir des notifications lorsqu'un état de pipeline change — Vous pouvez définir des EventBridge règles pour détecter les changemen ts d'état d'exécution d'un pipeline, d'une étape ou d'une action et y réagir. En savoir plus : • Surveillance des CodePipeline événements • Tutoriel : Configuration d'une règle d' CloudWatch événements pour recevoir des notifications par e-mail en cas de modification de l'état du pipeline
AWS Cloud9	AWS Cloud9 est un IDE en ligne auquel vous pouvez accéder via votre navigateur Web. L'IDE offre une expérience d'édition de code enrichie : il prend en charge plusieurs langages de programmation et débogueur s d'exécution, et comporte un terminal intégré. En arrière-plan, une instance Amazon EC2 héberge un environnement de AWS Cloud9 développement. Pour plus d'informations, consultez le Guide de l'utilisa teur AWS Cloud9. En savoir plus : • Configuration d' AWS Cloud9

AWS CloudTrail	CloudTrailcapture les appels d' AWS API et les événements associés effectués par ou pour le compte d'un AWS compte et envoie les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez configurer CloudTrail pour capturer les appels d'API depuis la CodePipeline console, CodePipeline les commandes depuis et depuis l' CodePipeline API. AWS CLI En savoir plus : • Journalisation des appels d' CodePipeline API avec AWS CloudTrail
AWS CodeStar Notifications	Vous pouvez configurer des notifications pour informer les utilisateurs des modifications importantes, par exemple, lors du démarrage de l'exécution d'un pipeline. Pour plus d'informations, consultez Création d'une règle de notification.

AWS Key Management Service	AWS KMS est un service géré qui facilite la création et le contrôle des clés de chiffrement utilisées pour chiffrer vos données. Par défaut, permet CodePipeline AWS KMS de chiffrer les artefacts pour les pipelines stockés dans des compartiments Amazon S3. En savoir plus : • Pour créer un pipeline qui utilise un compartiment source, un compartiment d'artefacts et un rôle de service d'un AWS compte et des CodeDeploy ressources d'un autre AWS compte, vous devez créer une clé KMS gérée par le client, ajouter la clé au pipeline et configurer des politiques et des rôles de compte pour permettre l'accès entre comptes. Pour plus d'informations, consultez Créez un pipeline CodePipeline qui utilise les ressources d'un autre AWS compte. • Pour créer un pipeline à partir d'un AWS compte qui déploie une CloudFormation pile vers un autre AWS compte, vous devez créer une clé KMS gérée par le client, ajouter la clé au pipeline et configurer des politiques de compte et des rôles pour déployer la pile sur un autre compte. AWS Pour plus d'informations, consultez Comment utiliser CodePipeline pour déployer une AWS CloudFormation pile dans un autre compte ? • Pour configurer le chiffrement côté serveur pour le compartiment d'artefacts S3 de votre pipeline, vous pouvez utiliser la clé KMS AWS gérée par défaut ou créer une clé KMS gérée par le client et configurer la politique de compartiment pour utiliser la clé de chiffrement. Pour plus d'informations, consultez Configurer le chiffrement côté serveur pour les artefacts stockés dans Amazon S3 pour CodePipeline. Pour un AWS KMS key, vous pouvez utiliser l'ID de clé, l'ARN de la clé ou l'alias ARN.

> **Note**
>
> Les alias ne sont reconnus que dans le compte qui a créé la clé KMS. Pour les actions entre comptes, vous ne pouvez utiliser que l'ID de clé ou l'ARN de la clé pour identifier la clé. Cross-account les actions impliquent l'utilisation du rôle de l'autre compte (AccountB), donc la spécification de l'ID de clé utilisera la clé de l'autre compte (AccountB).

Exemples issus de la communauté

Les sections suivantes fournissent des liens vers des billets de blogs, des articles et des exemples fournis par la communauté.

> **Note**
>
> Ces liens sont fournis à titre informatif uniquement et ne doivent pas être considérés comme une liste exhaustive ni comme une approbation du contenu des exemples. AWS n'est pas responsable du contenu ou de l'exactitude du contenu externe.

Rubriques

- Exemples d'intégration : Articles de blog

Exemples d'intégration : Articles de blog

- Suivi de l'état de AWS CodePipeline construction depuis le référentiel Git tiers

  Apprenez à configurer des ressources qui afficheront votre pipeline et créeront le statut des actions dans votre référentiel tiers, afin de permettre au développeur de suivre facilement le statut sans changer de contexte.

  Publié en mars 2021

- CI/CD Complet avec AWS CodeCommitAWS CodeBuild, AWS CodeDeploy, et AWS CodePipeline

Découvrez comment configurer un pipeline qui utilise les CodeDeploy services CodeCommit, CodePipeline, et pour compiler CodeBuild, créer et installer une application Java dont la version est contrôlée sur un ensemble d'instances Linux Amazon EC2.

Publié en septembre 2020

- Tester et créer des CI/CD pipelines pour AWS Step Functions

  Apprenez à configurer les ressources qui coordonneront votre machine d'état Step Functions et votre pipeline.

  Publié en mars 2020

- Mettre en œuvre DevSecOps en utilisant CodePipeline

  Découvrez comment utiliser un CI/CD pipeline pour automatiser les contrôles CodePipeline de sécurité préventifs et de détection. Cet article explique comment utiliser un pipeline pour créer un groupe de sécurité simple et effectuer des contrôles de sécurité pendant les étapes de source, de test et de production afin d'améliorer le niveau de sécurité de vos AWS comptes.

  Publié en mars 2017

- Déploiement continu sur Amazon ECS à CodePipeline l' CodeBuildaide d'Amazon ECR et CloudFormation

  Découvrez comment créer un pipeline de déploiement continu pour Amazon Elastic Container Service (Amazon ECS). Les applications sont livrées sous forme de conteneurs Docker à l'aide de CodePipeline CodeBuild,, Amazon ECR et. CloudFormation

  - Téléchargez un exemple de CloudFormation modèle et des instructions pour l'utiliser pour créer votre propre pipeline de déploiement continu à partir du référentiel ECS Reference Architecture : Continuous Deployment on GitHub.

  Publié en janvier 2017

- Déploiement continu pour les applications sans serveur

  Découvrez comment utiliser une collection de Services AWS pour créer un pipeline de déploiement continu pour vos applications sans serveur. Vous utiliserez le modèle d'application sans serveur (SAM) pour définir l'application et ses ressources et CodePipeline pour orchestrer le déploiement de votre application.

- Consultez un exemple d'application écrit en Go avec l'infrastructure Gin et un shim de proxy API Gateway.

Publié en décembre 2016

- Mise à l'échelle DevOps des déploiements avec CodePipeline et Dynatrace

  Découvrez comment utiliser les solutions de surveillance Dynatrace pour dimensionner les pipelines CodePipeline, analyser automatiquement les exécutions de tests avant que le code ne soit validé et maintenir des délais optimaux.

Publié en novembre 2016

- Créez un pipeline pour AWS Elastic Beanstalk CodePipeline utiliser AWS CloudFormation et CodeCommit

  Découvrez comment implémenter la livraison continue dans un CodePipeline pipeline pour une application dans AWS Elastic Beanstalk. Toutes les AWS ressources sont mises en service automatiquement à l'aide d'un CloudFormation modèle. Cette procédure pas à pas intègre également CodeCommit et Gestion des identités et des accès AWS (IAM).

Publié en mai 2016

- Automatisez CodeCommit et CodePipeline entrez CloudFormation

  CloudFormation À utiliser pour automatiser le provisionnement des AWS ressources pour un pipeline de livraison continue qui utilise CodeCommit, CodePipeline CodeDeploy, et Gestion des identités et des accès AWS.

Publié en avril 2016

- Créez un Cross-Account pipeline dans AWS CodePipeline

  Découvrez comment automatiser le provisionnement de l'accès entre comptes aux pipelines dans AWS CodePipeline en utilisant Gestion des identités et des accès AWS. Inclut des exemples dans un CloudFormation modèle.

Publié en mars 2016

- Exploration ASP.NET de la partie essentielle 2 : livraison continue

  Découvrez comment créer un système de diffusion continue complet pour une application ASP.NET principale à l'aide de CodeDeploy et AWS CodePipeline.

Publié en mars 2016

- Création d'un pipeline à l'aide de la AWS CodePipeline console

  Apprenez à utiliser la AWS CodePipeline console pour créer un pipeline en deux étapes dans le cadre d'une procédure pas à pas basée sur le. AWS CodePipeline Didacticiel : Création d'un pipeline à quatre étapes

  Publié en mars 2016

- Mocking AWS CodePipeline Pipelines avec AWS Lambda

  Découvrez comment invoquer une fonction Lambda qui vous permet de visualiser les actions et les étapes d'un processus de livraison de CodePipeline logiciels au fur et à mesure de sa conception, avant que le pipeline ne soit opérationnel. Lorsque vous concevez la structure de votre pipeline, vous pouvez utiliser la fonction Lambda pour vérifier si votre pipeline se terminera correctement.

  Publié en février 2016

- Exécution de AWS Lambda fonctions lors de CodePipeline l'utilisation AWS CloudFormation

  Découvrez comment créer une AWS CloudFormation pile qui fournit toutes les AWS ressources utilisées dans le cadre de la tâche du guide de l'utilisateurInvoquer un AWS Lambda fonction dans un pipeline dans CodePipeline.

  Publié en février 2016

- Provisionnement d' CodePipeline actions personnalisées dans AWS CloudFormation

  Découvrez comment utiliser AWS CloudFormation pour configurer des actions personnalisées dans CodePipeline.

  Publié en janvier 2016

- Approvisionnement CodePipeline avec AWS CloudFormation

  Découvrez comment mettre en place un pipeline de livraison continue de base CodePipeline lors de l'utilisation AWS CloudFormation.

  Publié en décembre 2015

- Déploiement de CodePipeline et à OpsWorks l'aide d'une action personnalisée et AWS Lambda

Découvrez comment configurer votre pipeline et la AWS Lambda fonction à déployer pour AWS OpsWorks l'utiliser CodePipeline.

Date de publication : juillet2015

CodePipeline tutoriels

Une fois les étapes terminéesCommencer avec CodePipeline, vous pouvez essayer l'un des AWS CodePipeline didacticiels de ce guide de l'utilisateur.

Rubriques

- Tutoriel : Déploiement sur des instances Amazon EC2 avec CodePipeline
- Tutoriel : Créez et transférez une image Docker vers Amazon ECR avec CodePipeline (type V2)
- Tutoriel : Déploiement sur Amazon EKS avec CodePipeline
- Tutoriel : Création d'un pipeline qui exécute des commandes avec compute (type V2)
- Tutoriel : Utiliser les balises Git pour démarrer votre pipeline
- Tutoriel : Filtrer les noms de branches pour les pull requests afin de démarrer votre pipeline (type V2)
- Tutoriel : Utiliser des variables au niveau du pipeline
- Didacticiel : Création d'un pipeline simple (compartiment S3)
- Tutoriel : Création d'un pipeline simple (CodeCommit référentiel)
- Didacticiel : Création d'un pipeline à quatre étapes
- Tutoriel : Configuration d'une règle d' CloudWatch événements pour recevoir des notifications par e-mail en cas de modification de l'état du pipeline
- Tutoriel : Créez un pipeline qui crée et teste votre application Android avec AWS Device Farm
- Tutoriel : Créez un pipeline qui teste votre application iOS avec AWS Device Farm
- Tutoriel : Création d'un pipeline qui se déploie vers Service Catalog
- Tutoriel : Création d'un pipeline avec AWS CloudFormation
- Tutoriel : Création d'un pipeline qui utilise des variables provenant de AWS CloudFormation actions de déploiement
- Tutoriel : Déploiement standard d'Amazon ECS avec CodePipeline
- Tutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR
- Didacticiel : Création d'un pipeline qui déploie un kit Amazon Alexa Skill
- Tutoriel : Création d'un pipeline qui utilise Amazon S3 comme fournisseur de déploiement
- Tutoriel : Créez un pipeline qui publie votre application sans serveur sur AWS Serverless Application Repository

- Tutoriel : Déploiements de fonctions Lambda avec CodePipeline
- Tutoriel : Utilisation de variables avec des actions d'invocation Lambda
- Tutoriel : Utiliser une AWS Step Functions action d'appel dans un pipeline
- Tutoriel : Création d'un pipeline utilisé AWS AppConfig comme fournisseur de déploiement
- Tutoriel : Utiliser un clone complet avec une source de GitHub pipeline
- Tutoriel : Utiliser un clone complet avec une source de CodeCommit pipeline
- Tutoriel : Création d'un pipeline avec AWS CloudFormation StackSets actions de déploiement
- Tutoriel : Création d'une règle de vérification des variables pour un pipeline en tant que condition d'entrée

Tutoriel : Déploiement sur des instances Amazon EC2 avec CodePipeline

Ce didacticiel vous aide à créer une action de déploiement CodePipeline qui déploie votre code sur les instances que vous avez configurées dans Amazon EC2.

> **Note**
>
> Dans le cadre de la création d'un pipeline dans la console, un compartiment d'artefacts S3 sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

> **Note**
>
> L'action de `EC2` déploiement n'est disponible que pour les pipelines de type V2.

Conditions préalables

Avant de pouvoir créer votre pipeline de déploiement continu à l'aide de ce didacticiel, vous devez avoir mis en place certaines ressources. Voici ce dont vous avez besoin pour commencer :

> **Note**
>
> Toutes ces ressources devraient être créées au sein de la même AWS région.

- Un référentiel de contrôle de source (utilisé dans ce didacticiel GitHub) dans lequel vous allez ajouter un exemple de `script.sh` fichier.
- Vous devez utiliser un rôle de CodePipeline service existant qui a été mis à jour avec les autorisations nécessaires pour cette action. Pour mettre à jour votre rôle de service, consultezAutorisations relatives à la politique de rôle de service pour l'action de déploiement d'EC2.

Dès lors que ces prérequis sont respectés, vous pouvez commencer à créer votre pipeline de déploiement continu à l'aide du didacticiel.

Étape 1 : Création d'instances Amazon EC2 Linux

Au cours de cette étape, vous allez créer les instances Amazon EC2 dans lesquelles vous allez déployer un exemple d'application. Dans le cadre de ce processus, créez un rôle d'instance dans IAM, si vous n'avez pas encore créé de rôle d'instance dans la région où vous souhaitez créer des ressources.

Pour créer un rôle d'instance

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le tableau de bord de la console, choisissez Rôles.
3. Choisissez Créer un rôle.
4. Sous Sélectionner le type d'entité de confiance, sélectionnez Service AWS. Dans Choisir un cas d'utilisation, sélectionnez EC2. Sous Select your use case (Sélectionner votre cas d'utilisation), choisissez EC2. Choisissez Suivant.
5. Recherchez et sélectionnez la politique nommée **`AmazonSSMManagedEC2InstanceDefaultPolicy`**.
6. Recherchez et sélectionnez la politique nommée **`AmazonSSMManagedInstanceCore`**. Choisissez Suivant : Balises.
7. Choisissez Suivant : Vérification. Saisissez un nom pour le rôle (par exemple, **`EC2InstanceRole`**).

> **Note**
>
> Notez le nom de votre rôle pour l'étape suivante. Vous choisissez ce rôle lorsque vous créez votre instance.

> **Note**
>
> Vous allez ajouter des autorisations à ce rôle afin de permettre l'accès au compartiment d'artefacts S3 pour votre pipeline après sa création.

Choisissez Créer un rôle.

Pour lancer des instances

1. Ouvrez la console Amazon EC2 à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans la navigation latérale, choisissez Instances, puis sélectionnez Lancer des instances en haut de la page.
3. Pour Name (Nom), entrez **`MyInstances`**. Cela affecte à l'instance une clé de balise de **`Name`** et une valeur de balise de**`MyInstances`**.
4. Sous Images de l'application et du système d'exploitation (Amazon Machine Image), recherchez l'option Amazon Linux AMI avec le AWS logo et assurez-vous qu'elle est sélectionnée. (Cette AMI est décrite comme l'AMI Amazon Linux 2 (HVM) et est étiquetée « éligible au niveau gratuit ».)
5. Dans Type d'instance, choisissez le `t2.micro` type éligible au niveau gratuit comme configuration matérielle de votre instance.
6. Sous Paire de clés (connexion), choisissez une paire de clés ou créez-en une.
7. Dans Paramètres réseau, assurez-vous que l'état est Activé.
8. Développez Advanced Details (Détails avancés). Dans le profil d'instance IAM, choisissez le rôle IAM que vous avez créé lors de la procédure précédente (par exemple,**`EC2InstanceRole`**).

> **Note**
>
> Ne laissez pas le rôle d'instance vide car cela crée un rôle par défaut et ne sélectionne pas le rôle que vous avez créé.

9. Dans Résumé, sous Nombre d'instances, saisissez2.
10. Choisissez Launch instance (Lancer une instance).
11. Sur la page Instances, vous pouvez afficher le statut du lancement. Lorsque vous lancez une instance, son état initial est `pending`. Une fois que l'instance a démarré, son état devient `running` et elle reçoit un nom DNS public. (Si la colonne DNS public n'est pas affichée, choisissez l'Show/Hideicône, puis sélectionnez DNS public.)

Étape 2 : ajouter des autorisations de compartiment d'artefacts au rôle d'instance EC2

Vous devez mettre à jour le rôle d'instance EC2 que vous avez créé pour votre instance afin de lui permettre d'accéder au compartiment d'artefacts de votre pipeline.

> **Note**
>
> Lorsque vous créez l'instance, vous créez ou utilisez un rôle d'instance EC2 existant. Pour éviter les `Access Denied` erreurs, vous devez ajouter des autorisations de compartiment S3 au rôle d'instance afin de donner à l'instance des autorisations d'accès au compartiment d' CodePipeline artefacts. Créez un rôle par défaut ou mettez à jour votre rôle existant avec l'`s3:GetObject`autorisation limitée au compartiment d'artefacts pour la région de votre pipeline.

1. Accédez à votre pipeline dans la CodePipeline console. Cliquez sur Paramètres. Afficher le nom et l'emplacement du magasin d'artefacts d'un pipeline existant. Prenez note du bucket d'artefacts Amazon Resource Name (ARN) et copiez-le.
2. Accédez à la console IAM, puis choisissez Roles (Rôles). Choisissez le rôle d'instance que vous avez créé à l'étape 1 de ce didacticiel.
3. Sous l'onglet Autorisations, sélectionnez Ajouter une politique en ligne.

4. Ajoutez le code JSON suivant au document de politique, en remplaçant la valeur du `Resource` champ par l'ARN du bucket.

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

5. Choisissez Mettre à jour.

Étape 3 : ajouter un fichier de script à votre référentiel

Collez cet exemple de texte pour créer votre `script.sh` fichier pour l'étape post-script du déploiement.

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

Pour ajouter un **`script.sh`** fichier dans votre référentiel source

1. Ouvrez un éditeur de texte, puis copiez et collez le fichier ci-dessus dans un nouveau fichier.
2. Validez votre fichier `script.sh` et transmettez-le à votre référentiel source.

   a. Ajoutez le fichier.

   ```
   git add .
   ```

   b. Validez la modification.

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

   c. Transmettez la validation.

   ```
   git push
   ```

   Notez le chemin dans votre référentiel.

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

Étape 4 : Création de votre pipeline

Utilisez l' CodePipeline assistant pour créer les étapes de votre pipeline et connecter votre référentiel source.

Pour créer le pipeline

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyPipeline`**.
5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.
6. Dans Rôle de service, choisissez Utiliser le rôle de service existant, puis choisissez le rôle de CodePipeline service qui a été mis à jour avec les autorisations requises pour cette action. Pour configurer votre rôle CodePipeline de service pour cette action, consultezAutorisations relatives à la politique de rôle de service pour l'action de déploiement d'EC2.
7. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
8. Sur la page Étape 3 : Ajouter une étape source, ajoutez une étape source :

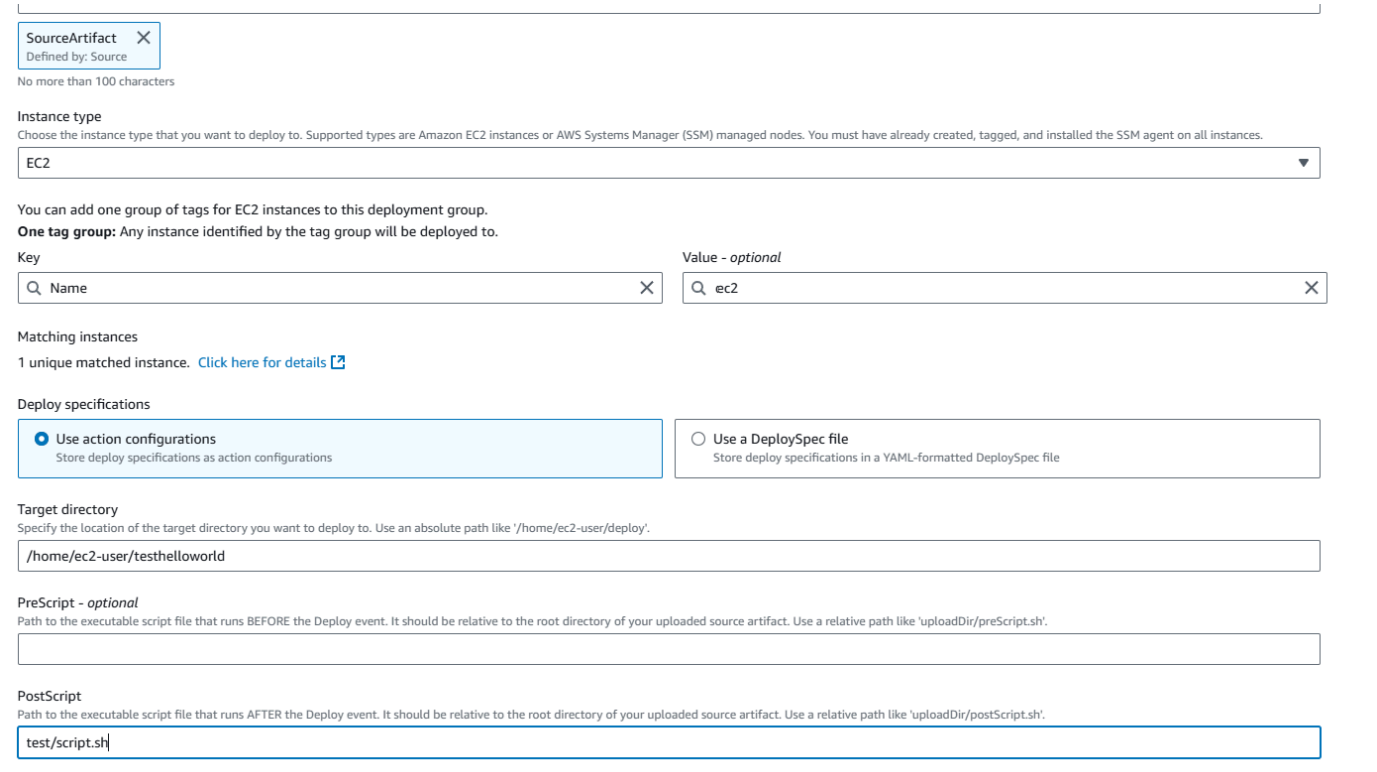
   a. Dans Source provider, choisissez GitHub (via GitHub App).

   b. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.

   c. Dans Nom du référentiel, choisissez le nom de votre référentiel GitHub .

   Choisissez Suivant.
9. Sur la page Étape 4 : Ajouter une étape de construction, choisissez Ignorer.

10. Sur la page Étape 5 : Ajouter une phase de déploiement, choisissez EC2.



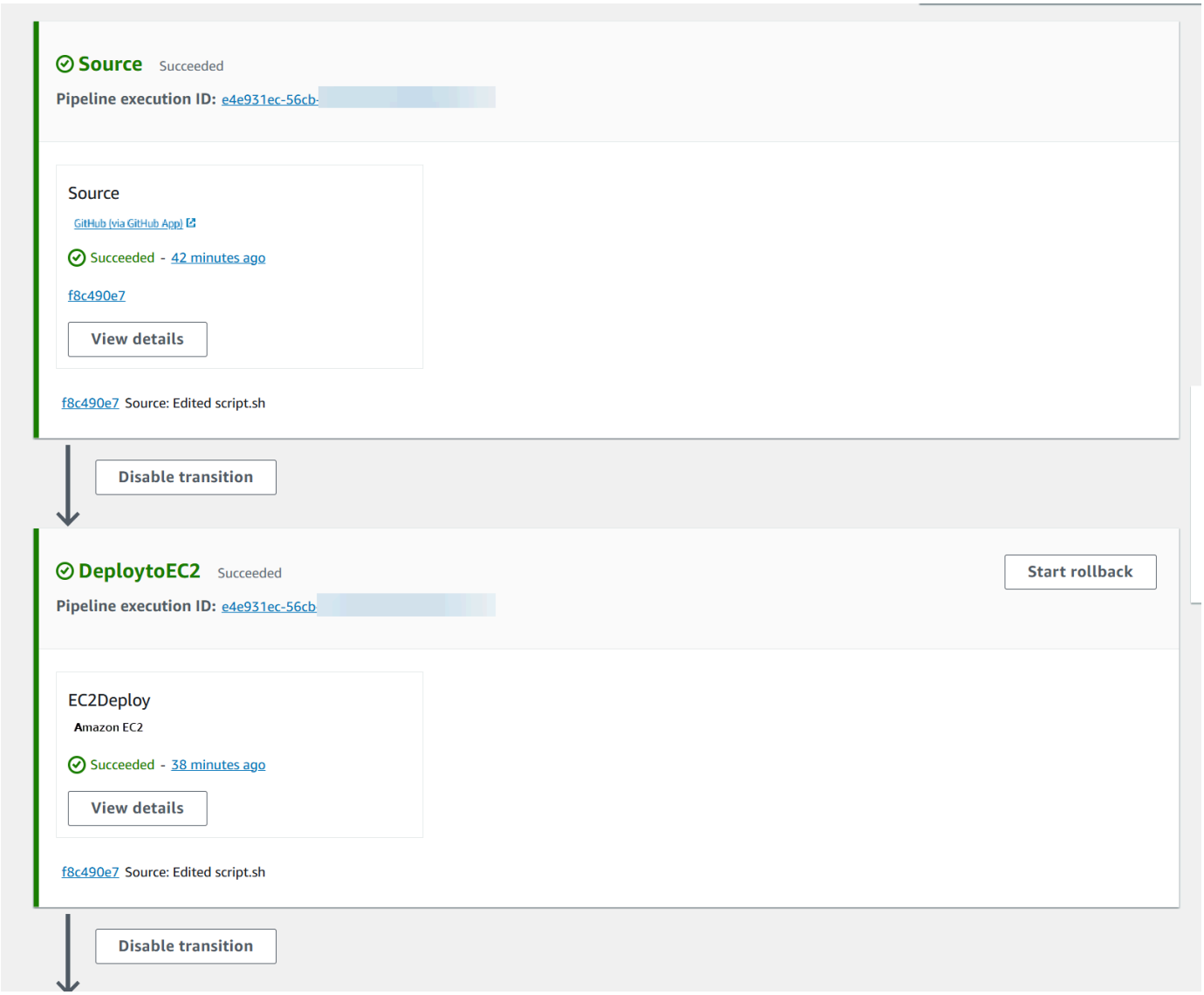
a. Pour le répertoire cible, entrez le répertoire de l'instance sur laquelle vous souhaitez effectuer le déploiement, par exemple`/home/ec2-user/testhelloworld`.

> **Note**
>
> Spécifiez le répertoire de déploiement que vous souhaitez que l'action utilise sur l'instance. L'action automatisera la création du répertoire spécifié sur l'instance dans le cadre du déploiement.

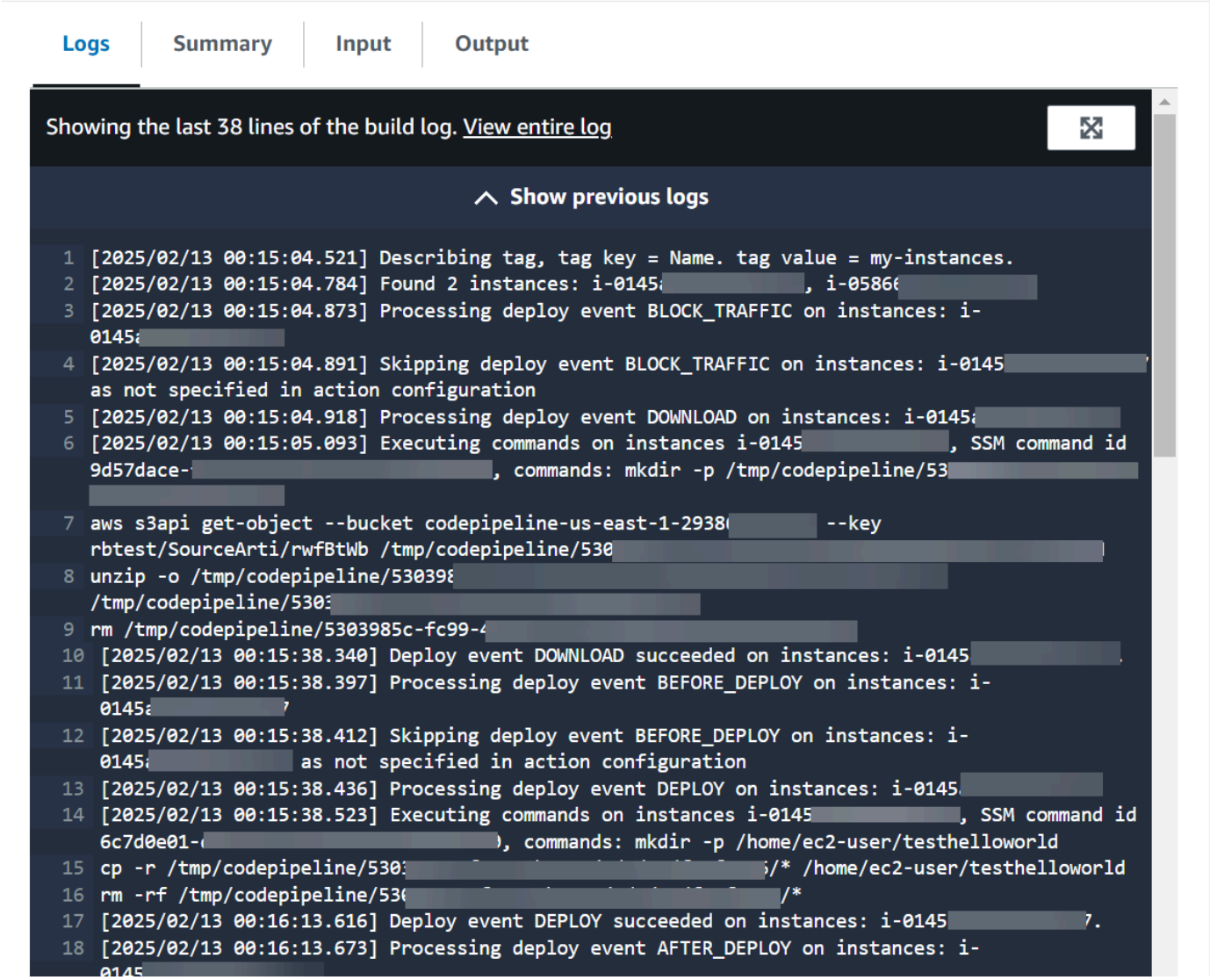
b. Pour PostScript, entrez le chemin et le nom de fichier de votre script, par exemple`test/script.sh`.

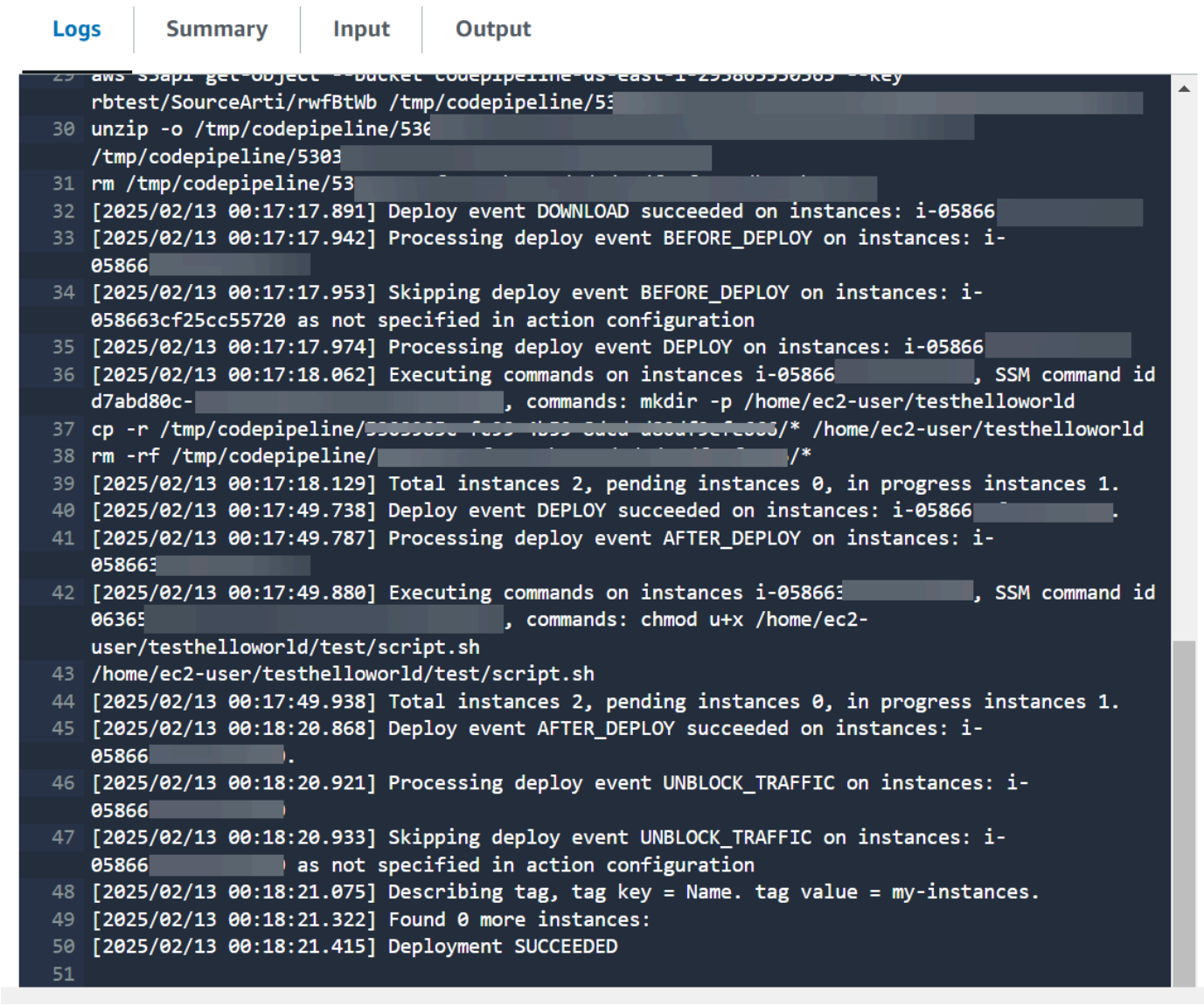
c. Choisissez Suivant.

11. Sur la page Étape 6 : Vérification, vérifiez la configuration de votre pipeline, puis choisissez Créer un pipeline pour créer le pipeline.



12. Une fois le pipeline exécuté avec succès, choisissez Afficher les détails pour afficher les journaux de l'action afin d'afficher le résultat de l'action de calcul gérée.


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



Étape 5 : Testez votre pipeline

Votre pipeline doit disposer de tous les éléments nécessaires pour exécuter un déploiement AWS continu natif de bout en bout. À présent, testez sa fonctionnalité en transmettant une modification de code à votre référentiel source.

Pour tester votre pipeline

1. Apportez une modification de code au référentiel source que vous avez configuré, validez-la, puis transmettez-la.
2. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.

3. Choisissez votre pipeline dans la liste.
4. Observez la progression du pipeline dans ses différentes phases. Votre pipeline devrait être terminé et votre action déploiera le script sur vos instances.
5. Pour plus d'informations sur la résolution des problèmes, consultezL'action de déploiement EC2 échoue avec un message d'erreur `Aucun fichier de ce type`.

Tutoriel : Créez et transférez une image Docker vers Amazon ECR avec CodePipeline (type V2)

Ce didacticiel vous aide à créer une action de génération CodePipeline qui s'exécute et transmet votre image Docker vers Amazon ECR après une modification de votre code source. Ce didacticiel explique également comment ajouter une action de déploiement Amazon ECS qui déploie votre image envoyée.

> **Important**
>
> Dans le cadre de la création d'un pipeline dans la console, un compartiment d'artefacts S3 sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

> **Note**
>
> Ce didacticiel concerne l'action de ECRBuildAndPublish création d'un CodePipeline pipeline avec un référentiel GitHub source et une action standard Amazon ECS pour le déploiement sur un cluster Amazon ECS. Pour un didacticiel utilisant un pipeline avec un référentiel d'images ECR comme source d'une action de CodeDeploy blue/green déploiement Amazon ECS CodePipeline, consultezTutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR.

> **Important**
>
> Cette action utilise le CodeBuild calcul CodePipeline géré pour exécuter des commandes dans un environnement de génération. L'exécution de l'action des commandes entraînera des frais distincts dans AWS CodeBuild.

Conditions préalables

Avant de pouvoir créer votre pipeline de déploiement continu à l'aide de ce didacticiel, vous devez avoir mis en place certaines ressources. Voici ce dont vous avez besoin pour commencer :

> **Note**
>
> Toutes ces ressources devraient être créées au sein de la même AWS région.

- Un référentiel de contrôle de source (utilisé dans ce didacticiel GitHub) dans lequel vous allez ajouter les éléments suivants pour ce didacticiel :
  - À l'étape 1, vous allez ajouter un exemple de Dockerfile à votre référentiel source en tant qu'artefact d'entrée pour l'action de ECRBuildAndPublish génération. CodePipeline
  - À l'étape 2, vous allez ajouter un exemple de fichier imagedefinitions.json à votre référentiel source comme condition préalable à l'action de déploiement standard d'Amazon ECS dans. CodePipeline
- Un référentiel d'images Amazon ECR qui contient une image que vous avez créée à partir de votre Dockerfile. Pour plus d'informations, consultez la section Création d'un référentiel et diffusion d'une image dans le guide de l'utilisateur d'Amazon Elastic Container Registry.
- Cluster et service Amazon ECS créés dans la même région que le référentiel d'images. Pour plus d'informations, consultez Création d'un cluster et création d'un service dans le guide du développeur Amazon Elastic Container Service.

Dès lors que ces prérequis sont respectés, vous pouvez commencer à créer votre pipeline de déploiement continu à l'aide du didacticiel.

Étape 1 : ajoutez un Dockerfile à votre référentiel source

Ce didacticiel utilise l' ECRBuildAndPublish action pour créer votre image Docker et la transmettre à Amazon ECR. L'action de calcul gérée est CodePipeline utilisée CodeBuild pour exécuter les commandes pour la connexion ECR et le push d'image. Il n'est pas nécessaire d'ajouter un `buildspec.yml` fichier à votre référentiel de code source pour indiquer CodeBuild comment procéder. Pour cet exemple, vous ne fournissez le Dockerfile dans votre référentiel que comme suit.

Collez cet exemple de texte pour créer votre `Dockerfile` fichier. Cet exemple de Dockerfile est identique à l'exemple utilisé dans les instructions relatives à l'image ECR dans les prérequis.

```
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
```

Pour ajouter un **`Dockerfile`** fichier dans votre référentiel source

1. Ouvrez un éditeur de texte, puis copiez et collez le Dockerfile ci-dessus dans un nouveau fichier.
2. Validez votre fichier `Dockerfile` et transmettez-le à votre référentiel source.

   a. Ajoutez le fichier.

   ```
   git add .
   ```

   b. Validez la modification.

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

c. Transmettez la validation.

```
git push
```

Veillez à placer le fichier à la racine de votre dépôt.

```
/ Dockerfile
```

Étape 2 : ajoutez un fichier imagedefinitions.json à votre référentiel source

Ce didacticiel utilise l'action de déploiement standard Amazon ECS CodePipeline pour déployer votre conteneur sur votre cluster Amazon ECS. L'action de déploiement standard d'Amazon ECS nécessite un fichier imagedefinitions.json contenant le nom et l'URI de votre image. Pour plus d'informations sur le fichier imagedefinitions.json, consultez. fichier imagedefinitions.json pour les actions de déploiement standard d'Amazon ECS

Collez cet exemple de texte pour créer votre `imagedefinitions.json` fichier. Utilisez le nom figurant dans votre Dockerfile, par exemple`hello-world`, et utilisez l'URI de votre référentiel Amazon ECR dans lequel l'image est stockée.

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

Pour ajouter un **`imagedefinitions.json`** fichier dans votre référentiel source

1. Ouvrez un éditeur de texte, puis copiez et collez l'exemple ci-dessus dans un nouveau fichier.
2. Validez votre fichier `imagedefinitions.json` et transmettez-le à votre référentiel source.

   a. Ajoutez le fichier.

```
git add .
```

b. Validez la modification.

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

c. Transmettez la validation.

```
git push
```

Veillez à placer le fichier à la racine de votre dépôt.

```
/ imagedefinitions.json
```

Étape 3 : Création de votre pipeline

Utilisez l' CodePipeline assistant pour créer les étapes de votre pipeline et connecter votre référentiel source.

Pour créer le pipeline

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**MyPipeline**.
5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.
6. Dans Rôle de service, choisissez Nouveau rôle de service pour permettre CodePipeline de créer un rôle de service dans IAM.
7. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
8. Sur la page Étape 3 : Ajouter une étape source, ajoutez une étape source :

a. Dans Source provider, choisissez GitHub (via GitHub App).

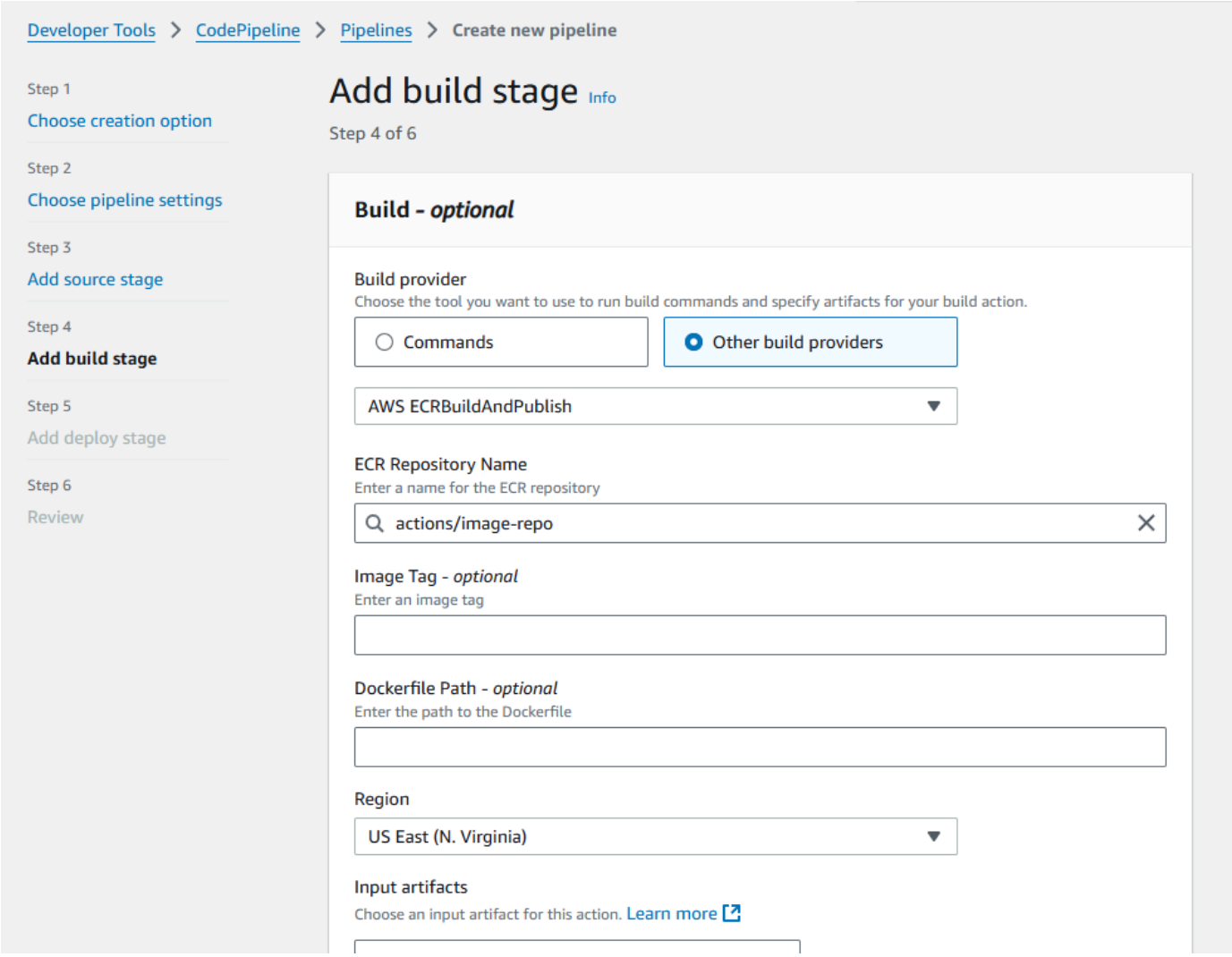
b. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.

c. Dans Nom du référentiel, choisissez le nom de votre référentiel GitHub .

d. Dans Branche par défaut, choisissez la branche que vous souhaitez spécifier lorsque le pipeline est démarré manuellement ou avec un événement source qui n'est pas une balise Git. Si la source de la modification n'est pas le déclencheur ou si l'exécution d'un pipeline a été démarrée manuellement, la modification utilisée sera le commit HEAD de la branche par défaut.

Choisissez Suivant.

9. Sur la page Étape 4 : Ajouter une étape de construction, choisissez Autres fournisseurs de build, puis choisissez ECRBuildAndPublish.



a. Pour le nom du référentiel ECR, choisissez votre référentiel d'images.

b. Choisissez Suivant.

10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.

11. Sur la page Étape 6 : Ajouter une phase de déploiement, choisissez Ignorer l'étape de déploiement. Vous allez ajouter l'action ECS à l'étape suivante.

12. Sur la page Étape 7 : Révision, passez en revue la configuration de votre pipeline et choisissez Créer un pipeline pour créer le pipeline.

13. Modifiez votre pipeline pour ajouter l'action de déploiement Amazon ECS à votre pipeline :

    a. Dans le coin supérieur droit, choisissez Modifier.

b. Au bas du diagramme, choisissez + Ajouter une étape. Dans le champ Nom de l'étape, saisissez un nom, tel que **`Deploy`**.

c. Choisissez + Ajouter un groupe d'actions.

d. Dans Nom de l'action, entrez un nom.

e. Dans Action provider, choisissez Amazon ECS. Acceptez la région du pipeline comme Région par défaut.

f. Dans Artefacts d'entrée, choisissez l'artefact d'entrée de l'étape source, par exemple`SourceArtifact`.

g. Pour Nom du cluster, choisissez le cluster Amazon ECS dans lequel votre service est exécuté.

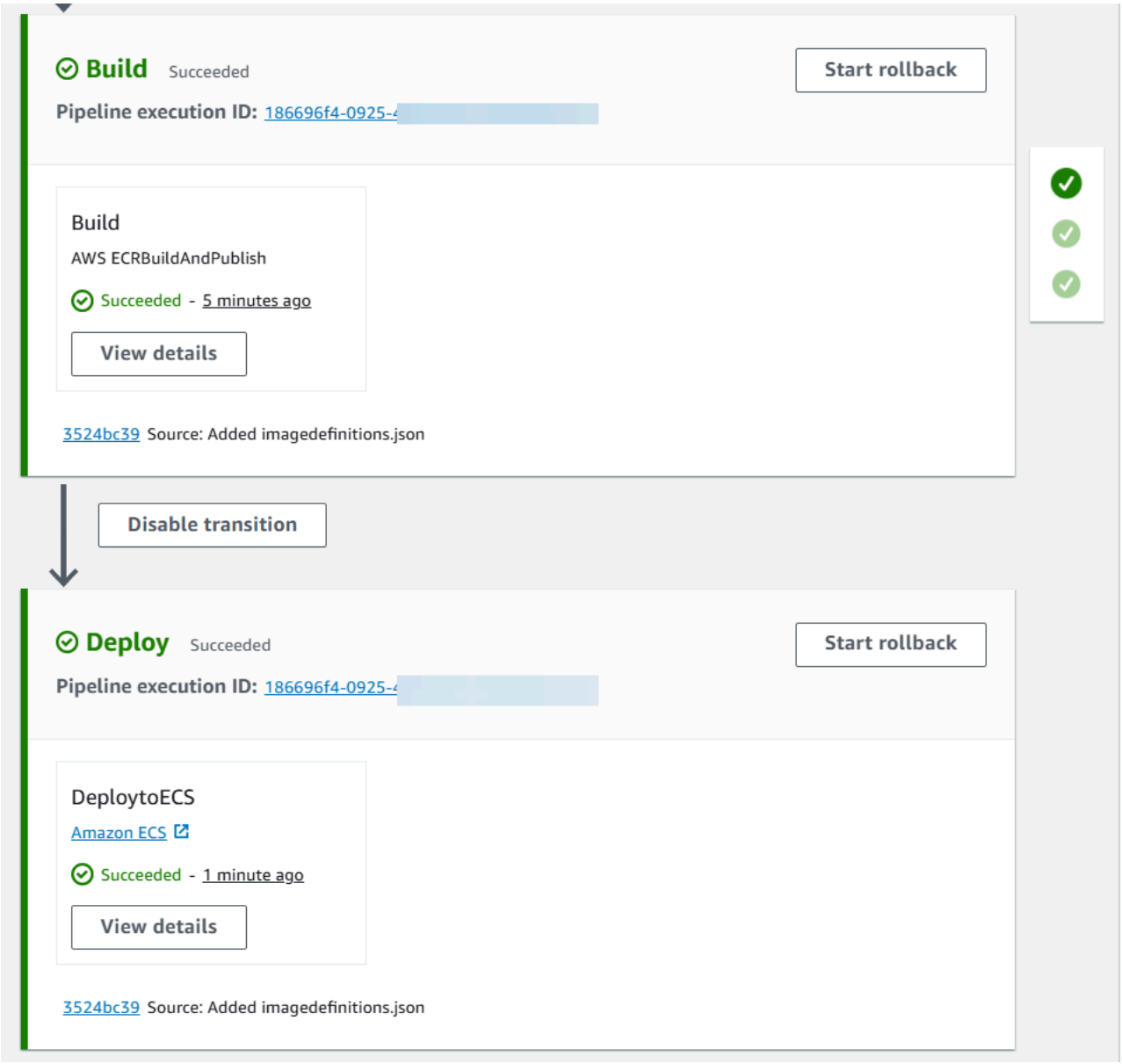
h. Dans Nom du service, choisissez le service à mettre à jour.

i. Choisissez Enregistrer.

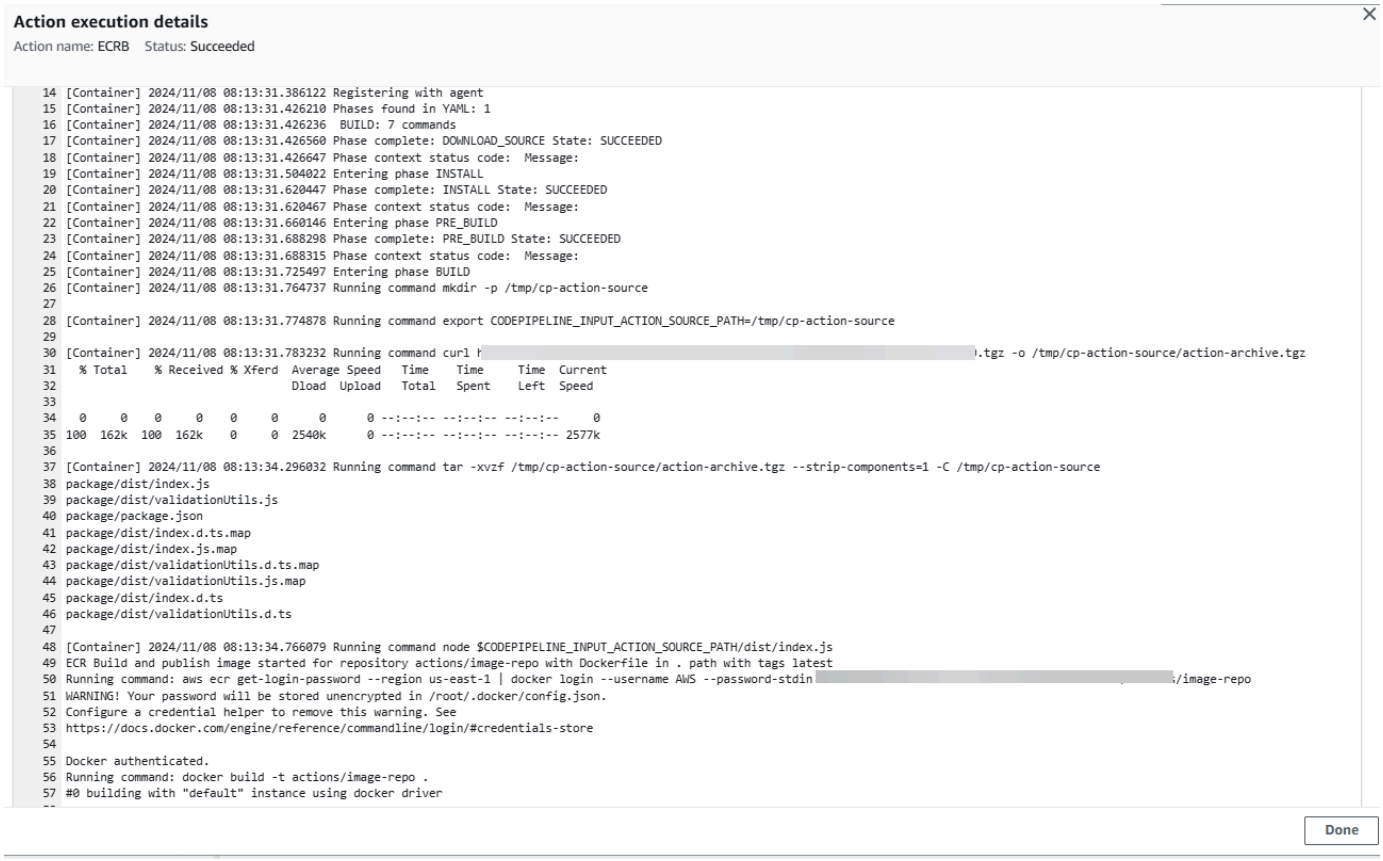
j. Dans l'étape que vous modifiez, choisissez Effectué. Dans le volet AWS CodePipeline , choisissez Enregistrer puis Enregistrer dans le message d'avertissement.

k. Pour soumettre vos modifications et lancer la génération d'un pipeline, choisissez Changement de version, puis Publication.

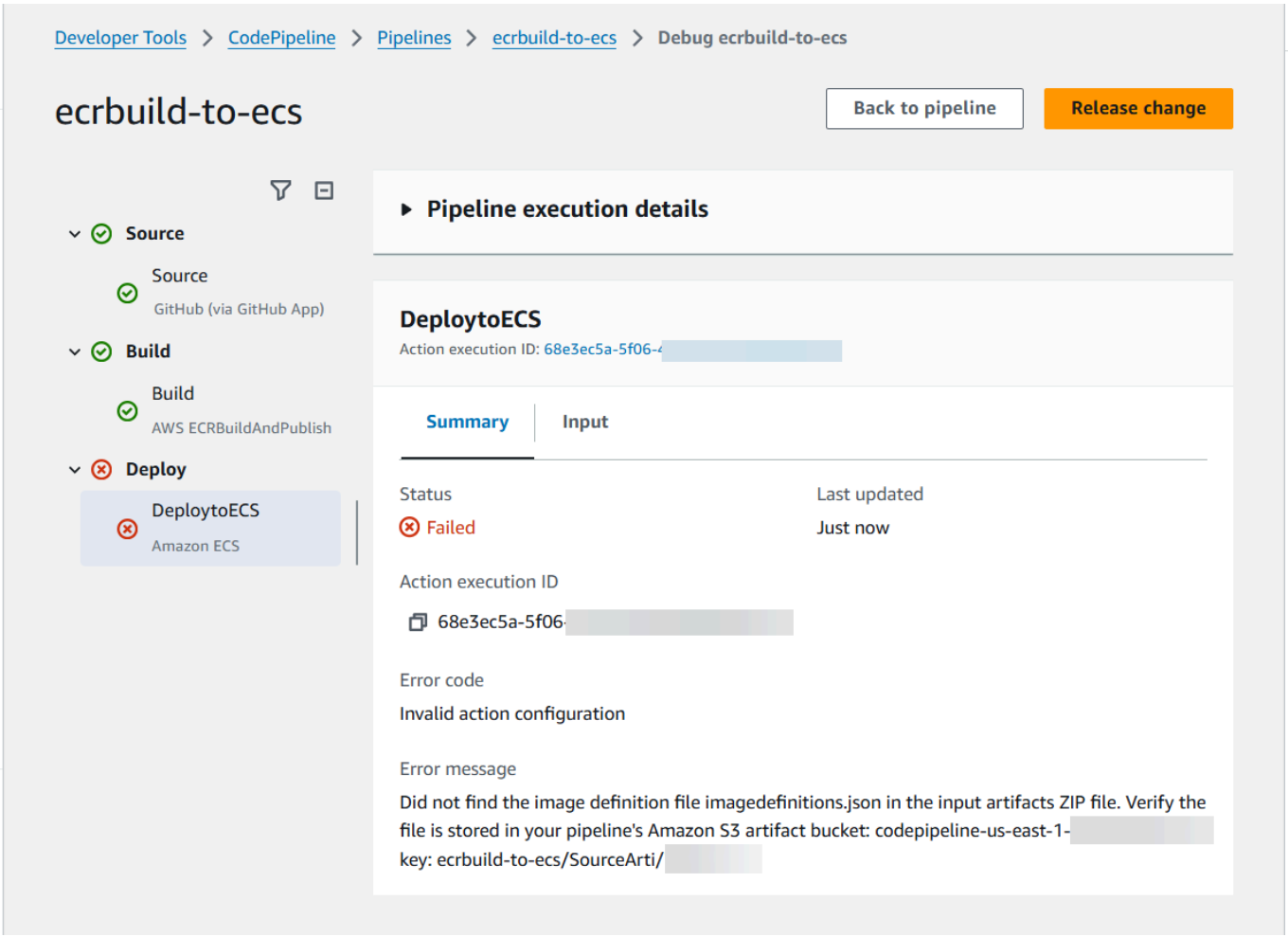
14. Une fois le pipeline exécuté, visualisez la structure et l'état du pipeline.



15. Une fois le pipeline exécuté avec succès, choisissez Afficher les détails pour afficher les journaux de l'action afin d'afficher le résultat de l'action de calcul gérée.



16. Résolvez les éventuelles actions qui ont échoué. Par exemple, l'action de déploiement d'ECS peut échouer si le fichier imagedefinitions.json ne se trouve pas dans le référentiel source. Voici un exemple du message d'erreur qui s'affiche lorsque le fichier imagedefinitions.json est manquant.



Étape 4 : Test de votre pipeline

Votre pipeline doit disposer de tous les éléments nécessaires pour exécuter un déploiement AWS continu natif de bout en bout. À présent, testez sa fonctionnalité en transmettant une modification de code à votre référentiel source.

Pour tester votre pipeline

1. Apportez une modification de code au référentiel source que vous avez configuré, validez-la, puis transmettez-la.
2. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
3. Choisissez votre pipeline dans la liste.

4. Observez la progression du pipeline dans ses différentes phases. Votre pipeline devrait être terminé et votre action redirige l'image Docker vers ECR qui a été créée à partir de votre modification de code.

Tutoriel : Déploiement sur Amazon EKS avec CodePipeline

Ce didacticiel vous aide à créer une action de déploiement CodePipeline qui déploie votre code sur un cluster que vous avez configuré dans Amazon EKS.

L'action EKS soutient les clusters EKS publics et privés. Les clusters privés sont le type recommandé par EKS ; toutefois, les deux types sont pris en charge.

> **Note**
>
> Dans le cadre de la création d'un pipeline dans la console, un compartiment d'artefacts S3 sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

> **Note**
>
> Cette action utilise le CodeBuild calcul CodePipeline géré pour exécuter des commandes dans un environnement de génération. L'exécution de l'action des commandes entraînera des frais distincts dans AWS CodeBuild.

> **Note**
>
> L'action de EKS déploiement n'est disponible que pour les pipelines de type V2.

Conditions préalables

Avant de pouvoir créer votre pipeline de déploiement continu à l'aide de ce didacticiel, vous devez avoir mis en place certaines ressources. Voici ce dont vous avez besoin pour commencer :

> **Note**
>
> Toutes ces ressources devraient être créées au sein de la même AWS région.

- Un référentiel de contrôle de source (utilisé dans ce didacticiel GitHub) dans lequel vous allez ajouter un exemple de `deployment.yaml` fichier.
- Vous devez utiliser un rôle de CodePipeline service existant que vous mettrez à jour avec les autorisations Étape 3 : mettre à jour la politique CodePipeline de rôle de service dans IAM ci-dessous pour cette action. Les autorisations nécessaires dépendent du type de cluster que vous créez. Pour de plus amples informations, veuillez consulter Autorisations relatives aux politiques de rôle de service.
- Une image de travail et une balise de référentiel que vous avez transmises à ECR ou à votre référentiel d'images.

Dès lors que ces prérequis sont respectés, vous pouvez commencer à créer votre pipeline de déploiement continu à l'aide du didacticiel.

Étape 1 : (Facultatif) Création d'un cluster dans Amazon EKS

Vous pouvez choisir de créer un cluster EKS avec un point de terminaison public ou privé.

Au cours des étapes suivantes, vous allez créer un cluster public ou privé dans EKS. Cette étape est facultative si vous avez déjà créé votre cluster.

Création d'un cluster public dans Amazon EKS

Au cours de cette étape, vous allez créer un cluster dans EKS.

Création d'un cluster public

1. Ouvrez la console EKS, puis choisissez Créer un cluster.
2. Dans Nom, donnez un nom à votre cluster. Choisissez Suivant.
3. Choisissez Créer.

Création d'un cluster privé dans Amazon EKS

Si vous choisissez de créer un cluster avec un point de terminaison privé, assurez-vous de connecter uniquement les sous-réseaux privés et assurez-vous qu'ils disposent d'une connexion Internet.

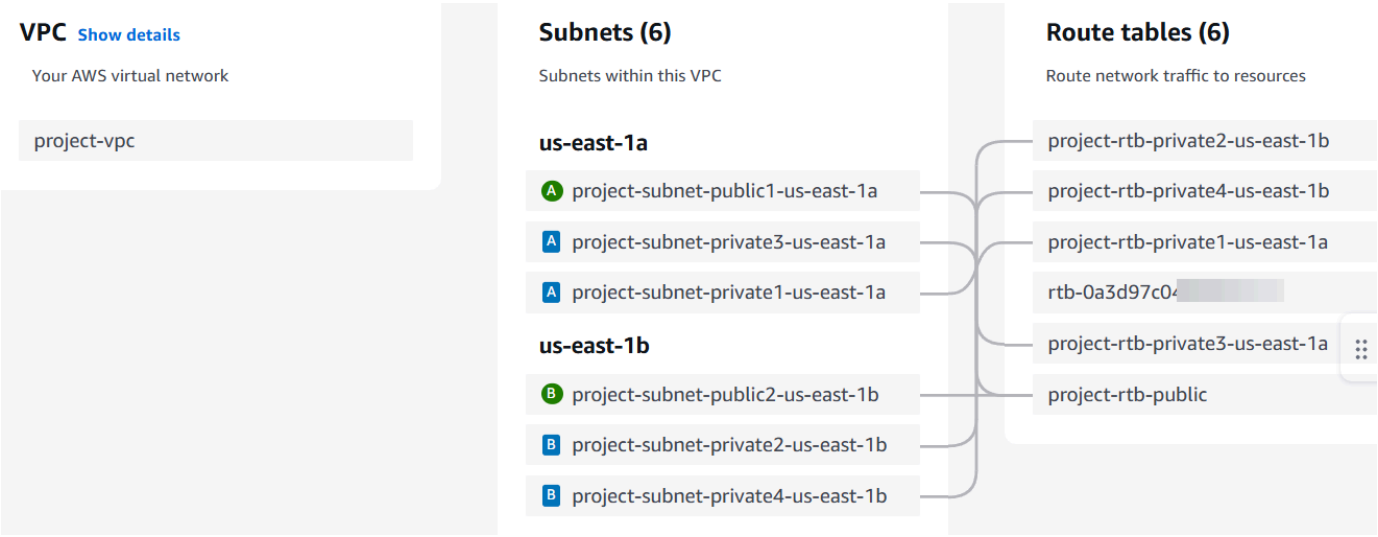
Suivez les cinq sous-étapes suivantes pour créer un cluster avec un point de terminaison privé.

Créez un VPC dans la console

1. Ouvrez la console VPC, puis choisissez Create VPC.
2. Sous Paramètres VPC, choisissez VPC et plus encore.
3. Choisissez de créer un sous-réseau public et quatre sous-réseaux privés. Sélectionnez Create VPC (Créer un VPC).
4. Sur la page des sous-réseaux, choisissez Privé.

Déterminez les sous-réseaux privés de votre VPC

1. Accédez à votre VPC et choisissez l'ID du VPC pour ouvrir la page de détails du VPC.
2. Sur la page des détails du VPC, choisissez l'onglet Resource map.
3. Consultez le schéma et notez vos sous-réseaux privés. Les sous-réseaux s'affichent avec des étiquettes indiquant leur statut public ou privé, et chaque sous-réseau est mappé à une table de routage.



Notez qu'un cluster privé contiendra tous les sous-réseaux privés.

4. Créez un sous-réseau public pour héberger la passerelle NAT. Vous ne pouvez attacher qu'une seule passerelle Internet à un VPC à la fois.

Création d'une passerelle NAT dans le sous-réseau public

1. Dans le sous-réseau public, créez une passerelle NAT. Accédez à la console VPC, puis choisissez Passerelles Internet. Choisissez Créer une passerelle Internet.
2. Dans Nom, entrez le nom de votre passerelle Internet. Choisissez Créer une passerelle Internet.

Mettez à jour la table de routage du sous-réseau privé afin de diriger le trafic vers la passerelle NAT.

Ajoutez la passerelle NAT à vos tables de routage pour vos sous-réseaux privés

1. Accédez à la console VPC, puis choisissez Subnets.
2. Pour chaque sous-réseau privé, choisissez-le, puis choisissez la table de routage pour ce sous-réseau sur la page de détails. Choisissez Modifier la table de routage.
3. Mettez à jour la table de routage du sous-réseau privé afin de diriger le trafic Internet vers la passerelle NAT. Choisissez Ajouter une route. Choisissez la passerelle NAT parmi les options à ajouter. Choisissez la passerelle Internet que vous avez créée.
4. Pour le sous-réseau public, créez une table de routage personnalisée. Vérifiez que la liste de contrôle d'accès réseau (ACL) de votre sous-réseau public autorise le trafic entrant en provenance du sous-réseau privé.
5. Sélectionnez Enregistrer les modifications.

Au cours de cette étape, vous allez créer un cluster dans EKS.

Création d'un cluster privé

1. Ouvrez la console EKS, puis choisissez Créer un cluster.
2. Dans Nom, donnez un nom à votre cluster. Choisissez Suivant.
3. Spécifiez votre VPC et d'autres informations de configuration. Choisissez Créer.

Votre cluster EKS peut être public ou privé. Cette étape concerne les clusters qui ont UNIQUEMENT un point de terminaison privé. Assurez-vous que votre cluster est privé.

Étape 2 : configurer votre cluster privé dans Amazon EKS

Cette étape n'est applicable que si vous avez créé un cluster privé. Cette étape concerne les clusters qui ont UNIQUEMENT un point de terminaison privé.

Configurez votre cluster

1. Attachez des sous-réseaux privés uniquement dans le cluster EKS sous l'onglet Réseau. Joignez les sous-réseaux privés capturés dans la section Déterminez les sous-réseaux privés de votre VPC ci-dessous. Étape 1 : (Facultatif) Création d'un cluster dans Amazon EKS
2. Assurez-vous que les sous-réseaux privés ont accès à Internet car ils CodePipeline stockent et récupèrent les artefacts du compartiment d'artefacts S3 pour votre pipeline.

Étape 3 : mettre à jour la politique CodePipeline de rôle de service dans IAM

Au cours de cette étape, vous allez mettre à jour un rôle de CodePipeline service existant**`cp-service-role`**, par exemple avec les autorisations requises CodePipeline pour vous connecter à votre cluster. Si vous n'avez pas de rôle existant, créez-en un nouveau.

Mettez à jour votre rôle de CodePipeline service en suivant les étapes suivantes.

Pour mettre à jour votre politique relative CodePipeline aux rôles de service

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le tableau de bord de la console, choisissez Rôles.
3. Recherchez votre rôle CodePipeline de service, par exemple**`cp-service-role`**.
4. Ajoutez une nouvelle politique en ligne.
5. Dans l'éditeur de politiques, saisissez les informations suivantes.

   - Pour un cluster public, ajoutez les autorisations suivantes.

     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": [
                "*"
            ]
        }
    ]
}
```

- Pour un cluster privé, ajoutez les autorisations suivantes. Les clusters privés nécessiteront des autorisations supplémentaires pour votre VPC, le cas échéant.

  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. Choisissez Mettre à jour une politique.

Étape 4 : Création d'une entrée d'accès pour le rôle CodePipeline de service

Au cours de cette étape, vous créez une entrée d'accès sur votre cluster qui ajoutera le rôle de CodePipeline service que vous avez mis à jour à l'étape 3, ainsi qu'une politique d'accès géré.

1. Ouvrez la console EKS et accédez à votre cluster.
2. Choisissez l'onglet Access.
3. Sous Entrées d'accès IAM, choisissez Créer une entrée d'accès.
4. Dans l'ARN principal d'IAM, entrez le rôle que vous venez de mettre à jour pour l'action, par exemple**`cp-service-role`**. Choisissez Suivant.

5. Sur la page Étape 2 : Ajouter une politique d'accès, dans Nom de la politique, choisissez la politique d'accès gérée, telle que`AmazonEKSClusterAdminPolicy`. Choisissez Add policy (Ajouter la politique). Choisissez Suivant.

> **Note**
>
> Il s'agit de la politique utilisée par l' CodePipeline action pour parler à Kubernetes. Il est recommandé de limiter les autorisations de votre politique avec le minimum de privilèges plutôt que la politique administrative, associez plutôt une stratégie personnalisée.

6. Sur la page de révision, choisissez Créer.

Étape 5 : Créez un référentiel source et ajoutez les fichiers de **`configuration du graphique`** Helm

Au cours de cette étape, vous créez un fichier de configuration adapté à votre action (fichiers manifestes Kubernetes ou graphique Helm) et vous stockez le fichier de configuration dans votre référentiel source. Utilisez le fichier adapté à votre configuration. Pour plus d'informations, consultez https://kubernetes.io/docs/reference/kubectl/quick-reference/ ou https://helm.sh/docs/topics/charts/.

- Pour Kubernetes, utilisez un fichier manifeste.
- Pour Helm, utilisez un graphique Helm.

1. Créez ou utilisez un GitHub référentiel existant.
2. Créez une nouvelle structure dans votre référentiel pour vos fichiers de diagrammes Helm, comme indiqué dans l'exemple ci-dessous.

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

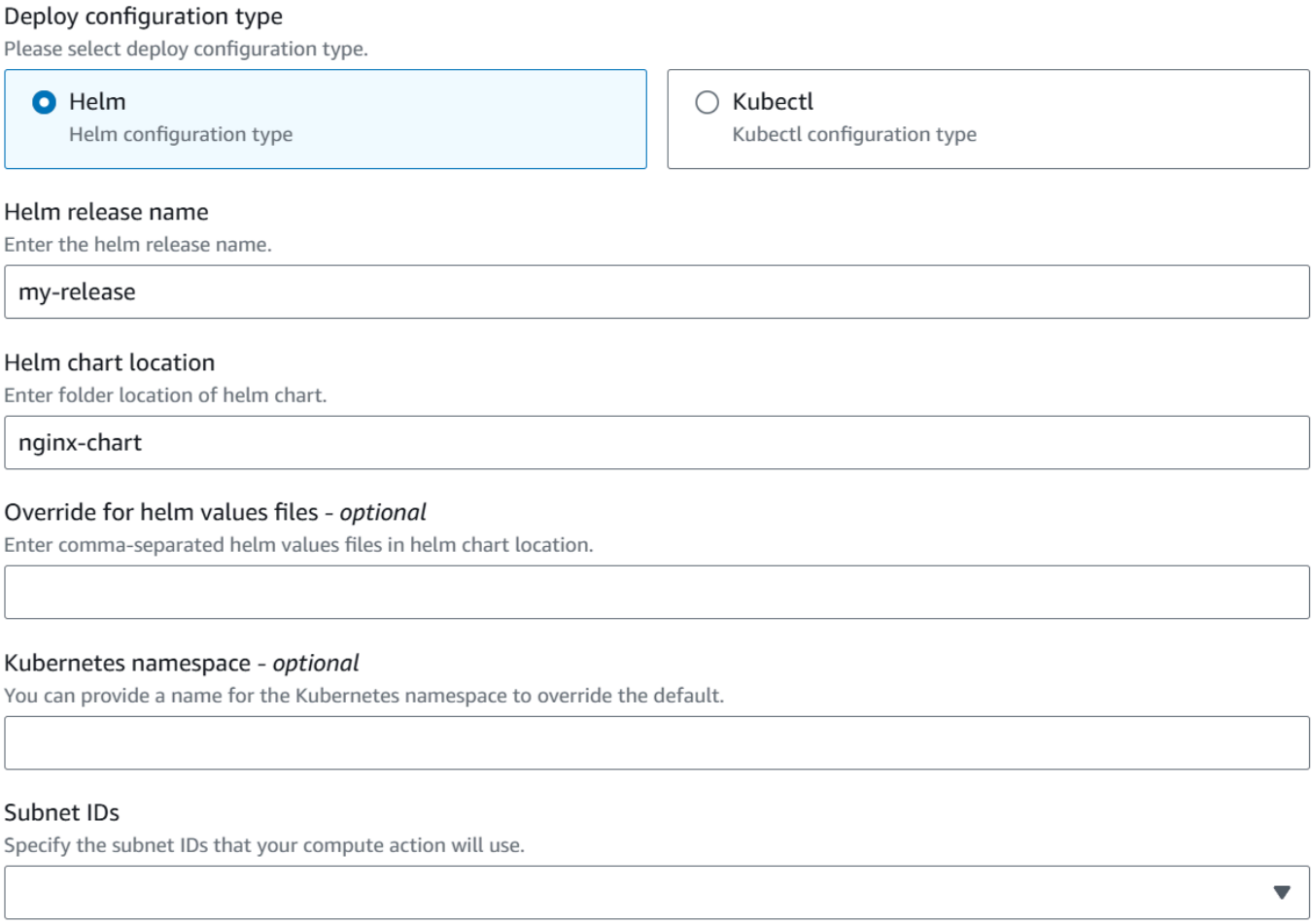
3. Ajoutez le fichier à la racine de votre référentiel.

Étape 6 : Création de votre pipeline

Utilisez l' CodePipeline assistant pour créer les étapes de votre pipeline et connecter votre référentiel source.

Pour créer le pipeline

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyEKSPipeline`**.
5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.
6. Dans Rôle de service, choisissez le rôle de service que vous avez mis à jour à l'étape 3.
7. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
8. Sur la page Étape 3 : Ajouter une étape source, pour Fournisseur source, choisissez de créer une connexion à votre GitHub référentiel.
9. Sur la page Étape 4 : Ajouter une étape de construction, choisissez Ignorer.
10. Sur la page Étape 5 : Ajouter une phase de déploiement, choisissez Amazon EKS.

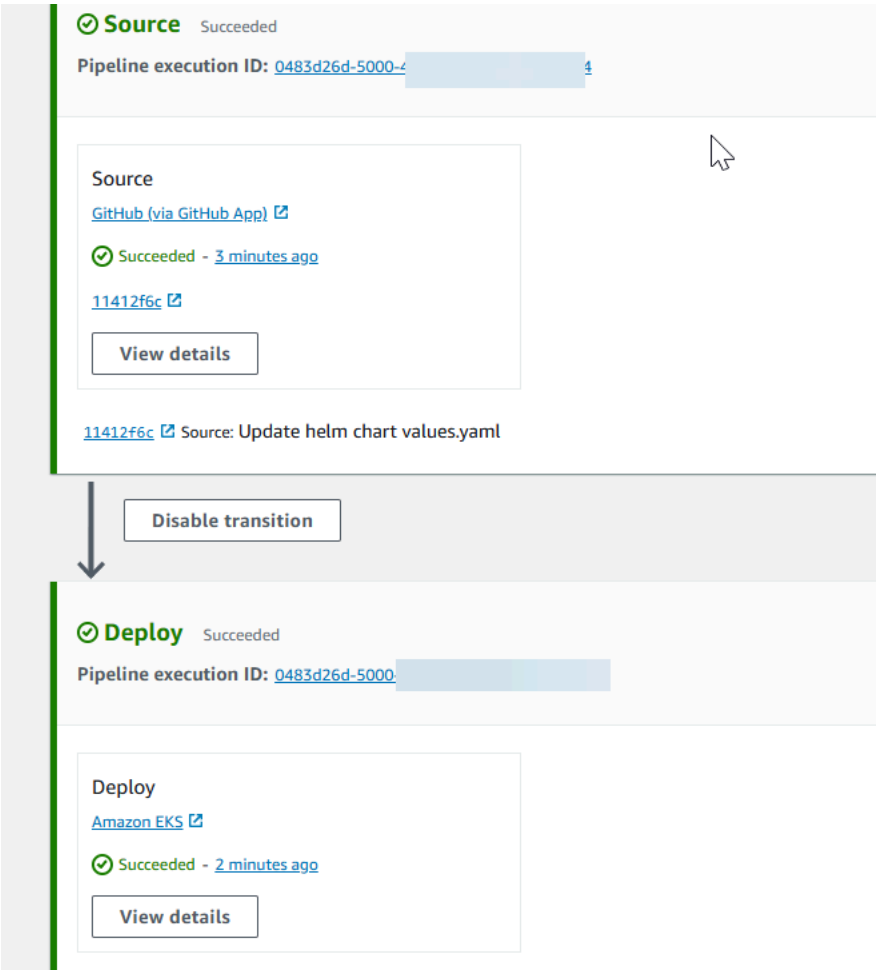


a. Dans Type de configuration de déploiement, choisissez Helm.

b. Dans Emplacement du graphique Helm, entrez le nom de la version, par exemple `my-release`. Pour l'emplacement de la carte Helm, entrez le chemin de vos fichiers de cartes Helm, par exemple `mychart`.

c. Choisissez Suivant.

11. Sur la page Étape 6 : Vérification, vérifiez la configuration de votre pipeline, puis choisissez Créer un pipeline pour créer le pipeline.



12. Une fois le pipeline exécuté avec succès, choisissez Afficher les détails pour afficher les journaux de l'action et voir le résultat de l'action.

Tutoriel : Création d'un pipeline qui exécute des commandes avec compute (type V2)

Dans ce didacticiel, vous allez configurer un pipeline qui exécute en continu les commandes de génération fournies à l'aide de l'action Commandes au cours d'une phase de génération. Pour plus d'informations sur l'action Commandes, consultezRéférence des actions des commandes.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un

compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Conditions préalables

Vous devez déjà disposer des éléments suivants :

- Un GitHub référentiel. Vous pouvez utiliser le GitHub référentiel que vous avez créé dansTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

Étape 1 : Créez des fichiers sources et transférez-les vers votre GitHub référentiel

Dans cette section, vous allez créer et envoyer vos exemples de fichiers source vers le référentiel que le pipeline utilise pour votre stage source. Pour cet exemple, vous produisez et envoyez les éléments suivants :

- Un fichier `README.txt`.

Pour créer des fichiers source

1. Créez un fichier contenant le texte suivant :

   ```
   Sample readme file
   ```

2. Enregistrez le fichier sous le nom `README.txt`.

Pour transférer des fichiers vers votre GitHub référentiel

1. Poussez ou chargez les fichiers dans votre référentiel . Ces fichiers constituent l'artefact source créé par l'assistant Create Pipeline (Création de pipeline) pour votre action de déploiement dans AWS CodePipeline. Vos fichiers doivent être similaires à ce qui suit dans votre répertoire local :

   ```
   README.txt
   ```

2. Pour utiliser la ligne de commande Git à partir d'un référentiel cloné sur votre ordinateur local :

a. Exécutez la commande suivante pour organiser tous vos fichiers à la fois :

```
git add -A
```

b. Exécutez la commande suivante pour valider les fichiers avec un message de validation.

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

c. Exécutez la commande suivante pour publier les fichiers depuis votre référentiel local vers votre référentiel :

```
git push
```

Étape 2 : Créer votre pipeline

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Une étape source avec une action GitHub (via GitHub App) pour le référentiel dans lequel les fichiers sources sont stockés.
- Une étape de construction avec l'action Commandes.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyCommandsPipeline`**.
5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.

6. Dans Rôle de service, choisissez Nouveau rôle de service pour permettre CodePipeline de créer un rôle de service dans IAM.

   > **Note**
   >
   > Si vous utilisez un rôle de service existant, pour utiliser l'action Commandes, vous devez ajouter les autorisations suivantes pour ce rôle de service. Réduisez les autorisations au niveau des ressources du pipeline en utilisant des autorisations basées sur les ressources dans la déclaration de politique des rôles de service. Pour plus d'informations, consultez l'exemple de politique dansAutorisations relatives aux politiques de rôle de service.
   >
   > - journaux : CreateLogGroup
   > - journaux : CreateLogStream
   > - journaux : PutLogEvents

7. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
8. Sur la page Étape 3 : Ajouter une étape source, ajoutez une étape source :

   a. Dans Source provider, choisissez GitHub (via GitHub App).

   b. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.

   c. Dans Nom du référentiel, choisissez le nom de votre référentiel GitHub.com .

   d. Dans Branche par défaut, choisissez la branche que vous souhaitez spécifier lorsque le pipeline est démarré manuellement ou avec un événement source qui n'est pas une balise Git. Si la source de la modification n'est pas le déclencheur ou si l'exécution d'un pipeline a été démarrée manuellement, la modification utilisée sera le commit HEAD de la branche par défaut. En option, vous pouvez également spécifier des webhooks avec filtrage (déclencheurs). Pour de plus amples informations, veuillez consulter Automatisez le démarrage des pipelines en utilisant des déclencheurs et des filtres.

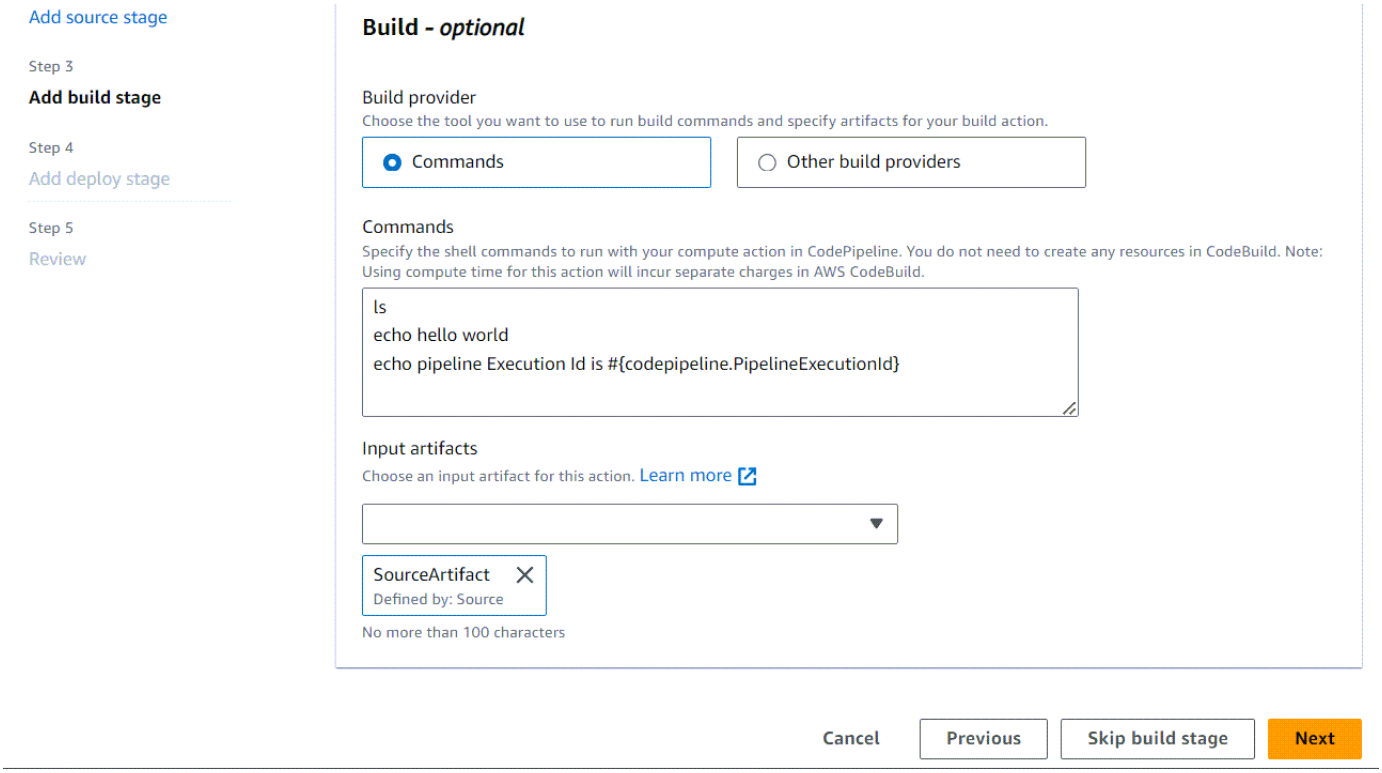
   Choisissez Suivant.
9. À l'étape 4 : Ajouter une étape de construction, choisissez Commandes.

> **Note**
>
> L'exécution de l'action Commandes entraînera des frais distincts dans AWS CodeBuild.

Entrez la commande suivante :

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

Choisissez Suivant.



10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.

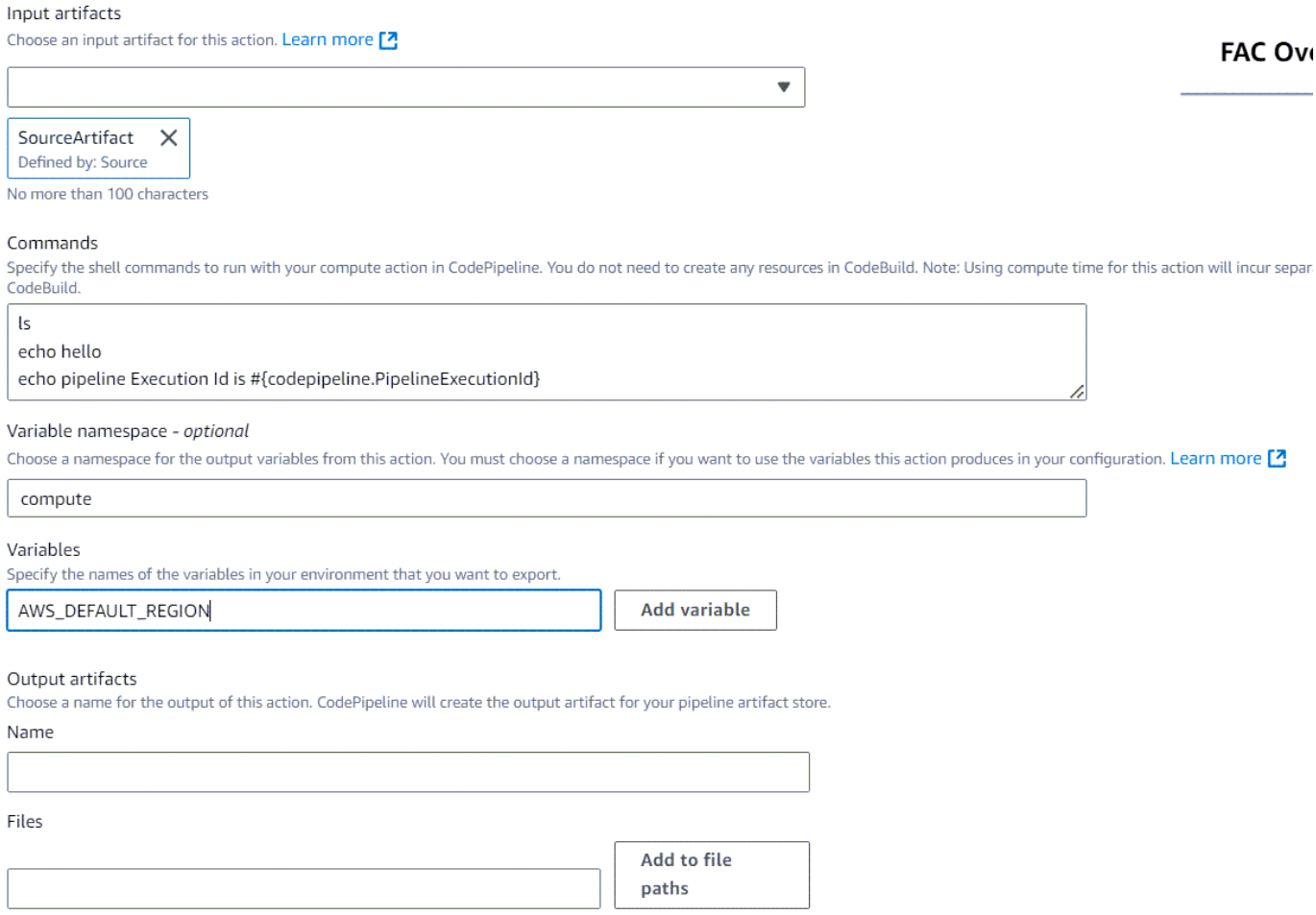
11. À l'étape 6 : Ajouter une phase de déploiement, choisissez Ignorer l'étape de déploiement, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

Choisissez Suivant.

12. À l'étape 7 : Passez en revue, passez en revue les informations, puis choisissez Créer un pipeline.

13. Pour terminer la création de votre action, ajoutez une variable d'environnement à l'action qui produira une variable de sortie pour l'action. Dans l'action Commandes, choisissez Modifier. Sur l'écran d'édition, spécifiez un espace de noms variable pour votre action en le saisissant `compute` dans le champ Espace de noms variable.

    Ajoutez la variable CodeBuild de sortie`AWS_Default_Region`, puis choisissez Ajouter une variable.



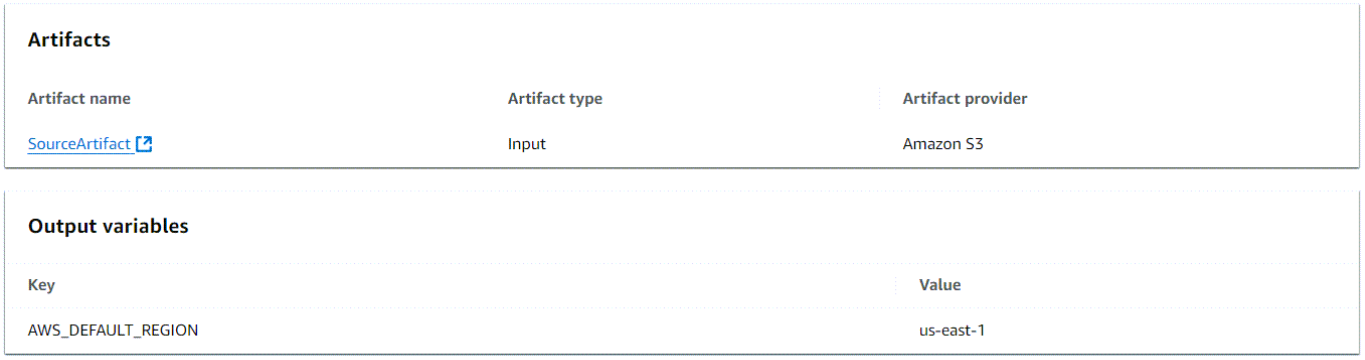
Étape 3 : Exécutez votre pipeline et vérifiez les commandes de construction

Publiez une modification pour exécuter votre pipeline. Vérifiez que les commandes de génération ont été exécutées en consultant l'historique des exécutions, les journaux de génération et les variables de sortie.

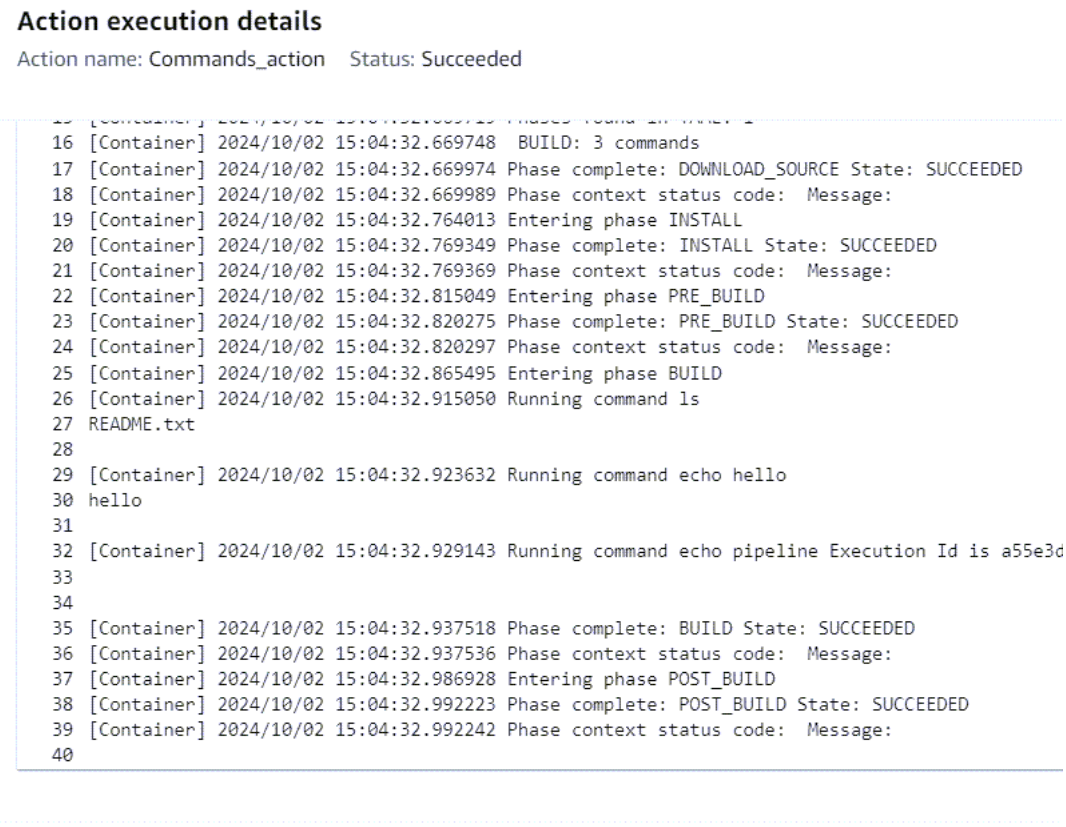
Pour afficher les journaux d'actions et les variables de sortie

1. Une fois le pipeline exécuté avec succès, vous pouvez consulter les journaux et les résultats de l'action.
2. Pour afficher les variables de sortie de l'action, choisissez Historique, puis Chronologie.

   Affichez la variable de sortie qui a été ajoutée à l'action. La sortie de l'action Commandes indique la variable de sortie résolue dans la région de l'action.



3. Pour afficher les journaux de l'action, choisissez Afficher les détails de l'action Commandes réussie. Consultez les journaux de l'action Commandes.



Tutoriel : Utiliser les balises Git pour démarrer votre pipeline

Dans ce didacticiel, vous allez créer un pipeline qui se connecte à votre GitHub référentiel dans lequel l'action source est configurée pour le type de déclencheur des balises Git. Lorsqu'une balise Git est créée lors d'un commit, votre pipeline démarre. Cet exemple montre comment créer un pipeline qui permet de filtrer les balises en fonction de la syntaxe du nom de la balise. Pour plus d'informations sur le filtrage à l'aide de modèles globulaires, consultezUtilisation de modèles globulaires dans la syntaxe.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Ce didacticiel se connecte GitHub via le type `CodeStarSourceConnection` d'action.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans les régions Asie-Pacifique (Hong Kong), Afrique (Le Cap), Moyen-Orient (Bahreïn) ou Europe (Zurich). Pour consulter les autres actions disponibles, consultezIntégrations de produits et de services avec CodePipeline. Pour les considérations relatives à cette action dans la région Europe (Milan), voir la note figurant dansCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

Rubriques

- Conditions préalables
- Étape 1 : Ouvrez CloudShell et clonez votre dépôt
- Étape 2 : Création d'un pipeline à déclencher sur les balises Git
- Étape 3 : Marquez vos commits en vue de leur publication
- Étape 4 : publier les modifications et afficher les journaux

Conditions préalables

Avant de commencer, vous devez exécuter les opérations suivantes :

- Créez un GitHub référentiel avec votre GitHub compte.
- Préparez vos GitHub informations d'identification. Lorsque vous utilisez le Console de gestion AWS pour établir une connexion, il vous est demandé de vous connecter avec vos GitHub informations d'identification.

Étape 1 : Ouvrez CloudShell et clonez votre dépôt

Vous pouvez utiliser une interface de ligne de commande pour cloner votre référentiel, effectuer des commits et ajouter des balises. Ce didacticiel lance une CloudShell instance pour l'interface de ligne de commande.

1. Connectez-vous au Console de gestion AWS.

2. Dans la barre de navigation supérieure, choisissez l' AWS icône. La page principale des Console de gestion AWS écrans.
3. Dans la barre de navigation supérieure, choisissez l' AWS CloudShell icône. CloudShell ouvre. Attendez que l' CloudShell environnement soit créé.

> **Note**
>
> Si vous ne voyez pas l' CloudShell icône, assurez-vous que vous vous trouvez dans une région prise en charge par CloudShell. Ce didacticiel part du principe que vous vous trouvez dans la région ouest des États-Unis (Oregon).

4. Dans GitHub, accédez à votre référentiel. Choisissez Code, puis HTTPS. Copiez le chemin. L'adresse pour cloner votre référentiel Git est copiée dans votre presse-papiers.
5. Exécutez la commande suivante pour cloner le référentiel.

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

6. Entrez votre GitHub compte `Username` et `Password` lorsque vous y êtes invité. Pour la `Password` saisie, vous devez utiliser un jeton créé par l'utilisateur plutôt que le mot de passe de votre compte.

Étape 2 : Création d'un pipeline à déclencher sur les balises Git

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Stage source avec connexion à votre GitHub référentiel et à votre action.
- Une étape de construction avec une action de AWS CodeBuild construction.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Bienvenue, Démarrez ou Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.

4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyGitHubTagsPipeline`**.
5. Dans Type de pipeline, conservez la sélection par défaut sur V2. Les types de pipelines diffèrent en termes de caractéristiques et de prix. Pour de plus amples informations, veuillez consulter Types de canalisations.
6. Dans Service role (Rôle de service), choisissez New service role (Nouveau rôle de service).

   > **Note**
   >
   > Si vous choisissez plutôt d'utiliser votre rôle de CodePipeline service existant, assurez-vous d'avoir ajouté l'autorisation `codestar-connections:UseConnection` IAM à votre politique de rôle de service. Pour obtenir des instructions relatives au rôle de CodePipeline service, voir Ajouter des autorisations au rôle CodePipeline de service.

7. Sous Paramètres avancés, conservez les valeurs par défaut. Dans le magasin d'artefacts choisissez Default location (Emplacement par défaut) pour utiliser le magasin d'artefacts par défaut, tel que le compartiment d'artefacts Amazon S3 désigné par défaut, pour votre pipeline dans la région que vous avez sélectionnée pour ce dernier.

   > **Note**
   >
   > Il ne s'agit pas du compartiment source de votre code source. Il s'agit du magasin d'artefacts pour votre pipeline. Un magasin d'artefacts distinct, tel qu'un compartiment S3, est nécessaire pour chaque pipeline.

   Choisissez Suivant.
8. Sur la page Étape 3 : Ajouter une étape source, ajoutez une étape source :
   a. Dans Source provider, choisissez GitHub (via GitHub App).
   b. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.
   c. Dans Nom du référentiel, choisissez le nom de votre référentiel GitHub.
   d. Dans Branche par défaut, choisissez la branche que vous souhaitez spécifier lorsque le pipeline est démarré manuellement ou avec un événement source qui n'est pas une balise Git. Si la source de la modification n'est pas le déclencheur ou si l'exécution d'un pipeline a

été démarrée manuellement, la modification utilisée sera le commit HEAD de la branche par défaut.

e. Sous Événements Webhook, dans Type de filtre, choisissez Tags.

Dans le champ Tags ou modèles, saisissez`release*`.

> **Important**
>
> Les pipelines qui commencent par un type de déclencheur de balises Git seront configurés pour les événements WebHookV2 et n'utiliseront pas l'événement Webhook (détection des modifications sur tous les événements push) pour démarrer le pipeline.

Choisissez Suivant.

9. Dans le champ Ajouter une étape de génération, ajoutez une étape de génération :

a. Dans le champ Fournisseur de génération, choisissez AWS CodeBuild. Acceptez la région du pipeline comme Région par défaut.

b. Sélectionnez Create a project (Créer un projet).

c. Dans Nom du projet, saisissez un nom pour ce projet de génération.

d. Dans le champ Image d'environnement, choisissez Image gérée. Pour Système d'exploitation, choisissez Ubuntu.

e. Pour Runtime (Exécution), sélectionnez Standard. Pour Image, choisissez aws/codebuild / standard:5.0.

f. Pour Rôle de service, choisissez Nouveau rôle de service.

> **Note**
>
> Notez le nom de votre rôle CodeBuild de service. Vous aurez besoin du nom du rôle pour la dernière étape de ce didacticiel.

g. Sous Buildspec, pour Build specifications (Spécifications de génération), choisissez Insert build commands (Insérer des commandes de génération). Choisissez Passer à l'éditeur et collez ce qui suit sous Commandes de création.

```
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. Choisissez Continuer vers CodePipeline. Cela revient à la CodePipeline console et crée un CodeBuild projet qui utilise vos commandes de build pour la configuration. Le projet de

génération utilise un rôle de service pour gérer les Service AWS autorisations. Cette étape peut prendre quelques minutes.

i. Choisissez Suivant.

10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.

11. Sur la page Étape 6 : Ajouter une phase de déploiement, choisissez Ignorer l'étape de déploiement, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer. Choisissez Suivant.

12. À l'étape 7 : Révision, choisissez Créer un pipeline.

Étape 3 : Marquez vos commits en vue de leur publication

Après avoir créé votre pipeline et spécifié des balises Git, vous pouvez baliser les commits dans votre GitHub référentiel. Au cours de ces étapes, vous allez baliser un commit avec le `release-1` tag. Chaque commit d'un dépôt Git doit avoir une balise Git unique. Lorsque vous choisissez le commit et que vous le balisez, cela vous permet d'intégrer les modifications provenant de différentes branches dans le déploiement de votre pipeline. Notez que le nom de la balise release ne s'applique pas au concept de version dans GitHub.

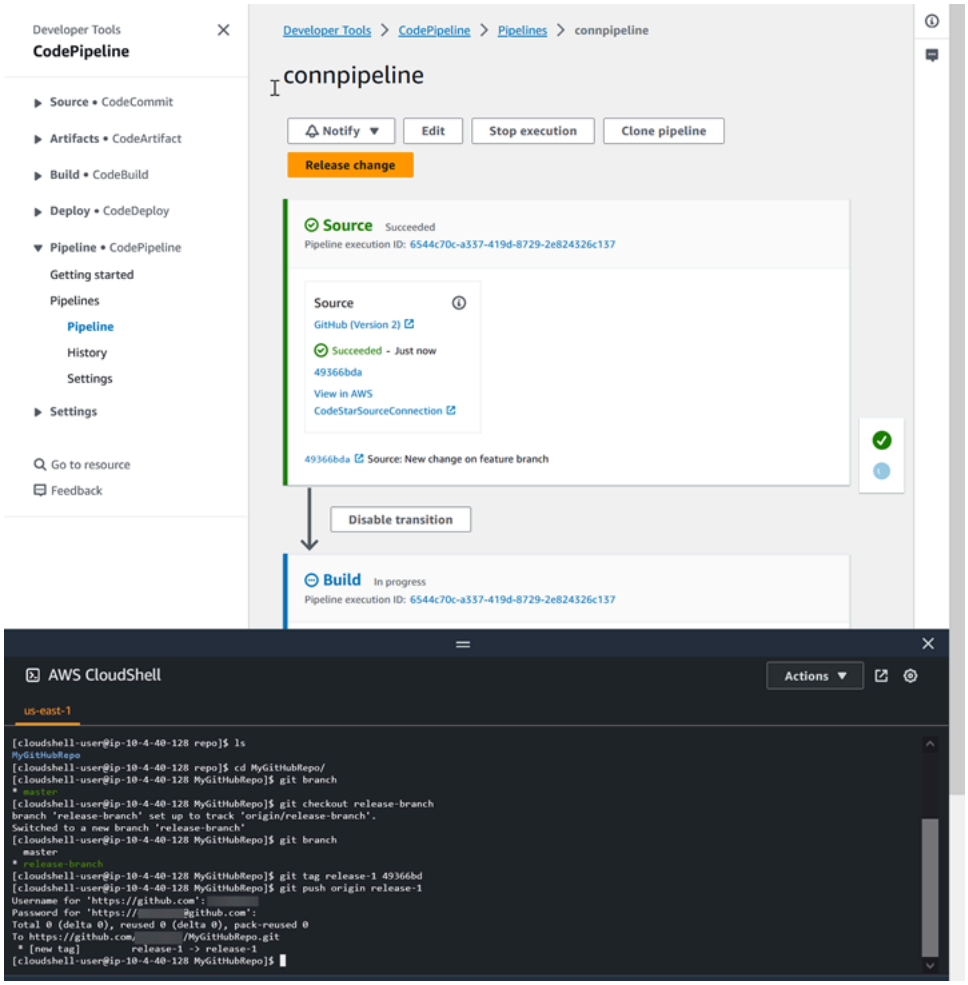
1. Faites référence aux ID de validation copiés que vous souhaitez baliser. Pour afficher les validations dans chaque branche, dans le CloudShell terminal, entrez la commande suivante pour capturer les ID de validation que vous souhaitez baliser :

   ```
   git log
   ```

2. Dans le CloudShell terminal, entrez la commande pour baliser votre commit et le transmettre à l'origine. Après avoir marqué votre commit, vous utilisez la commande git push pour envoyer le tag vers l'origine. Dans l'exemple suivant, entrez la commande suivante pour utiliser la `release-1` balise pour le deuxième commit avec ID49366bd. Cette balise sera filtrée par le filtre de `release*` balises du pipeline et démarrera le pipeline.

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

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



Étape 4 : publier les modifications et afficher les journaux

1. Une fois le pipeline exécuté avec succès, lors de la phase de création réussie, choisissez Afficher le journal.

   Sous Journaux, affichez le résultat de la CodeBuild compilation. Les commandes génèrent la valeur de la variable saisie.

2. Sur la page Historique, consultez la colonne Déclencheurs. Afficher le type de déclencheur GitTag : release-1.

Tutoriel : Filtrer les noms de branches pour les pull requests afin de démarrer votre pipeline (type V2)

Dans ce didacticiel, vous allez créer un pipeline qui se connecte à votre GitHub.com référentiel où l'action source est configurée pour démarrer votre pipeline avec une configuration de déclenchement qui filtre les demandes d'extraction. Lorsqu'un événement de demande d'extraction spécifié se produit pour une branche spécifiée, votre pipeline démarre. Cet exemple montre comment créer un pipeline qui permet de filtrer les noms de branches. Pour plus d'informations sur l'utilisation des déclencheurs, consultezAjouter des filtres pour les types d'événements de requêtes push et pull (CLI). Pour plus d'informations sur le filtrage à l'aide de modèles regex au format glob, consultez. Utilisation de modèles globulaires dans la syntaxe

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Ce didacticiel se connecte GitHub.com via le type `CodeStarSourceConnection` d'action.

Rubriques

- Conditions préalables
- Étape 1 : Création d'un pipeline à démarrer lors d'une demande d'extraction pour des branches spécifiées
- Étape 2 : Créez et fusionnez une pull request GitHub.com pour démarrer les exécutions de votre pipeline

Conditions préalables

Avant de commencer, vous devez exécuter les opérations suivantes :

- Créez un GitHub.com référentiel avec votre GitHub.com compte.

- Préparez vos GitHub informations d'identification. Lorsque vous utilisez le Console de gestion AWS pour établir une connexion, il vous est demandé de vous connecter avec vos GitHub informations d'identification.

Étape 1 : Création d'un pipeline à démarrer lors d'une demande d'extraction pour des branches spécifiées

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Stage source avec connexion à votre GitHub.com référentiel et à votre action.
- Une étape de construction avec une action de AWS CodeBuild construction.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Bienvenue, Démarrez ou Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyFilterBranchesPipeline`**.
5. Dans Type de pipeline, conservez la sélection par défaut sur V2. Les types de pipelines diffèrent en termes de caractéristiques et de prix. Pour de plus amples informations, veuillez consulter Types de canalisations.
6. Dans Service role (Rôle de service), choisissez New service role (Nouveau rôle de service).

   > **Note**
   >
   > Si vous choisissez plutôt d'utiliser votre rôle de CodePipeline service existant, assurez-vous d'avoir ajouté l'autorisation `codeconnections:UseConnection` IAM à votre politique de rôle de service. Pour obtenir des instructions relatives au rôle de CodePipeline service, voir Ajouter des autorisations au rôle CodePipeline de service.

7. Sous Paramètres avancés, conservez les valeurs par défaut. Dans le magasin d'artefacts choisissez Default location (Emplacement par défaut) pour utiliser le magasin d'artefacts par

défaut, tel que le compartiment d'artefacts Amazon S3 désigné par défaut, pour votre pipeline dans la région que vous avez sélectionnée pour ce dernier.

> **Note**
>
> Il ne s'agit pas du compartiment source de votre code source. Il s'agit du magasin d'artefacts pour votre pipeline. Un magasin d'artefacts distinct, tel qu'un compartiment S3, est nécessaire pour chaque pipeline.

Choisissez Suivant.

8. Sur la page Étape 3 : Ajouter une étape source, ajoutez une étape source :

   a. Dans Source provider, choisissez GitHub (via GitHub App).

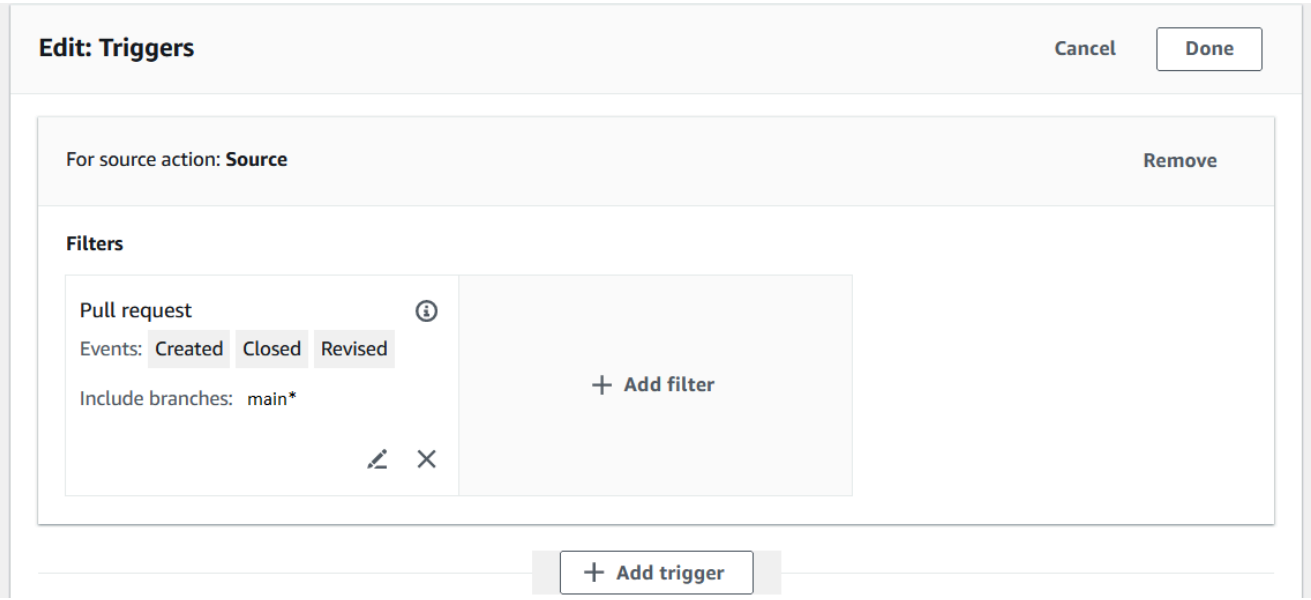
   b. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.

   c. Dans Nom du référentiel, choisissez le nom de votre référentiel GitHub.com.

   d. Sous Type de déclencheur, choisissez Spécifier le filtre.

      Sous Type d'événement, choisissez Pull request. Sélectionnez tous les événements sous demande d'extraction afin que l'événement se produise pour les demandes d'extraction créées, mises à jour ou fermées.

      Sous Branches, dans le champ Inclure, saisissez`main*`.



> **Important**
>
> Les pipelines qui commencent par ce type de déclencheur seront configurés pour les événements WebHookV2 et n'utiliseront pas l'événement Webhook (détection des modifications sur tous les événements push) pour démarrer le pipeline.

Choisissez Suivant.

9. À l'étape 4 : Ajouter une étape de construction, dans Build provider, choisissez AWS CodeBuild. Acceptez la région du pipeline comme Région par défaut. Choisissez ou créez le projet de construction comme indiqué dansTutoriel : Utiliser les balises Git pour démarrer votre pipeline. Cette action ne sera utilisée dans ce didacticiel que comme deuxième étape nécessaire à la création de votre pipeline.
10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.
11. Sur la page Étape 6 : Ajouter une phase de déploiement, choisissez Ignorer l'étape de déploiement, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer. Choisissez Suivant.
12. À l'étape 7 : Révision, choisissez Créer un pipeline.

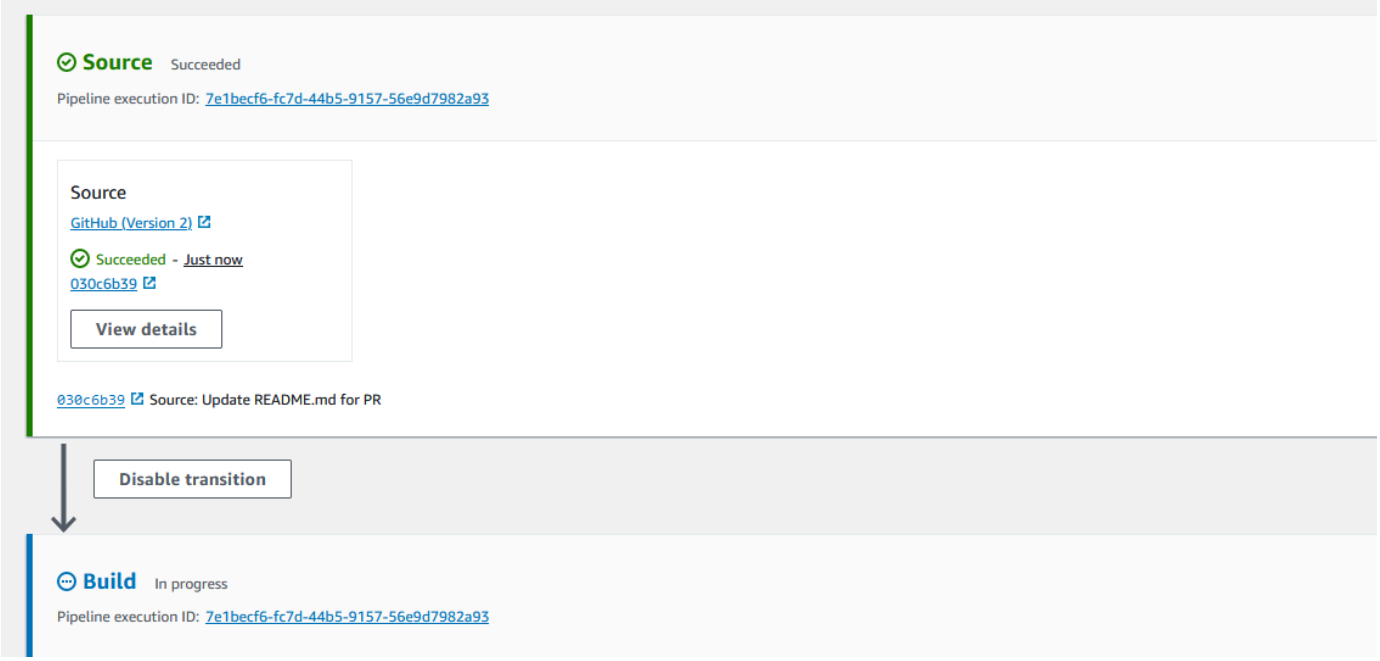
Étape 2 : Créez et fusionnez une pull request GitHub.com pour démarrer les exécutions de votre pipeline

Dans cette section, vous allez créer et fusionner une pull request. Cela démarre votre pipeline, avec une exécution pour la demande d'extraction ouverte et une exécution pour la demande d'extraction fermée.

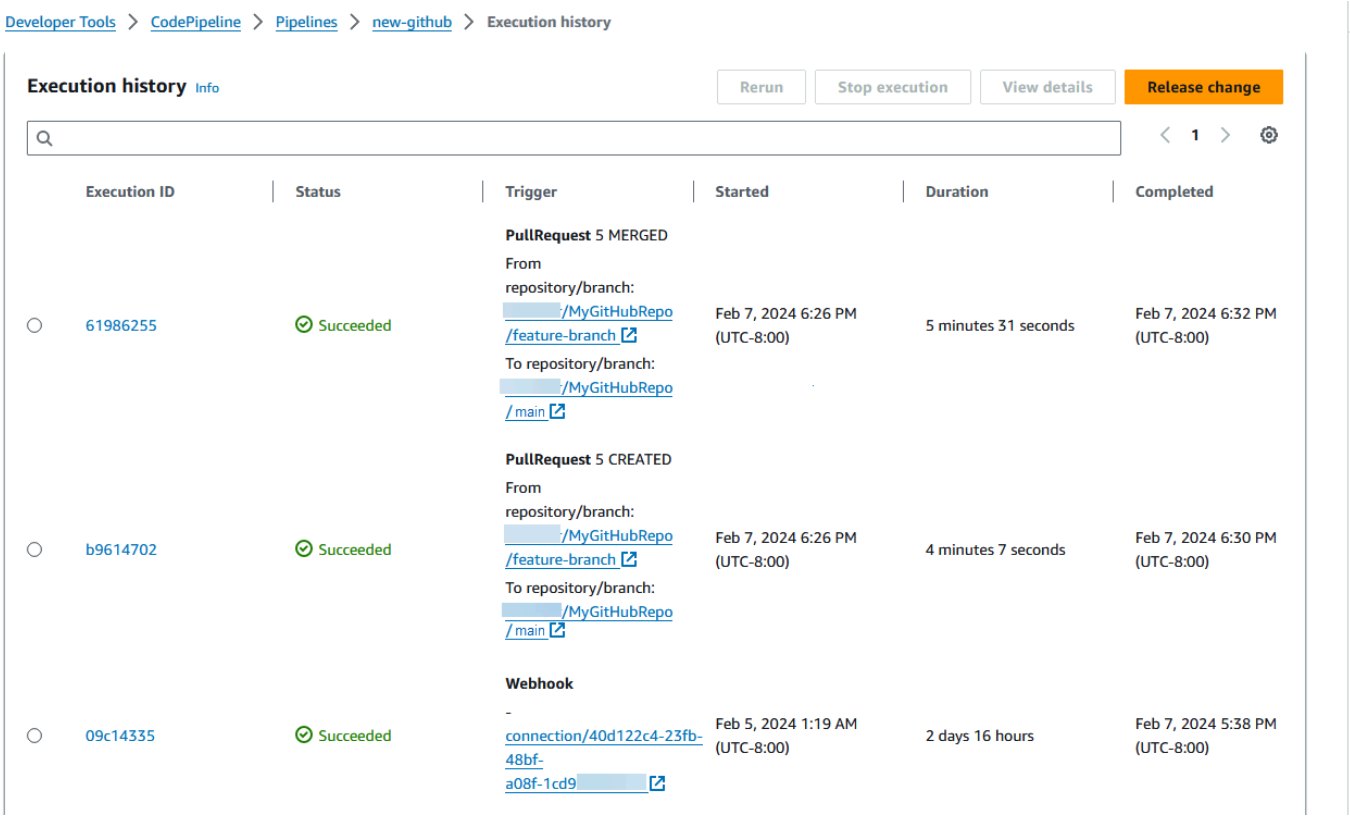
Pour créer une pull request et démarrer votre pipeline

1. Dans GitHub.com, créez une demande d'extraction en modifiant la README.md branche d'une fonctionnalité et en soumettant une demande d'extraction à la `main` branche. Validez la modification avec un message tel que`Update README.md for PR`.

2. Le pipeline commence par la révision de la source affichant le message source pour la pull request sous la forme Update README.md for PR.



3. Choisissez History (Historique). Dans l'historique des exécutions du pipeline, consultez les événements d'état des demandes d'extraction CREATED et MERGED qui ont déclenché les exécutions du pipeline.



Tutoriel : Utiliser des variables au niveau du pipeline

Dans ce didacticiel, vous allez créer un pipeline dans lequel vous ajoutez une variable au niveau du pipeline et exécutez une action de CodeBuild génération qui génère la valeur de votre variable.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Rubriques

- Conditions préalables
- Étape 1 : Créez votre pipeline et créez un projet
- Étape 2 : publier les modifications et afficher les journaux

Conditions préalables

Avant de commencer, vous devez exécuter les opérations suivantes :

- Créez un CodeCommit référentiel.
- Ajoutez un fichier .txt au référentiel.

Étape 1 : Créez votre pipeline et créez un projet

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Stage source avec connexion à votre CodeCommit référentiel.
- Une étape de construction avec une action de AWS CodeBuild construction.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Bienvenue, Démarrez ou Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyVariablesPipeline`**.
5. Dans Type de pipeline, conservez la sélection par défaut sur V2. Les types de pipelines diffèrent en termes de caractéristiques et de prix. Pour de plus amples informations, veuillez consulter Types de canalisations.
6. Dans Service role (Rôle de service), choisissez New service role (Nouveau rôle de service).

   > **Note**
   >
   > Si vous choisissez plutôt d'utiliser votre rôle de CodePipeline service existant, assurez-vous d'avoir ajouté l'autorisation `codeconnections:UseConnection` IAM à votre politique de rôle de service. Pour obtenir des instructions relatives au rôle de CodePipeline service, voir Ajouter des autorisations au rôle CodePipeline de service.

7. Sous Variables, choisissez Ajouter une variable. Pour Name (Nom), entrez `timeout`. Dans la zone Par défaut, entrez 1000. Dans la description, entrez la description suivante :**`Timeout`**.

   Cela créera une variable dans laquelle vous pourrez déclarer la valeur lorsque l'exécution du pipeline démarre. Les noms des variables doivent correspondre `[A-Za-z0-9@\-_]+` et peuvent être n'importe quoi sauf une chaîne vide.
8. Sous Paramètres avancés, conservez les valeurs par défaut. Dans le magasin d'artefacts choisissez Default location (Emplacement par défaut) pour utiliser le magasin d'artefacts par défaut, tel que le compartiment d'artefacts Amazon S3 désigné par défaut, pour votre pipeline dans la région que vous avez sélectionnée pour ce dernier.

> **Note**
>
> Il ne s'agit pas du compartiment source de votre code source. Il s'agit du magasin d'artefacts pour votre pipeline. Un magasin d'artefacts distinct, tel qu'un compartiment S3, est nécessaire pour chaque pipeline.

Choisissez Suivant.

9. Sur la page Étape 3 : Ajouter une étape source, ajoutez une étape source :

   a. Dans Fournisseur de source, choisissez AWS CodeCommit.

   b. Dans Nom du référentiel et Nom de la branche, choisissez votre référentiel et votre branche.

   Choisissez Suivant.

10. À l'étape 4 : Ajouter une étape de construction, ajoutez une étape de construction :

    a. Dans le champ Fournisseur de génération, choisissez AWS CodeBuild. Acceptez la région du pipeline comme Région par défaut.

    b. Sélectionnez Create a project (Créer un projet).

    c. Dans Nom du projet, saisissez un nom pour ce projet de génération.

    d. Dans le champ Image d'environnement, choisissez Image gérée. Pour Système d'exploitation, choisissez Ubuntu.

    e. Pour Runtime (Exécution), sélectionnez Standard. Pour Image, choisissez aws/codebuild / standard:5.0.

    f. Pour Rôle de service, choisissez Nouveau rôle de service.

    > **Note**
    >
    > Notez le nom de votre rôle CodeBuild de service. Vous aurez besoin du nom du rôle pour la dernière étape de ce didacticiel.

    g. Sous Buildspec, pour Build specifications (Spécifications de génération), choisissez Insert build commands (Insérer des commandes de génération). Choisissez Passer à l'éditeur et collez ce qui suit sous Commandes de création. Dans la spécification de construction,

la variable client `$CUSTOM_VAR1` sera utilisée pour générer la variable de pipeline dans le journal de génération. Vous allez créer la variable `$CUSTOM_VAR1` de sortie en tant que variable d'environnement à l'étape suivante.

```
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. Choisissez Continuer vers CodePipeline. Cela revient à la CodePipeline console et crée un CodeBuild projet qui utilise vos commandes de build pour la configuration. Le projet de génération utilise un rôle de service pour gérer les Service AWS autorisations. Cette étape peut prendre quelques minutes.

i. Sous Variables d'environnement - facultatif , pour créer une variable d'environnement en tant que variable d'entrée pour l'action de génération qui sera résolue par la variable au niveau du pipeline, choisissez Ajouter une variable d'environnement. Cela créera la variable spécifiée dans la spécification de construction sous la forme. `$CUSTOM_VAR1` Pour Name (Nom), entrez `CUSTOM_VAR1`. Dans Value (Valeur), entrez `#{variables.timeout}`. Dans Type, choisissez`Plaintext`.

   La `#{variables.timeout}` valeur de la variable d'environnement est basée sur l'espace de noms de la variable au niveau du pipeline `variables` et sur la variable au niveau du pipeline `timeout` créée pour le pipeline à l'étape 7.

j. Choisissez Suivant.

11. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.

12. Sur la page Étape 6 : Ajouter une phase de déploiement, choisissez Ignorer l'étape de déploiement, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer. Choisissez Suivant.

13. À l'étape 7 : Révision, choisissez Créer un pipeline.

Étape 2 : publier les modifications et afficher les journaux

1. Une fois le pipeline exécuté avec succès, lors de la phase de création réussie, choisissez Afficher les détails.

   Sur la page de détails, choisissez l'onglet Journaux. Afficher le résultat CodeBuild de la compilation. Les commandes génèrent la valeur de la variable saisie.

2. Dans la barre de navigation de gauche, sélectionnez Historique.

Choisissez l'exécution récente, puis cliquez sur l'onglet Variables. Affichez la valeur résolue pour la variable de pipeline.

Didacticiel : Création d'un pipeline simple (compartiment S3)

Le moyen le plus simple de créer un pipeline est d'utiliser l'assistant de création de pipeline dans la AWS CodePipeline console.

Dans ce didacticiel, vous allez créer un pipeline en deux étapes qui utilise un compartiment source S3 versionné et CodeDeploy pour publier un exemple d'application.

> **Note**
>
> Lorsqu'Amazon S3 est le fournisseur source de votre pipeline, vous pouvez compresser votre ou vos fichiers source dans un seul fichier .zip et charger le fichier .zip dans votre compartiment source. Vous pouvez également charger un seul fichier décompressé ; toutefois, les actions en aval qui attendent un fichier .zip échoueront.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Une fois ce pipeline simple créé, vous ajoutez une autre étape, puis vous désactivez et activez la transition entre les étapes.

> **Important**
>
> La plupart des actions que vous ajoutez à votre pipeline au cours de cette procédure impliquent AWS des ressources que vous devez créer avant de créer le pipeline. AWS les ressources pour vos actions source doivent toujours être créées dans la même AWS région que celle dans laquelle vous créez votre pipeline. Par exemple, si vous créez votre pipeline

dans la région USA Est (Ohio), votre CodeCommit référentiel doit se trouver dans la région USA Est (Ohio).
Vous pouvez ajouter des actions interrégionales lorsque vous créez votre pipeline. AWS les ressources pour les actions interrégionales doivent se trouver dans la même AWS région que celle où vous prévoyez d'exécuter l'action. Pour de plus amples informations, veuillez consulter Ajouter une action interrégionale dans CodePipeline.

Avant de commencer, suivez les étapes requises détaillées dans Commencer avec CodePipeline.

Rubriques

- Étape 1 : créer un compartiment source S3 pour votre application
- Étape 2 : Création d'instances Windows Amazon EC2 et installation de l'agent CodeDeploy
- Étape 3 : Créez une application dans CodeDeploy
- Étape 4 : Créez votre premier pipeline dans CodePipeline
- (Facultatif) Étape 5 : Ajouter une autre étape à votre pipeline
- (Facultatif) Étape 6 : Désactivez et activez les transitions entre les étapes dans CodePipeline
- Étape 7 : Nettoyer les ressources

Étape 1 : créer un compartiment source S3 pour votre application

Vous pouvez stocker vos fichiers ou applications source dans n'importe quel emplacement versionné. Dans ce didacticiel, vous allez créer un compartiment S3 pour les exemples de fichiers d'application et activer la gestion des versions sur ce compartiment. Une fois que vous aurez activé la gestion des versions, vous pourrez copier les modèles d'application dans ce compartiment.

Pour créer un compartiment S3

1. Connectez-vous à la console à l'adresse Console de gestion AWS. Ouvrez la console S3.
2. Choisissez Créer un compartiment.
3. Dans Nom du compartiment, saisissez un nom pour votre compartiment (par exemple, **`awscodepipeline-demobucket-example-date`**).

> **Note**
>
> Étant donné que tous les noms de compartiment dans Amazon S3 doivent être uniques, utilisez l'un des vôtres, et non le nom indiqué dans l'exemple. Vous pouvez modifier le nom de l'exemple simplement en y ajoutant la date. Notez ce nom, car vous l'utiliserez durant ce didacticiel.

Dans Région, choisissez la région dans laquelle vous souhaitez créer votre pipeline, par exemple USA West (Oregon), puis choisissez Create bucket.

4. Une fois le compartiment créé, une bannière de réussite apparaît. Choisissez Go to bucket details (Accéder aux détails du compartiment).
5. Dans l'onglet Propriétés, choisissez Versioning. Choisissez Activer la gestion des versions, puis Enregistrer.

   Lorsque la gestion des versions est activée, Amazon S3 enregistre chaque version de chaque objet du compartiment.
6. Sous l'onglet Autorisations, laissez les valeurs par défaut. Pour de plus amples informations sur les autorisations des objets et des compartiments S3, veuillez consulter Spécification des autorisations d'une stratégie.
7. Ensuite, téléchargez un exemple et enregistrez-le dans un dossier ou un répertoire sur votre ordinateur local.

   a. Choisissez l'une des options suivantes. Choisissez `SampleApp_Windows.zip` si vous souhaitez suivre les étapes de ce didacticiel pour les instances Windows Server.

      - Si vous souhaitez effectuer un déploiement sur des instances Amazon Linux à l'aide de CodeDeploy, téléchargez l'exemple d'application ici : SampleApp _ Linux.zip.
      - Si vous souhaitez déployer sur des instances Windows Server à l'aide de CodeDeploy, téléchargez l'exemple d'application ici : SampleApp _ Windows.zip.

      L'exemple d'application contient les fichiers suivants à déployer avec CodeDeploy :

      - `appspec.yml`— Le fichier de spécification de l'application (AppSpecfichier) est un fichier au format YAML utilisé CodeDeploy pour gérer un déploiement. Pour plus d'informations

sur le AppSpec fichier, consultez la section Référence CodeDeploy AppSpec du fichier dans le Guide de AWS CodeDeploy l'utilisateur.

- `index.html`— Le fichier d'index contient la page d'accueil de l'exemple d'application déployé.
- `LICENSE.txt`— Le fichier de licence contient les informations de licence de l'exemple d'application.
- Fichiers pour scripts : l'exemple d'application utilise des scripts pour écrire des fichiers texte à un emplacement de votre instance. Un fichier est écrit pour chacun des événements du cycle de vie du CodeDeploy déploiement, comme suit :
  - `scripts`Dossier (exemple Linux uniquement) — Le dossier contient les scripts shell suivants pour installer les dépendances et démarrer et arrêter l'exemple d'application pour le déploiement automatique : `install_dependencies`, `start_server`, et`stop_server`.
  - (Exemple Windows uniquement) `before-install.bat` — Il s'agit d'un script batch pour l'événement du cycle de vie du `BeforeInstall` déploiement, qui sera exécuté pour supprimer les anciens fichiers écrits lors des déploiements précédents de cet exemple et créer un emplacement sur votre instance dans lequel écrire les nouveaux fichiers.

b. Téléchargez le fichier compressé (zippé). Ne décompressez pas le fichier.

8. Dans la console Amazon S3, pour votre compartiment, chargez le fichier :

   a. Choisissez Charger.

   b. Faites glisser et déposez le fichier ou choisissez Ajouter des fichiers et recherchez le fichier.

   c. Choisissez Charger.

Étape 2 : Création d'instances Windows Amazon EC2 et installation de l'agent CodeDeploy

> **Note**
>
> Ce didacticiel fournit des exemples d'étapes pour créer des instances Windows Amazon EC2. Pour des exemples d'étapes permettant de créer des instances Amazon EC2 Linux, consultezÉtape 3 : Création d'une instance Amazon EC2 Linux et installation de l'agent

CodeDeploy. Lorsque vous êtes invité à indiquer le nombre d'instances à créer, spécifiez 2 instances.

Au cours de cette étape, vous allez créer les instances Amazon EC2 de Windows Server sur lesquelles vous allez déployer un exemple d'application. Dans le cadre de ce processus, vous créez un rôle d'instance avec des politiques qui permettent l'installation et la gestion de l' CodeDeploy agent sur les instances. L' CodeDeploy agent est un progiciel qui permet d'utiliser une instance dans le cadre de CodeDeploy déploiements. Vous joignez également des politiques qui permettent à l'instance de récupérer les fichiers que l' CodeDeploy agent utilise pour déployer votre application et de permettre à l'instance d'être gérée par SSM.

Pour créer un rôle d'instance

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le tableau de bord de la console, choisissez Rôles.
3. Choisissez Créer un rôle.
4. Sous Sélectionner le type d'entité de confiance, sélectionnez Service AWS. Sous Choose a use case (Choisir un cas d'utilisation), sélectionnez EC2, puis Suivant : Autorisations.
5. Recherchez et sélectionnez la politique nommée **`AmazonEC2RoleforAWSCodeDeploy`**.
6. Recherchez et sélectionnez la politique nommée **`AmazonSSMManagedInstanceCore`**. Choisissez Suivant : Balises.
7. Choisissez Suivant : Vérification. Saisissez un nom pour le rôle (par exemple, **`EC2InstanceRole`**).

   > **Note**
   >
   > Notez le nom de votre rôle pour l'étape suivante. Vous choisissez ce rôle lorsque vous créez votre instance.

   Choisissez Créer un rôle.

Pour lancer des instances

1. Ouvrez la console Amazon EC2 à l'adresse https://console.aws.amazon.com/ec2/.

2. Dans la navigation latérale, choisissez Instances, puis sélectionnez Lancer des instances en haut de la page.
3. Sous Nom et balises, dans Nom, entrez**MyCodePipelineDemo**. Cela affecte aux instances une clé de balise de **Name** et une valeur de balise de**MyCodePipelineDemo**. Vous créez ultérieurement une CodeDeploy application qui déploie l'exemple d'application sur les instances. CodeDeploysélectionne les instances à déployer en fonction des balises.
4. Sous Images de l'application et du système d'exploitation (Amazon Machine Image), choisissez l'option Windows. (Cette AMI est décrite comme Microsoft Windows Server 2019 Base et est étiquetée « éligible au niveau gratuit » et se trouve sous Quick Start.)
5. Dans Type d'instance, choisissez le `t2.micro` type éligible au niveau gratuit comme configuration matérielle de votre instance.
6. Sous Paire de clés (connexion), choisissez une paire de clés ou créez-en une.

   Vous pouvez également choisir Continuer sans paire de touches.

   > **Note**
   >
   > Pour les besoins de ce didacticiel, vous pouvez procéder sans paire de clés. Pour utiliser SSH pour vous connecter à vos instances, créez ou utilisez une paire de clés.

7. Sous Paramètres réseau, procédez comme suit.

   Dans IP Auto-assign publique, assurez-vous que le statut est Activé.

   - En regard de Attribuer un groupe de sécurité, choisissez Créer un groupe de sécurité.
   - Dans la ligne SSH, sous Type de source, choisissez My IP.
   - Choisissez Ajouter un groupe de sécurité, choisissez HTTP, puis sous Type de source, choisissez Mon IP.
8. Développez Advanced Details (Détails avancés). Dans le profil d'instance IAM, choisissez le rôle IAM que vous avez créé lors de la procédure précédente (par exemple,**EC2InstanceRole**).
9. Dans Résumé, sous Nombre d'instances, saisissez2.
10. Choisissez Launch instance (Lancer une instance).
11. Sélectionnez View all instances (Afficher toutes les instances) pour fermer la page de confirmation et revenir à la console.
12. Sur la page Instances, vous pouvez afficher le statut du lancement. Lorsque vous lancez une instance, son état initial est `pending`. Une fois que l'instance a démarré, son état devient

`running` et elle reçoit un nom DNS public. (Si la colonne DNS public n'est pas affichée, choisissez l'Show/Hideicône, puis sélectionnez DNS public.)

13. Vous devrez peut-être patienter quelques minutes avant de pouvoir vous connecter à l'instance. Vérifiez que votre instance a réussi les contrôles de statut. Cette information est visible dans la colonne Contrôles des statuts.

Étape 3 : Créez une application dans CodeDeploy

Dans CodeDeploy, une application est un identifiant, sous la forme d'un nom, pour le code que vous souhaitez déployer. CodeDeploy utilise ce nom pour garantir que la combinaison correcte de révision, de configuration de déploiement et de groupe de déploiement est référencée lors d'un déploiement. Vous sélectionnez le nom de l' CodeDeploy application que vous créez à cette étape lorsque vous créerez votre pipeline ultérieurement dans ce didacticiel.

Vous devez d'abord créer un rôle de service CodeDeploy à utiliser. Si vous avez déjà créé un rôle de service, il n'est pas nécessaire d'en créer un autre.

Pour créer un rôle CodeDeploy de service

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le tableau de bord de la console, choisissez Rôles.
3. Choisissez Créer un rôle.
4. Sous Sélectionner une entité de confiance, choisissez Service AWS. Sous Use case (Cas d'utilisation), choisissez CodeDeploy. Choisissez CodeDeploy parmi les options répertoriées. Choisissez Suivant. La stratégie gérée `AWSCodeDeployRole` est déjà attachée au rôle.
5. Choisissez Suivant.
6. Entrez un nom pour le rôle (par exemple, **`CodeDeployRole`**), puis choisissez Créer un rôle.

Pour créer une application dans CodeDeploy

1. Ouvrez la CodeDeploy console à l'adresse https://console.aws.amazon.com/codedeploy.
2. Si la page Applications n'apparaît pas, AWS CodeDeploy dans le menu, choisissez Applications.
3. Choisissez Créer une application.
4. Dans Nom de l'application, saisissez `MyDemoApplication`.

5. Dans Compute Platform, sélectionnez EC2/On-premises.
6. Choisissez Créer une application.

Pour créer un groupe de déploiement dans CodeDeploy

1. Sur la page qui affiche votre application, choisissez Créer un groupe de déploiement.
2. Dans Nom du groupe de déploiement, saisissez **`MyDemoDeploymentGroup`**.
3. Dans Rôle de service, choisissez le rôle de service que vous avez créé précédemment. Vous devez utiliser un rôle de service approuvé AWS CodeDeploy avec, au minimum, la confiance et les autorisations décrites dans Créer un rôle de service pour CodeDeploy. Pour connaître le rôle de service ARN, consultez Obtention de l'ARN du rôle de service (console).
4. Dans Type de déploiement, sélectionnez In-place.
5. Sous Configuration de l'environnement, choisissez Instances Amazon EC2. Choisissez Nom dans le champ Clé, puis saisissez dans le champ Valeur**`MyCodePipelineDemo`**.

   > **Important**
   >
   > Vous devez choisir ici la même valeur pour la clé Nom que la valeur que vous avez attribuée à vos instances EC2 lorsque vous les avez créées. Si vous avez balisé vos instances avec une valeur autre que **`MyCodePipelineDemo`**, veillez à l'utiliser ici.

6. Dans Configuration de l'agent avec AWS Systems Manager, choisissez Maintenant et planifiez les mises à jour. Cela permet d'installer l'agent sur l'instance. L'instance Windows est déjà configurée avec l'agent SSM et va maintenant être mise à jour avec l' CodeDeploy agent.
7. Dans Paramètres de déploiement, choisissez`CodeDeployDefault.OneAtaTime`.
8. Dans Load Balancer, assurez-vous que la case Activer l'équilibrage de charge n'est pas sélectionnée. Vous n'avez pas besoin de configurer un équilibreur de charge ou de choisir un groupe cible pour cet exemple. Une fois que vous avez désélectionné la case, les options de l'équilibreur de charge ne s'affichent pas.
9. Dans la section Avancé, laissez les valeurs par défaut.
10. Choisissez Créer un groupe de déploiement.

Étape 4 : Créez votre premier pipeline dans CodePipeline

Dans cette partie du didacticiel, vous créez le pipeline. Le modèle s'exécute automatiquement par le biais du pipeline.

Pour créer un processus de publication CodePipeline automatique

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyFirstPipeline`**.

   > **Note**
   >
   > Si vous choisissez un autre nom pour votre pipeline, veillez à utiliser ce nom au lieu de **`MyFirstPipeline`** pendant le reste de ce didacticiel. Une fois le pipeline créé, vous ne pouvez plus modifier son nom. Les noms de pipeline sont soumis à des limites. Pour de plus amples informations, veuillez consulter Quotas dans AWS CodePipeline.

5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.
6. Dans Rôle du service, sélectionnez l'une des options suivantes :

   - Choisissez Nouveau rôle de service pour permettre CodePipeline de créer un nouveau rôle de service dans IAM.
   - Choisissez Existing service role (Rôle de service existant) pour utiliser un rôle de service déjà créé dans IAM. Dans Role name (Nom du rôle), choisissez votre rôle de service à partir de la liste.

7. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
8. À l'étape 3 : Ajouter une étape source, dans Source provider, choisissez Amazon S3. Dans Bucket (Compartiment), entrez le nom du compartiment S3 que vous avez créé dans Étape 1 :

créer un compartiment source S3 pour votre application. Dans S3 object key (Clé d'objet S3), entrez la clé d'objet avec ou sans chemin d'accès au fichier, sans oublier d'inclure l'extension de fichier. Par exemple, pour `SampleApp_Windows.zip`, saisissez le nom de l'exemple de fichier, comme indiqué dans cet exemple :

```
SampleApp_Windows.zip
```

Choisissez Étape suivante.

Sous Modifier les options de détection, ne modifiez pas les valeurs par défaut. Cela permet CodePipeline d'utiliser Amazon CloudWatch Events pour détecter les modifications dans votre compartiment source.

Choisissez Suivant.

9. À l'étape 4 : Ajouter une étape de construction, choisissez Ignorer l'étape de génération, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer. Choisissez Suivant.
10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.
11. À l'étape 6 : Ajouter une étape de déploiement, dans Deploy provider, choisissez CodeDeploy. Par défaut, le champ Région est Région AWS identique à celui de votre pipeline. Dans Application name (Nom de l'application), saisissez `MyDemoApplication`, ou cliquez sur le bouton Refresh (Actualiser) et choisissez le nom de l'application dans la liste. Dans Groupe de déploiement, entrez **`MyDemoDeploymentGroup`** ou choisissez-le dans la liste, puis choisissez Suivant.

    > **Note**
    >
    > « Déploiement » est le nom donné par défaut à l'étape du pipeline créée à l'Étape 4 : Ajouter une étape de déploiement, tout comme « Source » est le nom donné à la première étape du pipeline.
12. À l'étape 7 : Passez en revue, passez en revue les informations, puis choisissez Créer un pipeline.

13. Le pipeline commence à s'exécuter. Vous pouvez consulter les messages de progression, de réussite et d'échec au fur et à mesure que l' CodePipeline exemple déploie une page Web sur chacune des instances Amazon EC2 du CodeDeploy déploiement.

Félicitations ! Vous venez de créer un pipeline simple dans CodePipeline. Le pipeline comporte deux étapes :

- Une étape source nommée Source, qui détecte les modifications apportées à l'exemple d'application versionnée stocké dans le compartiment S3 et extrait ces modifications dans le pipeline.
- Une étape Déploiement qui déploie ces modifications dans les instances EC2 avec CodeDeploy.

Maintenant, vérifiez les résultats.

Pour vérifier que votre pipeline a été exécuté avec succès

1. Affichez la progression initiale du pipeline. L'état de chaque étape passe de Pas encore d'exécution à En cours, puis devient soit Réussi, soit Échec. Le pipeline doit terminer la première exécution en quelques minutes.
2. Lorsque l'état de l'action affiche Réussi, dans la zone d'état de l'étape Déployer, choisissez Détails. Cela ouvre la CodeDeploy console.
3. Dans l'onglet Groupe de déploiement, sous Événements du cycle de vie de déploiement, choisissez un ID d'instance. Ceci ouvre la console EC2.
4. Dans l'onglet Description, sous DNS public, copiez l'adresse, puis collez-la dans la barre d'adresses de votre navigateur web. Veuillez consulter la page d'index pour le modèle d'application que vous avez chargé dans votre compartiment S3.

   La page Web s'affiche pour l'exemple d'application que vous avez chargé dans votre compartiment S3.

Pour plus d'informations sur les étapes, les actions et le fonctionnement des pipelines, consultez CodePipeline concepts.

(Facultatif) Étape 5 : Ajouter une autre étape à votre pipeline

Ajoutez maintenant une autre étape dans le pipeline, pour un déploiement depuis des serveurs intermédiaires aux serveurs de production à l'aide d' CodeDeploy. Tout d'abord, vous créez un autre

groupe de déploiement CodePipelineDemoApplication dans le CodeDeploy. Ensuite, vous ajoutez une étape qui inclut une action utilisant ce groupe de déploiement. Pour ajouter une autre étape, vous utilisez la CodePipeline console ou le AWS CLI pour récupérer et modifier manuellement la structure du pipeline dans un fichier JSON, puis exécutez la update-pipeline commande pour mettre à jour le pipeline avec vos modifications.

Rubriques

- Créez un deuxième groupe de déploiement dans CodeDeploy
- Ajout du groupe de déploiement en tant qu'étape supplémentaire dans votre pipeline

Créez un deuxième groupe de déploiement dans CodeDeploy

> **Note**
>
> Dans cette partie du didacticiel, vous allez créer un deuxième groupe de déploiement, mais vous le déployez sur les mêmes instances Amazon EC2 que précédemment. Cette étape est uniquement abordée à des fins pédagogiques. Il est délibérément conçu pour ne pas vous montrer comment les erreurs sont affichées dans. CodePipeline

Pour créer un deuxième groupe de déploiement dans CodeDeploy

1. Ouvrez la CodeDeploy console à l'adresse https://console.aws.amazon.com/codedeploy.
2. Choisissez Applications, puis dans la liste des applications, choisissez `MyDemoApplication`.
3. Choisissez l'onglet Groupes de déploiement, puis choisissez Créer un groupe de déploiement.
4. Sur la page Créer un groupe de déploiement, sous Nom du groupe de déploiement, saisissez un nom pour le deuxième groupe de déploiement (par exemple, **`CodePipelineProductionFleet`**).
5. Dans Service Role, choisissez le même rôle CodeDeploy de service que vous avez utilisé pour le déploiement initial (pas le rôle CodePipeline de service).
6. Dans Type de déploiement, sélectionnez In-place.
7. Sous Configuration de l'environnement, choisissez Instances Amazon EC2. Choisissez Nom dans la zone Clé et, dans la zone Valeur, choisissez `MyCodePipelineDemo` dans la liste. Conservez la configuration par défaut pour Paramètres de déploiement.
8. Dans Configuration de déploiement, choisissez `CodeDeployDefault.OneAtaTime`.

9. Dans Équilibreur de charge, désélectionnez Activer l'équilibrage de charge.
10. Choisissez Créer un groupe de déploiement.

Ajout du groupe de déploiement en tant qu'étape supplémentaire dans votre pipeline

Maintenant que vous disposez d'un autre groupe de déploiement, vous pouvez ajouter une étape qui utilise ce groupe de déploiement pour procéder à un déploiement sur les mêmes instances EC2 que vous avez utilisées précédemment. Vous pouvez utiliser la CodePipeline console ou le AWS CLI pour ajouter cette étape.

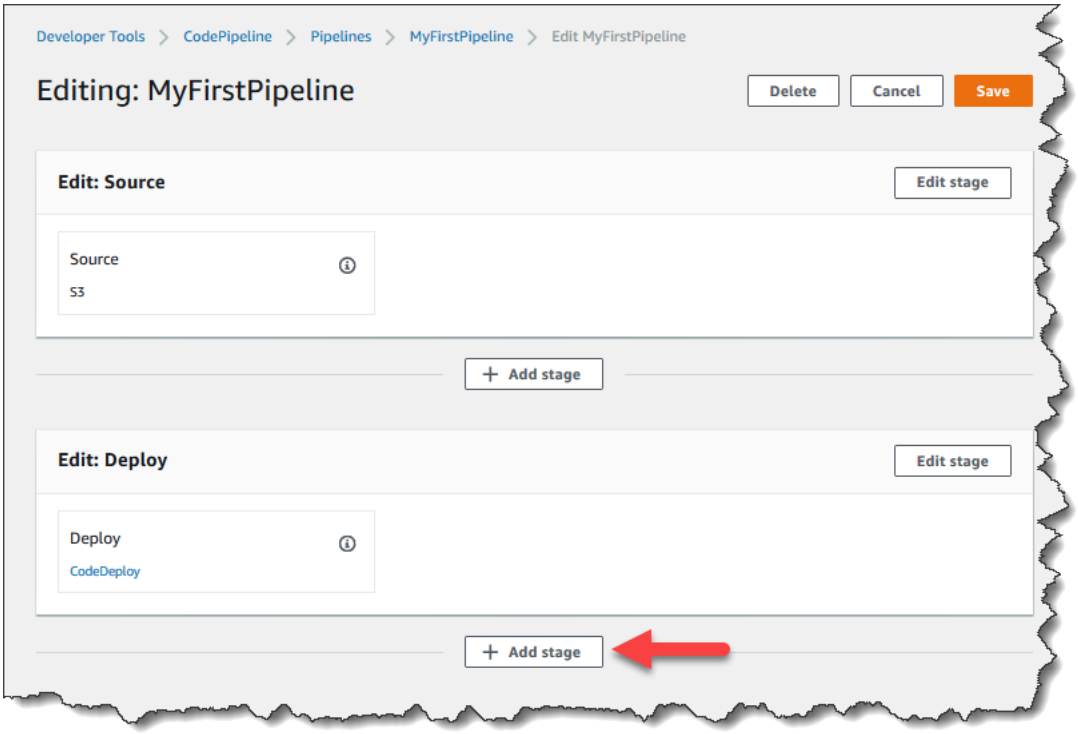
Rubriques

- Création d'une troisième étape (console)
- Création d'une troisième étape (interface de ligne de commande)

Création d'une troisième étape (console)

Vous pouvez utiliser la CodePipeline console pour ajouter une nouvelle étape qui utilise le nouveau groupe de déploiement. Ce groupe de déploiement se déploie dans les instances EC2 que vous avez déjà utilisées ; par conséquent l'action de déploiement échoue dans cette étape.

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Dans Nom, choisissez le nom du pipeline que vous avez créé, MyFirstPipeline.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Sur la page Modifier, choisissez + Ajouter une étape pour ajouter une étape immédiatement après l'étape Déploiement.



5. Dans Add stage (Ajouter une étape), dans Stage name (Nom de l'étape), saisissez **`Production`**. Sélectionnez Ajouter une étape.
6. Dans la nouvelle étape, choisissez + Add action group (+ Ajouter un groupe d'actions).
7. Dans Edit action (Modifier l'action), dans Action name (Nom de l'action), saisissez **`Deploy-Second-Deployment`**. Dans Action provider, sous Déployer, choisissez CodeDeploy.
8. Dans la CodeDeploy section, dans Nom `MyDemoApplication` de l'application, choisissez dans la liste déroulante, comme vous l'avez fait lors de la création du pipeline. Dans Groupe de déploiement, choisissez le groupe de déploiement que vous venez de créer, **`CodePipelineProductionFleet`**. Dans Input artifacts (Artefacts d'entrée), choisissez l'artefact d'entrée à partir de l'action source. Choisissez Enregistrer.
9. Sur la page Modifier, choisissez Enregistrer. Dans Enregistrer les modifications du pipeline, choisissez Enregistrer.
10. Bien que la nouvelle étape ait été ajoutée à votre pipeline, l'état affiché est Pas encore d'exécution, car aucune modification n'a déclenché une autre exécution du pipeline. Vous devez relancer manuellement la dernière révision pour voir comment le pipeline modifié s'exécute. Sur la page des détails du pipeline, choisissez Release change, puis choisissez Release lorsque vous y êtes invité. Cette opération exécute la révision la plus récente disponible dans chaque emplacement source spécifié d'une action source à travers le pipeline.

Vous pouvez également utiliser le AWS CLI pour réexécuter le pipeline, depuis un terminal sur votre machine Linux, macOS ou Unix locale, ou depuis une invite de commande sur votre machine Windows locale, exécutez la start-pipeline-execution commande en spécifiant le nom du pipeline. Cette action exécute l'application dans votre compartiment source par le biais du pipeline une seconde fois.

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

Cette commande renvoie un objet `pipelineExecutionId`.

11. Revenez à la CodePipeline console et, dans la liste des pipelines, choisissez MyFirstPipeline d'ouvrir la page d'affichage.

    Le pipeline affiche trois étapes et l'état de l'artefact en cours d'exécution durant ces trois étapes. Le pipeline exécute alors toutes les étapes, ce qui peut prendre jusqu'à cinq minutes. Vous voyez que le déploiement est réussi lors des deux premières étapes, comme précédemment, mais la phase de production indique que l'Deploy-Second-Deploymentaction a échoué.

12. Dans l'Deploy-Second-Deploymentaction, choisissez Détails. Vous êtes redirigé vers la page concernant le déploiement CodeDeploy. Dans ce cas, l'échec résulte du fait que, le premier groupe d'instance ayant été déployé sur toutes les instances EC2, aucune instance n'est disponible pour le second groupe de déploiement.

    > **Note**
    >
    > Cet échec est intentionnel, afin de démontrer ce qui se passe en cas d'échec dans une étape du pipeline.

Création d'une troisième étape (interface de ligne de commande)

Bien que l'utilisation AWS CLI de pour ajouter une étape à votre pipeline soit plus complexe que l'utilisation de la console, elle offre une meilleure visibilité sur la structure du pipeline.

Pour créer une troisième étape pour votre pipeline

1. Ouvrez une session de terminal sur votre machine Linux, macOS ou Unix locale, ou une invite de commande sur votre machine Windows locale, puis exécutez la get-pipeline commande pour

afficher la structure du pipeline que vous venez de créer. Pour **MyFirstPipeline**, saisissez la commande suivante :

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

Cette commande renvoie la structure de MyFirstPipeline. La première partie du résultat doit être semblable à l'exemple suivant :

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

La dernière partie de la sortie inclut les métadonnées du pipeline et doit être similaire à ce qui suit :

```
    ...
        ],
        "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. Copiez et collez cette structure dans un éditeur de texte brut et enregistrez le fichier en tant que **pipeline.json**. Pour plus de commodité, enregistrez ce fichier dans le même répertoire où vous exécutez les commandes aws codepipeline.

> **Note**
>
> Vous pouvez diriger le JSON directement dans un fichier avec la commande get-pipeline, en procédant comme suit :
>
> ```
> aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
> ```

3. Copiez la section de l'étape Déployer et collez-la après les deux premières étapes. Il s'agit d'une étape de déploiement, tout comme l'étape Déployer, par conséquent vous l'utiliserez comme modèle pour la troisième étape.
4. Modifiez le nom de l'étape et les détails du groupe de déploiement.

   L'exemple suivant montre le JSON que vous ajoutez au fichier pipeline.json après la phase de déploiement. Modifiez les éléments mis en surbrillance en insérant les nouvelles valeurs. N'oubliez pas d'inclure une virgule pour séparer les définitions des étapes Déployer et Production.

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

```
        }
    ]
}
```

5. Si vous utilisez la structure de pipeline extraite à l'aide de la commande get-pipeline, vous devez supprimer les lignes `metadata` du fichier JSON. Sinon, la commande update-pipeline ne peut pas l'utiliser. Supprimez les lignes `"metadata": { }` et les champs `"updated"`, `"created"` et `"pipelineARN"`.

   Par exemple, supprimez les lignes suivantes de la structure :

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

   Enregistrez le fichier.

6. Exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline d'une manière similaire à l'exemple suivant :

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

   Cette commande renvoie toute la structure du pipeline mise à jour.

   > **Important**
   >
   > N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

7. Exécutez la commande start-pipeline-execution en spécifiant le nom du pipeline. Cette action exécute l'application dans votre compartiment source par le biais du pipeline une seconde fois.

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

   Cette commande renvoie un objet `pipelineExecutionId`.

8. Ouvrez la CodePipeline console et faites votre choix MyFirstPipeline dans la liste des pipelines.

Le pipeline affiche trois étapes et l'état de l'artefact en cours d'exécution durant ces trois étapes. Le pipeline exécute alors toutes les étapes, ce qui peut prendre jusqu'à cinq minutes. Bien que le déploiement soit un succès lors des deux premières étapes, tout comme auparavant, la phase de production montre que l'Deploy-Second-Deploymentaction a échoué.

9. Dans l'Deploy-Second-Deploymentaction, choisissez Détails pour voir les détails de l'échec. Vous êtes redirigé vers la page de détails du CodeDeploy déploiement. Dans ce cas, l'échec résulte du fait que, le premier groupe d'instance ayant été déployé sur toutes les instances EC2, aucune instance n'est disponible pour le second groupe de déploiement.

> **Note**
>
> Cet échec est intentionnel, afin de démontrer ce qui se passe en cas d'échec dans une étape du pipeline.

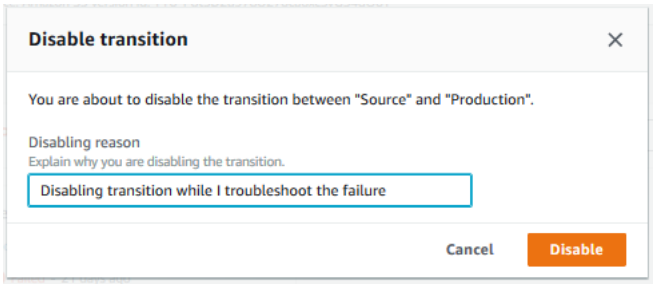
(Facultatif) Étape 6 : Désactivez et activez les transitions entre les étapes dans CodePipeline

Vous pouvez activer ou désactiver la transition entre les étapes d'un pipeline. La désactivation de la transition entre les étapes vous permet de contrôler manuellement les transitions d'une étape à une autre. Par exemple, vous pouvez exécuter les deux premières étapes d'un pipeline, puis désactiver les transitions vers la troisième étape tant que vous n'êtes pas prêt à procéder au déploiement sur la production, ou bien si vous devez remédier à un problème ou à un échec dans cette étape.

Pour activer et désactiver les transitions entre les étapes d'un CodePipeline pipeline

1. Ouvrez la CodePipeline console et faites votre choix MyFirstPipeline dans la liste des pipelines.
2. Sur la page des détails du pipeline, choisissez le bouton Désactiver la transition entre la deuxième étape, (Déployer) et la troisième étape que vous avez ajoutée à la section précédente (Production).
3. Dans Disable transition (Désactiver la transition), saisissez une raison pour désactiver la transition entre les étapes et choisissez Disable (Désactiver).

   La flèche entre les étapes affiche une icône et le changement de couleur, ainsi que le bouton Enable transition (Activer la transition).



4. Chargez à nouveau votre modèle dans le compartiment S3. Le compartiment étant versionné, ce changement lance le pipeline.
5. Revenez à la page des détails de votre pipeline et observez l'état des étapes. Le pipeline affiche alors les réussites et les progrès en cours dans les deux premières étapes, mais aucune modification ne se produit au niveau de la troisième étape. Ce processus peut prendre quelques minutes.
6. Activez la transition en cliquant sur le bouton Enable transition (Activer la transition) entre les deux étapes. Dans la boîte de dialogue Activation d'une transition, choisissez Activer. L'étape lance l'exécution en quelques minutes et essaie de traiter l'artefact qui a déjà été exécuté dans les deux premières étapes du pipeline.

> **Note**
>
> Si vous voulez que cette troisième étape réussisse, modifiez le groupe de CodePipelineProductionFleet déploiement avant d'activer la transition et spécifiez un autre ensemble d'instances EC2 dans lequel l'application est déployée. Pour de plus amples informations sur cette étape, veuillez consulter Modification des paramètres de groupe de déploiement. La création d'instances EC2 supplémentaires peut entraîner des frais.

Étape 7 : Nettoyer les ressources

Vous pouvez utiliser certaines des ressources que vous avez créées au cours de ce didacticiel pour le Didacticiel : Création d'un pipeline à quatre étapes. Par exemple, vous pouvez réutiliser l' CodeDeploy application et le déploiement. Vous pouvez configurer une action de génération auprès d'un fournisseur tel que CodeBuild, qui est un service de génération entièrement géré dans le cloud. Vous pouvez également configurer une action de génération qui utilise un fournisseur avec un serveur ou un système de génération, tel que Jenkins.

Toutefois, une fois ce didacticiel terminé, ou d'autres, vous devez supprimer le pipeline, ainsi que les ressources qu'il utilise, afin que l'utilisation de ces ressources ne vous soit pas facturée. Supprimez d'abord le pipeline, puis l' CodeDeploy application et ses instances Amazon EC2 associées, et enfin, le compartiment S3.

Pour nettoyer les ressources utilisées dans ce didacticiel

1. Pour nettoyer vos CodePipeline ressources, suivez les instructions de la section Supprimer un pipeline dans AWS CodePipeline.
2. Pour nettoyer vos CodeDeploy ressources, suivez les instructions de la section Pour nettoyer les ressources (console).
3. Pour supprimer le compartiment S3, suivez les instructions contenues dans Suppression ou vidage d'un compartiment. Si vous ne souhaitez pas créer davantage de pipelines, supprimez le compartiment S3 créé pour le stockage de vos artefacts de pipeline. Pour plus d'informations sur ce compartiment, consultez CodePipeline concepts.

Tutoriel : Création d'un pipeline simple (CodeCommit référentiel)

Dans ce didacticiel, vous allez CodePipeline déployer du code maintenu dans un CodeCommit référentiel sur une seule instance Amazon EC2. Votre pipeline est déclenché lorsque vous envoyez une modification au CodeCommit référentiel. Le pipeline déploie vos modifications sur une instance Amazon EC2 en CodeDeploy tant que service de déploiement.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Le pipeline comporte deux étapes :

- Une étape source (Source) pour votre action CodeCommit source.
- Une étape de déploiement (Déploiement) pour votre action CodeDeploy de déploiement.

La méthode la plus simple pour démarrer AWS CodePipeline est d'utiliser l'assistant de création de pipeline dans la CodePipeline console.

> **Note**
>
> Avant de commencer, assurez-vous d'avoir configuré votre client Git pour qu'il fonctionne avec CodeCommit. Pour obtenir des instructions, veuillez consulter Configuration pour CodeCommit.

Étape 1 : Création d'un CodeCommit référentiel

Tout d'abord, vous créez un référentiel dans CodeCommit. Votre pipeline obtient un code source à partir de ce référentiel lorsqu'il s'exécute. Vous créez également un référentiel local dans lequel vous gérez et mettez à jour le code avant de le transférer vers le CodeCommit référentiel.

Pour créer un CodeCommit référentiel

1. Ouvrez la CodeCommit console à l'adresse https://console.aws.amazon.com/codecommit/.
2. Dans le sélecteur de région, choisissez l' Région AWS endroit où vous souhaitez créer le référentiel et le pipeline. Pour plus d'informations, consultez Régions AWS and Endpoints.
3. Dans la page Référentiels, choisissez Créer un référentiel.
4. Sur la page Créer un référentiel, pour Nom du référentiel, entrez un nom pour votre référentiel (par exemple, **`MyDemoRepo`**).
5. Choisissez Créer.

> **Note**
>
> Les étapes restantes de ce didacticiel sont utilisées **`MyDemoRepo`** pour le nom de votre CodeCommit référentiel. Si vous choisissez un autre nom, veillez à l'utiliser tout au long de ce didacticiel.

Pour configurer un référentiel local

Au cours de cette étape, vous configurez un référentiel local pour vous connecter à votre référentiel CodeCommit distant.

> **Note**
>
> Vous n'êtes pas obligé de configurer un référentiel local. Vous pouvez également utiliser la console pour télécharger des fichiers comme décrit dansÉtape 2 : ajouter un exemple de code à votre CodeCommit référentiel.

1. Une fois votre nouveau référentiel ouvert dans la console, choisissez Clone URL (URL du clone) en haut à droite de la page, puis Clone SSH (Cloner SSH). L'adresse pour cloner votre référentiel Git est copiée dans votre presse-papiers.
2. Dans votre terminal ou votre ligne de commande, accédez à un répertoire local dans lequel vous souhaitez que votre référentiel local soit stocké. Dans ce didacticiel, nous utilisons `/tmp`.
3. Exécutez la commande suivante pour cloner le référentiel, en remplaçant l'adresse SSH par celle que vous avez copiée à l'étape précédente. Cette commande crée un répertoire appelé `MyDemoRepo`. Vous copiez un exemple d'application dans ce répertoire.

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

Étape 2 : ajouter un exemple de code à votre CodeCommit référentiel

Au cours de cette étape, vous téléchargez le code d'un exemple d'application qui a été créé pour un CodeDeploy exemple de présentation, et vous l'ajoutez à votre CodeCommit référentiel.

1. Ensuite, téléchargez un exemple et enregistrez-le dans un dossier ou un répertoire sur votre ordinateur local.

   a. Choisissez l'une des options suivantes. Choisissez `SampleApp_Linux.zip` si vous souhaitez suivre les étapes de ce didacticiel pour les instances Linux.

   - Si vous souhaitez effectuer un déploiement sur des instances Amazon Linux à l'aide de CodeDeploy, téléchargez l'exemple d'application ici : SampleApp _ Linux.zip.
   - Si vous souhaitez déployer sur des instances Windows Server à l'aide de CodeDeploy, téléchargez l'exemple d'application ici : SampleApp _ Windows.zip.

   L'exemple d'application contient les fichiers suivants à déployer avec CodeDeploy :

- `appspec.yml`— Le fichier de spécification de l'application (AppSpecfichier) est un fichier au format YAML utilisé CodeDeploy pour gérer un déploiement. Pour plus d'informations sur le AppSpec fichier, consultez la section Référence CodeDeploy AppSpec du fichier dans le Guide de AWS CodeDeploy l'utilisateur.
- `index.html`— Le fichier d'index contient la page d'accueil de l'exemple d'application déployé.
- `LICENSE.txt`— Le fichier de licence contient les informations de licence de l'exemple d'application.
- Fichiers pour scripts : l'exemple d'application utilise des scripts pour écrire des fichiers texte à un emplacement de votre instance. Un fichier est écrit pour chacun des événements du cycle de vie du CodeDeploy déploiement, comme suit :
  - `scripts`Dossier (exemple Linux uniquement) — Le dossier contient les scripts shell suivants pour installer les dépendances et démarrer et arrêter l'exemple d'application pour le déploiement automatique : `install_dependencies`, `start_server`, et`stop_server`.
  - (Exemple Windows uniquement) `before-install.bat` — Il s'agit d'un script batch pour l'événement du cycle de vie du `BeforeInstall` déploiement, qui sera exécuté pour supprimer les anciens fichiers écrits lors des déploiements précédents de cet exemple et créer un emplacement sur votre instance dans lequel écrire les nouveaux fichiers.

b. Téléchargez le fichier compressé (zippé).

2. Décompressez les fichiers de SampleApp _ Linux.zip dans le répertoire local que vous avez créé précédemment (par exemple, `/tmp/MyDemoRepo` ou`c:\temp\MyDemoRepo`).

   Veillez à placer les fichiers directement dans votre répertoire local. N'incluez pas un dossier `SampleApp_Linux`. Sur votre machine Linux, macOS ou Unix locale, par exemple, votre hiérarchie de répertoires et de fichiers doit ressembler à ceci :

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

```
            #-- stop_server
```

3. Pour télécharger des fichiers dans votre référentiel, utilisez l'une des méthodes suivantes.

   a. Pour télécharger vos fichiers à l' CodeCommit aide de la console, procédez comme suit :

      i. Ouvrez la CodeCommit console et choisissez votre référentiel dans la liste des référentiels.

      ii. Choisissez Ajouter un fichier, puis choisissez Charger le fichier.

      iii. Sélectionnez Choose file (Choisir un fichier), puis naviguez vers votre fichier. Pour ajouter un fichier dans un dossier, choisissez Créer un fichier, puis entrez le nom du dossier avec le nom du fichier, par exemple`scripts/install_dependencies`. Collez le contenu du fichier dans le nouveau fichier.

      Validez la modification en entrant votre nom d'utilisateur et votre adresse e-mail.

      Choisissez Valider les modifications.

      iv. Répétez cette étape pour chaque fichier.

      Le contenu de votre référentiel devrait ressembler à ceci :

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

   b. Pour utiliser les commandes git afin de télécharger vos fichiers, procédez comme suit :

      i. Passez dans le répertoire de votre référentiel local :

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

      ii. Exécutez la commande suivante pour organiser tous vos fichiers à la fois :

```
git add -A
```

iii. Exécutez la commande suivante pour valider les fichiers avec un message de validation :

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

iv. Exécutez la commande suivante pour publier les fichiers depuis votre référentiel local vers votre référentiel CodeCommit :

```
git push
```

4. Les fichiers que vous avez téléchargés et ajoutés à votre dépôt local ont maintenant été ajoutés à la `main` branche de votre CodeCommit `MyDemoRepo` référentiel et sont prêts à être inclus dans un pipeline.

Étape 3 : Création d'une instance Amazon EC2 Linux et installation de l'agent CodeDeploy

Au cours de cette étape, vous créez l'instance Amazon EC2 dans laquelle vous déployez un exemple d'application. Dans le cadre de ce processus, créez un rôle d'instance qui permet d'installer et de gérer l' CodeDeploy agent sur l'instance. L' CodeDeploy agent est un progiciel qui permet d'utiliser une instance dans le cadre de CodeDeploy déploiements. Vous joignez également des politiques qui permettent à l'instance de récupérer les fichiers que l' CodeDeploy agent utilise pour déployer votre application et de permettre à l'instance d'être gérée par SSM.

Pour créer un rôle d'instance

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le tableau de bord de la console, choisissez Rôles.
3. Choisissez Créer un rôle.
4. Sous Sélectionner le type d'entité de confiance, sélectionnez Service AWS. Dans Choisir un cas d'utilisation, sélectionnez EC2. Sous Select your use case (Sélectionner votre cas d'utilisation), choisissez EC2. Choisissez Suivant : Autorisations.
5. Recherchez et sélectionnez la politique nommée **`AmazonEC2RoleforAWSCodeDeploy`**.
6. Recherchez et sélectionnez la politique nommée **`AmazonSSMManagedInstanceCore`**. Choisissez Suivant : Balises.

7. Choisissez Suivant : Vérification. Saisissez un nom pour le rôle (par exemple, **`EC2InstanceRole`**).

   > **Note**
   >
   > Notez le nom de votre rôle pour l'étape suivante. Vous choisissez ce rôle lorsque vous créez votre instance.

   Choisissez Créer un rôle.

Pour lancer une instance

1. Ouvrez la console Amazon EC2 à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans la navigation latérale, choisissez Instances, puis sélectionnez Lancer des instances en haut de la page.
3. Pour Name (Nom), entrez **`MyCodePipelineDemo`**. Cela affecte à l'instance une clé de balise de **`Name`** et une valeur de balise de**`MyCodePipelineDemo`**. Vous créerez ultérieurement une CodeDeploy application qui déploie l'exemple d'application sur cette instance. CodeDeploysélectionne les instances à déployer en fonction des balises.
4. Sous Images de l'application et du système d'exploitation (Amazon Machine Image), recherchez l'option Amazon Linux AMI avec le AWS logo et assurez-vous qu'elle est sélectionnée. (Cette AMI est décrite comme l'AMI Amazon Linux 2 (HVM) et est étiquetée « éligible au niveau gratuit ».)
5. Dans Type d'instance, choisissez le `t2.micro` type éligible au niveau gratuit comme configuration matérielle de votre instance.
6. Sous Paire de clés (connexion), choisissez une paire de clés ou créez-en une.

   Vous pouvez également choisir Continuer sans paire de touches.

   > **Note**
   >
   > Pour les besoins de ce didacticiel, vous pouvez procéder sans paire de clés. Pour utiliser SSH pour vous connecter à vos instances, créez ou utilisez une paire de clés.

7. Sous Paramètres réseau, procédez comme suit.

Dans IP Auto-assign publique, assurez-vous que le statut est Activé.

Pour le groupe de sécurité créé, choisissez HTTP, puis sous Type de source, choisissez My IP.

8. Développez Advanced Details (Détails avancés). Dans le profil d'instance IAM, choisissez le rôle IAM que vous avez créé lors de la procédure précédente (par exemple,**`EC2InstanceRole`**).
9. Dans Résumé, sous Nombre d'instances, saisissez`1`.
10. Choisissez Launch instance (Lancer une instance).
11. Sur la page Instances, vous pouvez afficher le statut du lancement. Lorsque vous lancez une instance, son état initial est `pending`. Une fois que l'instance a démarré, son état devient `running` et elle reçoit un nom DNS public. (Si la colonne DNS public n'est pas affichée, choisissez l'Show/Hideicône, puis sélectionnez DNS public.)

Étape 4 : Créez une application dans CodeDeploy

Dans CodeDeploy, une application est une ressource qui contient l'application logicielle que vous souhaitez déployer. Plus tard, vous utiliserez cette application CodePipeline pour automatiser les déploiements de l'exemple d'application sur votre instance Amazon EC2.

Tout d'abord, vous devez créer un rôle qui permet d' CodeDeploy effectuer des déploiements. Ensuite, vous créez une application CodeDeploy .

Pour créer un rôle CodeDeploy de service

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le tableau de bord de la console, choisissez Rôles.
3. Choisissez Créer un rôle.
4. Sous Sélectionner une entité de confiance, choisissez Service AWS. Sous Use case (Cas d'utilisation), choisissez CodeDeploy. Choisissez CodeDeploy parmi les options répertoriées. Choisissez Suivant. La stratégie gérée `AWSCodeDeployRole` est déjà attachée au rôle.
5. Choisissez Suivant.
6. Entrez un nom pour le rôle (par exemple, **`CodeDeployRole`**), puis choisissez Créer un rôle.

Pour créer une application dans CodeDeploy

1. Ouvrez la CodeDeploy console à l'adresse https://console.aws.amazon.com/codedeploy.
2. Si la page Applications n'apparaît pas, dans le menu, choisissez Applications.
3. Choisissez Créer une application.
4. Dans Nom de l'application, saisissez **`MyDemoApplication`**.
5. Dans Compute Platform, sélectionnez EC2/On-premises.
6. Choisissez Créer une application.

Pour créer un groupe de déploiement dans CodeDeploy

Un groupe de déploiement est une ressource qui définit les paramètres liés au déploiement, tels que les instances sur lesquelles déployer et la vitesse du déploiement.

1. Sur la page qui affiche votre application, choisissez Créer un groupe de déploiement.
2. Dans Nom du groupe de déploiement, saisissez **`MyDemoDeploymentGroup`**.
3. Dans Rôle de service, choisissez l'ARN du rôle de service que vous avez créé précédemment (par exemple,**`arn:aws:iam::`*`account_ID`*`:role/CodeDeployRole`**).
4. Dans Type de déploiement, sélectionnez In-place.
5. Sous Configuration de l'environnement, choisissez Instances Amazon EC2. Dans le champ Clé, saisissez**`Name`**. Dans le champ Valeur, entrez le nom que vous avez utilisé pour baliser l'instance (par exemple,**`MyCodePipelineDemo`**).
6. Dans Configuration de l'agent avec AWS Systems Manager, choisissez Maintenant et planifiez les mises à jour. Cela permet d'installer l'agent sur l'instance. L'instance Linux est déjà configurée avec l'agent SSM et va maintenant être mise à jour avec l' CodeDeploy agent.
7. Dans Configuration de déploiement, choisissez `CodeDeployDefault.OneAtaTime`.
8. Dans Load Balancer, assurez-vous que l'option Activer l'équilibrage de charge n'est pas sélectionnée. Vous n'avez pas besoin de configurer un équilibreur de charge ou de choisir un groupe cible pour cet exemple.
9. Choisissez Créer un groupe de déploiement.

Étape 5 : Créez votre premier pipeline dans CodePipeline

Vous êtes maintenant prêt à créer et à exécuter votre premier pipeline. Au cours de cette étape, vous créez un pipeline qui s'exécute automatiquement lorsque le code est envoyé vers votre CodeCommit référentiel.

Pour créer un CodePipeline pipeline

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyFirstPipeline`**.
5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.
6. Dans Rôle de service, choisissez Nouveau rôle de service pour permettre CodePipeline de créer un rôle de service dans IAM.
7. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
8. À l'étape 3 : Ajouter une étape source, dans Source provider, choisissez CodeCommit. Dans Nom du référentiel, choisissez le nom du CodeCommit référentiel dans lequel vous avez créé Étape 1 : Création d'un CodeCommit référentiel. Dans Nom de branche, choisissez `main`, puis Étape suivante.

   Une fois que vous avez sélectionné le nom du référentiel et la branche, un message affiche la règle Amazon CloudWatch Events à créer pour ce pipeline.

   Sous Modifier les options de détection, ne modifiez pas les valeurs par défaut. Cela permet d' CodePipeline utiliser Amazon CloudWatch Events pour détecter les modifications apportées à votre référentiel source.

Choisissez Suivant.

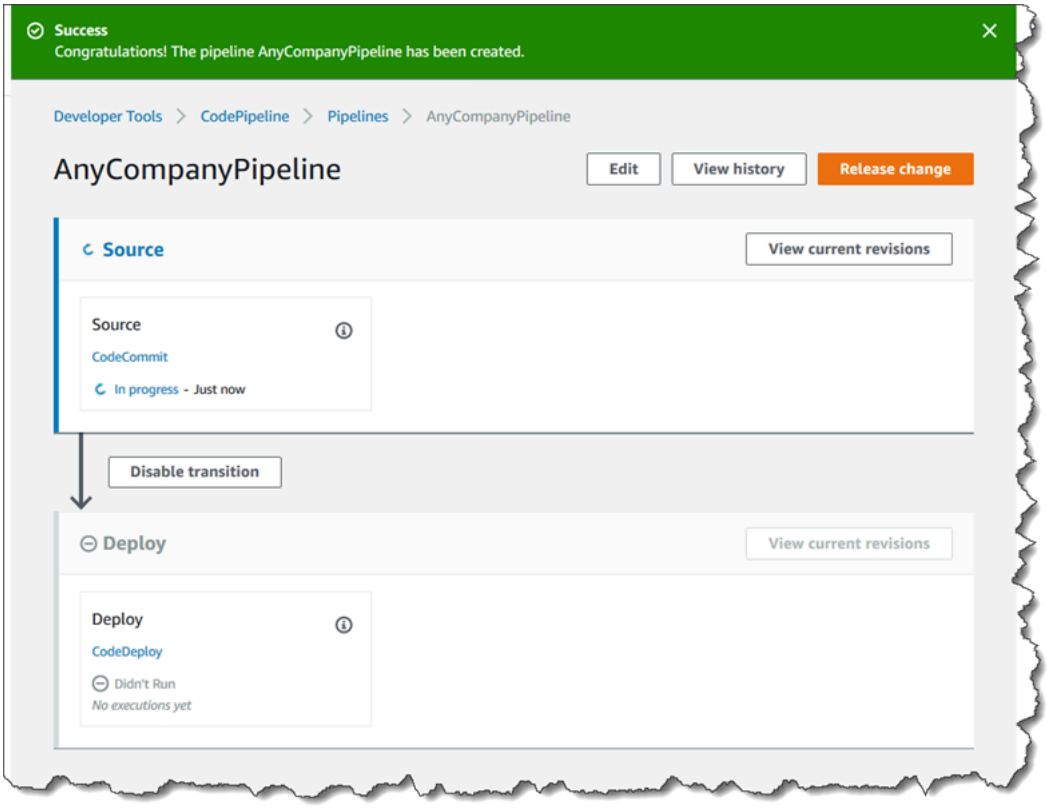
9. À l'étape 4 : Ajouter une étape de construction, choisissez Ignorer l'étape de génération, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer. Choisissez Suivant.

> **Note**
>
> Dans ce didacticiel, vous déployez un code qui ne requiert aucun service de génération. Vous pouvez donc ignorer cette étape. Toutefois, si votre code source doit être généré avant d'être déployé sur des instances, vous pouvez configurer CodeBuild au cours de cette étape.

10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.

11. À l'étape 6 : Ajouter une étape de déploiement, dans Deploy provider, choisissez CodeDeploy. Dans Nom de l'application, choisissez **`MyDemoApplication`**. Dans Groupe de déploiement, choisissez **`MyDemoDeploymentGroup`**, puis Étape suivante.
12. À l'étape 7 : Passez en revue, passez en revue les informations, puis choisissez Créer un pipeline.
13. Le pipeline commence à s'exécuter une fois qu'il a été créé. Il télécharge le code depuis votre CodeCommit référentiel et crée un CodeDeploy déploiement sur votre instance EC2. Vous pouvez consulter les messages de progression, de réussite et d'échec au fur et à mesure que l' CodePipeline exemple déploie la page Web sur l'instance Amazon EC2 du CodeDeploy déploiement.



Félicitations ! Vous venez de créer un pipeline simple dans CodePipeline.

Vérifiez ensuite les résultats.

Pour vérifier que votre pipeline a été exécuté avec succès

1. Affichez la progression initiale du pipeline. L'état de chaque étape passe de Pas encore d'exécution à En cours, puis devient soit Réussi, soit Échec. Le pipeline doit terminer la première exécution en quelques minutes.
2. Une fois que Succeeded est affiché pour l'état du pipeline, dans la zone d'état de la phase de déploiement, choisissez CodeDeploy. Cela ouvre la CodeDeploy console. Si le message Succeded n'est pas affiché, consultezRésolution des problèmes CodePipeline.
3. Dans l'onglet Déploiements, choisissez l'ID de déploiement. Sur la page du déploiement, sous Événements du cycle de vie de déploiement, choisissez l'ID de l'instance. Ceci ouvre la console EC2.
4. Dans l'onglet Description, sous DNS public, copiez l'adresse (par exemple, `ec2-192-0-2-1.us-west-2.compute.amazonaws.com`), puis collez-la dans la barre d'adresses de votre navigateur web.

La page Web s'affiche pour l'exemple d'application que vous avez téléchargé et transféré CodeCommit dans votre référentiel.

Pour plus d'informations sur les étapes, les actions et le fonctionnement des pipelines, consultez CodePipeline concepts.

Étape 6 : Modifier le code de votre CodeCommit référentiel

Votre pipeline est configuré de sorte à s'exécuter chaque fois que le code est modifié dans votre référentiel CodeCommit. Au cours de cette étape, vous apportez des modifications au fichier HTML qui fait partie de l'exemple d' CodeDeployapplication du CodeCommit référentiel. Lorsque vous transmettez ces modifications, votre pipeline s'exécute à nouveau et les modifications que vous apportez sont visibles à l'adresse web à laquelle vous avez accédé précédemment.

1. Passez dans le répertoire de votre référentiel local :

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

2. Utilisez un éditeur de texte pour modifier le fichier `index.html` :

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

3. Révisez le contenu du fichier `index.html` pour modifier la couleur d'arrière-plan et une partie du texte sur la page web, puis enregistrez le fichier.

```
<!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. Validez et transférez vos modifications CodeCommit dans votre référentiel en exécutant les commandes suivantes, une par une :

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

```
git push
```

Pour vérifier que votre pipeline a été exécuté avec succès

1. Affichez la progression initiale du pipeline. L'état de chaque étape passe de Pas encore d'exécution à En cours, puis devient soit Réussi, soit Échec. L'exécution du pipeline se terminer en quelques minutes.

2. Une fois que Réussi s'affiche pour l'état de l'action, actualisez la page de démonstration à laquelle vous avez accédé précédemment dans votre navigateur.

   La page Web mise à jour s'affiche.

Étape 7 : Nettoyer les ressources

Vous pouvez utiliser certaines des ressources que vous avez créées dans ce didacticiel pour d'autres didacticiels de ce guide. Par exemple, vous pouvez réutiliser l' CodeDeploy application et le déploiement. Toutefois, une fois ce didacticiel terminé (ou d'autres didacticiels), vous devez supprimer le pipeline, ainsi que les ressources qu'il utilise, afin que l'utilisation de ces ressources ne vous soit pas facturée. Supprimez d'abord le pipeline, puis l' CodeDeploy application et son instance Amazon EC2 associée, et enfin, le CodeCommit référentiel.

Pour nettoyer les ressources utilisées dans ce didacticiel

1. Pour nettoyer vos CodePipeline ressources, suivez les instructions de la section Supprimer un pipeline dans AWS CodePipeline.
2. Pour nettoyer vos CodeDeploy ressources, suivez les instructions de la section Ressources relatives à la procédure pas à pas pour nettoyer le déploiement.
3. Pour supprimer le CodeCommit référentiel, suivez les instructions de la section Supprimer un CodeCommit référentiel.

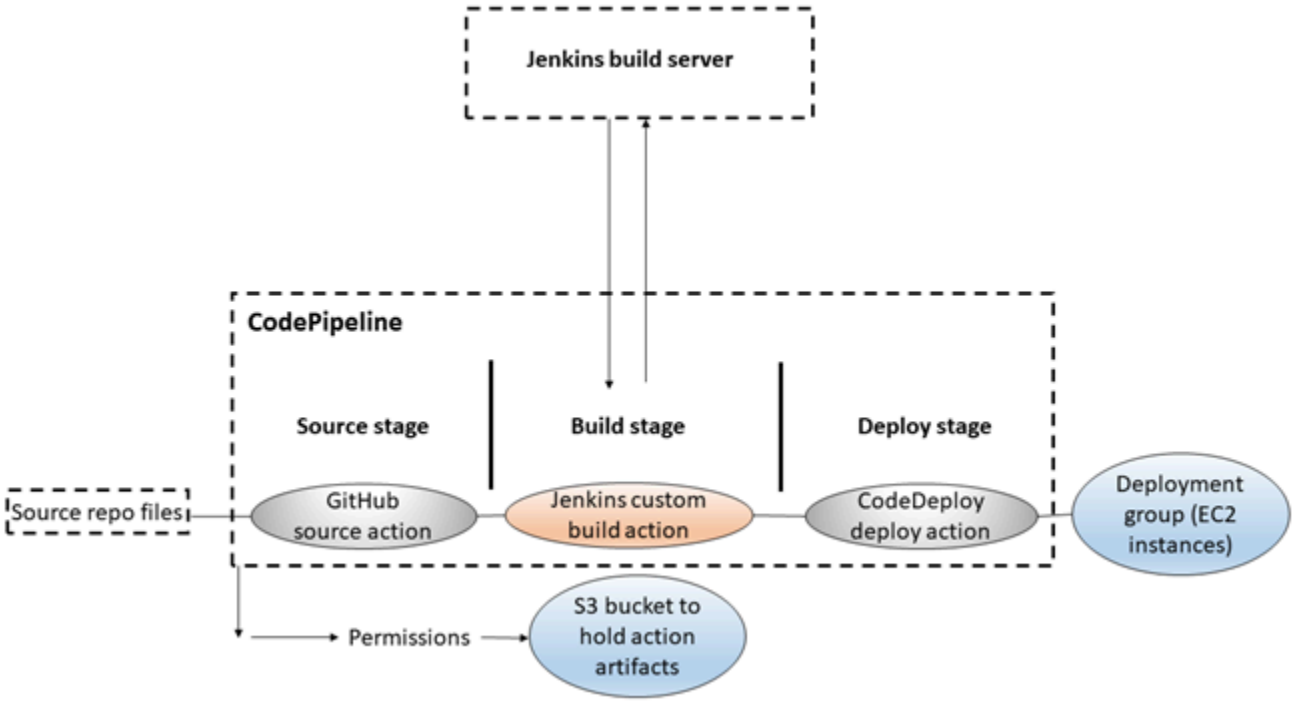
Étape 8 : Suggestions de lecture

En savoir plus sur le CodePipeline fonctionnement :

- Pour plus d'informations sur les étapes, les actions et le fonctionnement des pipelines, consultez CodePipeline concepts.
- Pour plus d'informations sur les actions que vous pouvez effectuer à l'aide de CodePipeline, consultezIntégrations avec les types CodePipeline d'action.
- Essayez ce didacticiel plus avancé, Didacticiel : Création d'un pipeline à quatre étapes. Celui-ci crée un pipeline en plusieurs étapes incluant une étape qui génère le code avant son déploiement.

Didacticiel : Création d'un pipeline à quatre étapes

Maintenant que vous avez créé votre premier pipeline dans Didacticiel : Création d'un pipeline simple (compartiment S3) ou Tutoriel : Création d'un pipeline simple (CodeCommit référentiel), vous pouvez commencer à créer des pipelines plus complexes. Ce didacticiel vous expliquera comment créer un pipeline en quatre étapes qui utilise un GitHub référentiel pour votre source, un serveur de compilation Jenkins pour créer le projet et une CodeDeploy application pour déployer le code généré sur un serveur intermédiaire. Le schéma suivant montre le pipeline initial en trois étapes.



Une fois le pipeline créé, vous allez le modifier pour ajouter une étape comprenant une action de test pour tester le code, également à l'aide de Jenkins.

Avant de pouvoir créer ce pipeline, vous devez configurer les ressources requises. Par exemple, si vous souhaitez utiliser un GitHub référentiel pour votre code source, vous devez créer le référentiel avant de pouvoir l'ajouter à un pipeline. Dans le cadre de la configuration, ce didacticiel vous guide dans la configuration de Jenkins sur une instance EC2, à des fins de démonstration.

> **Important**
>
> La plupart des actions que vous ajoutez à votre pipeline au cours de cette procédure impliquent AWS des ressources que vous devez créer avant de créer le pipeline. AWS les ressources pour vos actions source doivent toujours être créées dans la même AWS région

que celle dans laquelle vous créez votre pipeline. Par exemple, si vous créez votre pipeline dans la région USA Est (Ohio), votre CodeCommit référentiel doit se trouver dans la région USA Est (Ohio).
Vous pouvez ajouter des actions interrégionales lorsque vous créez votre pipeline. AWS les ressources pour les actions interrégionales doivent se trouver dans la même AWS région que celle où vous prévoyez d'exécuter l'action. Pour de plus amples informations, veuillez consulter Ajouter une action interrégionale dans CodePipeline.

⚠ Important

Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Avant de commencer ce didacticiel, assurez-vous de réunir les prérequis généraux indiqués dans Commencer avec CodePipeline.

Rubriques

- Étape 1 : Vérifier les conditions préalables
- Étape 2 : Créez un pipeline dans CodePipeline
- Étape 3 : Ajouter une autre étape à votre pipeline
- Étape 4 : Nettoyer les ressources

Étape 1 : Vérifier les conditions préalables

Pour intégrer Jenkins, vous devez AWS CodePipeline installer le CodePipeline plug-in pour Jenkins sur n'importe quelle instance de Jenkins que vous souhaitez utiliser. CodePipeline Vous devez également configurer un utilisateur ou un rôle IAM dédié à utiliser pour les autorisations entre votre projet Jenkins et. CodePipeline Le moyen le plus simple d'intégrer Jenkins CodePipeline est d'installer Jenkins sur une instance EC2 qui utilise un rôle d'instance IAM que vous créez pour l'intégration de Jenkins. Afin que les liens des actions Jenkins du pipeline soient bien connectés, vous devez configurer les paramètres du pare-feu et de proxy sur le serveur ou l'instance EC2 pour

autoriser les connexions entrantes sur le port utilisé par votre projet Jenkins. Assurez-vous d'avoir configuré Jenkins afin d'authentifier les utilisateurs et d'appliquer le contrôle d'accès avant d'autoriser les connexions sur ces ports (par exemple, 443 et 8443 si vous avez sécurisé Jenkins de sorte à ce qu'il n'utilise que des connexions HTTPS, ou 80 et 8080 si vous autorisez les connexions HTTP). Pour plus d'informations, consultez Sécurisation de Jenkins.

> **Note**
>
> Ce didacticiel utilise un modèle de code et configure les étapes de génération qui convertissent ce modèle, de Haml à HTML. Vous pouvez télécharger l'exemple de code open source depuis le GitHub référentiel en suivant les étapes décrites dansCopiez ou clonez l'échantillon dans un GitHub référentiel. Vous aurez besoin de l'intégralité de l'échantillon dans votre GitHub référentiel, et pas seulement du fichier .zip.
> Ce didacticiel présume également que :
>
> - Vous êtes familiarisé avec l'installation et la gestion de Jenkins, ainsi qu'avec la création de projets Jenkins.
> - Vous avez installé Rake et le gem Haml pour Ruby sur le même ordinateur ou la même instance qui héberge votre projet Jenkins.
> - Vous avez défini les variables requises de l'environnement système afin que les commandes Rake puissent être exécutées à partir du terminal ou de la ligne de commande (par exemple, sur les systèmes Windows, modifiez la variable PATH pour inclure le répertoire où vous avez installé Rake).

Rubriques

- Copiez ou clonez l'échantillon dans un GitHub référentiel
- Création d'un rôle IAM à utiliser pour l'intégration de Jenkins
- Installation et configuration de Jenkins et du CodePipeline plug-in pour Jenkins

Copiez ou clonez l'échantillon dans un GitHub référentiel

Pour cloner l'échantillon et le transférer vers un GitHub référentiel

1. Téléchargez l'exemple de code depuis le GitHub référentiel ou clonez les référentiels sur votre ordinateur local. Il y a deux modèles de modules :

- Si vous comptez déployer votre exemple sur des instances Amazon Linux, RHEL ou Ubuntu Server, choisissez codepipeline-jenkins-aws-codedeploy_linux.zip.
- Si vous comptez déployer votre exemple sur des instances Windows Server, choisissez CodePipeline-Jenkins-AWSCodeDeploy _ Windows.zip.

2. Dans le référentiel, choisissez Fork pour cloner l'exemple de dépôt dans un dépôt de votre GitHub compte. Pour plus d'informations, consultez la documentation GitHub.

Création d'un rôle IAM à utiliser pour l'intégration de Jenkins

Il est recommandé d'envisager de lancer une instance EC2 pour héberger votre serveur Jenkins et d'utiliser un rôle IAM pour accorder à l'instance les autorisations requises pour interagir avec celle-ci. CodePipeline

1. Connectez-vous au Console de gestion AWS et ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le panneau de navigation de la console IAM, choisissez Rôles, puis Créer un rôle.
3. Sous Select type of trusted entity (Sélectionner le type d'entité approuvée), choisissez Service AWS. Sous Choose the service that will use this role (Choisir le service qui utilisera ce rôle), choisissez EC2. Sous Select your use case (Sélectionner votre cas d'utilisation), choisissez EC2.
4. Choisissez Suivant : Autorisations. Sur la page Attach permissions policies (Attacher des stratégies d'autorisations), sélectionnez la stratégie gérée `AWSCodePipelineCustomActionAccess`, puis choisissez Next: Tags (Suivant : balises). Choisissez Suivant : Vérification.
5. Dans la page Vérification, dans Nom du rôle, entrez le nom du rôle à créer spécifiquement pour l'intégration Jenkins (par exemple, *`JenkinsAccess`*), puis choisissez Créer un rôle.

Lorsque vous créez l'instance EC2 où vous allez installer Jenkins, dans Étape 3 : Configurer les détails de l'instance, assurez-vous de choisir le rôle de l'instance (par exemple, *`JenkinsAccess`*).

Pour plus d'informations sur les rôles d'instance et Amazon EC2, consultez les sections Rôles IAM pour Amazon EC2, Utilisation des rôles IAM pour accorder des autorisations aux applications exécutées sur des instances Amazon EC2 et Création d'un rôle pour déléguer des autorisations à un. Service AWS

Installation et configuration de Jenkins et du CodePipeline plug-in pour Jenkins

Pour installer Jenkins et le CodePipeline plug-in pour Jenkins

1. Créez une instance EC2 où vous allez installer Jenkins, et dans Étape 3 : Configurer les détails de l'instance, assurez-vous de choisir le rôle de l'instance que vous avez créée (par exemple, *JenkinsAccess*). Pour plus d'informations sur la création d'instances EC2, consultez la section Lancer une instance Amazon EC2 dans le guide de l'utilisateur Amazon EC2.

   > **Note**
   >
   > Si vous souhaitez déjà utiliser des ressources Jenkins, vous pouvez le faire, mais vous devez créer un utilisateur IAM spécial, appliquer la politique `AWSCodePipelineCustomActionAccess` gérée à cet utilisateur, puis configurer et utiliser les informations d'accès de cet utilisateur sur votre ressource Jenkins. Si vous souhaitez que l'interface utilisateur Jenkins fournisse les informations d'identification, configurez Jenkins de sorte à uniquement autoriser les connexions HTTPS. Pour de plus amples informations, veuillez consulter Résolution des problèmes CodePipeline.

2. Installez Jenkins sur l'instance EC2. Pour plus d'informations, consultez la documentation de Jenkins relative à l'installation de Jenkins et à la mise en route et à l'accès à Jenkins, ainsi que details of integration with Jenkins dans Intégrations de produits et de services avec CodePipeline.
3. Lancez Jenkins et sur la page d'accueil, choisissez Manage Jenkins.
4. Sur la page Manage Jenkins, choisissez Manage Plugins.
5. Choisissez l'onglet Disponible et, dans la zone de recherche Filtre, entrez **`AWS CodePipeline`**. Choisissez CodePipeline Plugin pour Jenkins dans la liste, puis choisissez Télécharger maintenant et installer après le redémarrage.
6. Sur la Plugins/Upgrades page Installation, sélectionnez Redémarrer Jenkins lorsque l'installation est terminée et qu'aucune tâche n'est en cours d'exécution.
7. Choisissez Back to Dashboard.
8. Sur la page principale, choisissez New Item.
9. Dans Nom de l'article, entrez un nom pour le projet Jenkins (par exemple,*MyDemoProject*). Choisissez Freestyle project, puis OK.

> **Note**
>
> Assurez-vous que le nom de votre projet répond aux exigences relatives à CodePipeline. Pour de plus amples informations, veuillez consulter Quotas dans AWS CodePipeline.

10. Sur la page de configuration du projet, cochez la case Execute concurrent builds if necessary. Dans Gestion du code source, choisissez AWS CodePipeline. Si vous avez installé Jenkins sur une instance EC2 et que vous l'avez configuré AWS CLI avec le profil de l'utilisateur IAM que vous avez créé pour l'intégration entre CodePipeline et Jenkins, laissez tous les autres champs vides.
11. Choisissez Avancé, puis dans Fournisseur, entrez le nom du fournisseur de l'action tel qu'il apparaîtra dans CodePipeline (par exemple,*MyJenkinsProviderName*). Assurez-vous que ce nom est unique et facile à retenir. Vous l'utiliserez lorsque vous ajouterez une action de génération à votre pipeline un peu plus tard dans ce didacticiel, et à nouveau lorsque vous ajouterez une action de test.

> **Note**
>
> Le nom de cette action doit respecter les exigences d'attribution de noms pour les actions dans CodePipeline. Pour de plus amples informations, veuillez consulter Quotas dans AWS CodePipeline.

12. Dans Build Triggers, décochez toutes les cases et sélectionnez Poll SCM. Dans Schedule, tapez cinq astérisques séparés par des espaces, comme suit :

```
* * * * *
```

Ce sondage se CodePipeline fait chaque minute.

13. Dans Build, choisissez Add build step. Choisissez Execute shell (Amazon Linux, RHEL ou Ubuntu Server) Exécuter la commande batch (Windows Server), puis saisissez les informations suivantes :

```
rake
```

> **Note**
>
> Assurez-vous que votre environnement est configuré avec les variables et les paramètres requis pour exécuter Rake ; dans le cas contraire, la génération échouera.

14. Choisissez Ajouter une action après la création, puis choisissez AWS CodePipeline Publisher. Choisissez Add et dans Build Output Locations, laissez l'emplacement vide. Il s'agit de la configuration par défaut. Celle-ci crée un fichier compressé à la fin du processus de génération.
15. Choisissez Save pour enregistrer votre projet Jenkins.

Étape 2 : Créez un pipeline dans CodePipeline

Dans cette partie du didacticiel, vous créez le pipeline à l'aide de l'assistant Créer un pipeline.

Pour créer un processus de publication CodePipeline automatique

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Si nécessaire, utilisez le sélecteur de région pour définir la même région que celle où se trouvent vos ressources de pipeline. Par exemple, si vous avez créé des ressources pour le didacticiel précédent dans`us-east-2`, assurez-vous que le sélecteur de région est réglé sur USA Est (Ohio).

   Pour plus d'informations sur les régions et les points de terminaison disponibles CodePipeline, consultez la section AWS CodePipeline Points de terminaison et quotas.
3. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
4. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
5. Sur la page Étape 2 : Choisissez les paramètres du pipeline, dans Nom du pipeline, entrez le nom de votre pipeline.
6. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.

7. Dans Rôle de service, choisissez Nouveau rôle de service pour permettre CodePipeline de créer un rôle de service dans IAM.
8. Conservez les valeurs par défaut sous Advanced settings (Paramètres avancés), puis choisissez Suivant.
9. Sur la page Étape 3 : Ajouter une étape source, dans Fournisseur source, choisissez GitHub.
10. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.
11. À l'étape 4 : Ajouter une étape de construction, choisissez Ajouter Jenkins. Dans Nom du fournisseur, entrez le nom de l'action que vous avez fournie dans le CodePipeline plug-in pour Jenkins (par exemple*`MyJenkinsProviderName`*). Ce nom doit correspondre exactement au nom indiqué dans le CodePipeline plugin pour Jenkins. Dans URL du serveur, entrez l'URL de l'instance EC2 où Jenkins est installé. Dans Nom du projet, entrez le nom du projet que vous avez créé dans Jenkins, par exemple*`MyDemoProject`*, puis choisissez Suivant.
12. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.
13. À l'étape 6 : ajouter une phase de déploiement, réutilisez l' CodeDeploy application et le groupe de déploiement que vous avez créés dansDidacticiel : Création d'un pipeline simple (compartiment S3). Dans Fournisseur de déploiement, choisissez CodeDeploy. Dans Nom de l'application, entrez **`CodePipelineDemoApplication`**, ou choisissez le bouton d'actualisation et choisissez le nom de l'application dans la liste. Dans Groupe de déploiement, entrez **`CodePipelineDemoFleet`** ou choisissez-le dans la liste, puis choisissez Suivant.

    > **Note**
    >
    > Vous pouvez utiliser vos propres CodeDeploy ressources ou en créer de nouvelles, mais cela peut entraîner des coûts supplémentaires.
14. À l'étape 7 : Passez en revue, passez en revue les informations, puis choisissez Créer un pipeline.
15. Le pipeline démarre automatiquement et exécute le modèle à travers le pipeline. Vous pouvez consulter les messages de progression, de réussite et d'échec au fur et à mesure que le pipeline crée l'exemple Haml au format HTML et le déploie sous forme de page Web sur chacune des instances Amazon EC2 du déploiement. CodeDeploy

Étape 3 : Ajouter une autre étape à votre pipeline

Vous allez maintenant ajouter une étape de test, puis une action de test à cette étape, laquelle utilise le test Jenkins inclus dans le modèle pour déterminer si la page web possède un contenu. Ce test est donné uniquement à des fins de démonstration.

> **Note**
>
> Si vous ne souhaitez pas ajouter une autre étape à votre pipeline, vous pouvez ajouter une action de test à l'étape intermédiaire de celui-ci, avant ou après l'action de déploiement.

Ajout d'une étape de test à votre pipeline

Rubriques

- Recherche de l'adresse IP d'une instance
- Création d'un projet Jenkins pour tester le déploiement
- Création d'une quatrième étape

Recherche de l'adresse IP d'une instance

Pour vérifier l'adresse IP d'une instance où vous avez déployé votre code

1. Lorsque l'état du pipeline affiche Succeeded dans la zone d'état de l'étape Intermédiaire, choisissez Details.
2. Dans la section Deployment Details, sous Instance ID, choisissez l'identifiant d'instance de l'une des instances déployées avec succès.
3. Copiez l'adresse IP de l'instance (par exemple,`192.168.0.4`). Vous utiliserez cette adresse IP dans votre test Jenkins.

Création d'un projet Jenkins pour tester le déploiement

Création d'un projet Jenkins

1. Dans l'instance où vous avez installé Jenkins, ouvrez Jenkins et choisissez New Item sur la page d'accueil.

2. Dans Nom de l'article, entrez un nom pour le projet Jenkins (par exemple,*MyTestProject*). Choisissez Freestyle project, puis OK.

   > **Note**
   >
   > Assurez-vous que le nom de votre projet répond aux CodePipeline exigences. Pour de plus amples informations, veuillez consulter Quotas dans AWS CodePipeline.

3. Sur la page de configuration du projet, cochez la case Execute concurrent builds if necessary. Dans Gestion du code source, choisissez AWS CodePipeline. Si vous avez installé Jenkins sur une instance EC2 et que vous l'avez configuré AWS CLI avec le profil de l'utilisateur IAM que vous avez créé pour l'intégration entre CodePipeline et Jenkins, laissez tous les autres champs vides.

   > **Important**
   >
   > Si vous configurez un projet Jenkins et qu'il n'est pas installé sur une instance Amazon EC2, ou s'il est installé sur une instance EC2 qui exécute un système d'exploitation Windows, complétez les champs conformément aux paramètres de votre hôte proxy et de votre port, et fournissez les informations d'identification de l'utilisateur ou du rôle IAM que vous avez configuré pour l'intégration entre Jenkins et. CodePipeline

4. Choisissez Advanced et dans Category, choisissez Test.
5. Dans Provider, entrez le même nom que celui que vous avez utilisé pour le projet de génération (par exemple,*MyJenkinsProviderName*). Vous utiliserez ce nom lorsque vous ajouterez l'action de test à votre pipeline un peu plus tard dans ce didacticiel.

   > **Note**
   >
   > Ce nom doit répondre aux exigences de CodePipeline dénomination des actions. Pour de plus amples informations, veuillez consulter Quotas dans AWS CodePipeline.

6. Dans Build Triggers, décochez toutes les cases et sélectionnez Poll SCM. Dans Schedule, tapez cinq astérisques séparés par des espaces, comme suit :

   ```
   * * * * *
   ```

   Ce sondage se CodePipeline fait chaque minute.

7. Dans Build, choisissez Add build step. Si vous effectuez un déploiement sur des instances Amazon Linux, RHEL ou Ubuntu Server, choisissez Execute shell. Entrez ensuite ce qui suit, où l'adresse IP est l'adresse de l'instance EC2 que vous avez copiée précédemment :

   ```
   TEST_IP_ADDRESS=192.168.0.4 rake test
   ```

   Si vous effectuez un déploiement sur des instances Windows Server, choisissez Execute batch command, puis entrez ce qui suit, où l'adresse IP est l'adresse de l'instance EC2 que vous avez copiée précédemment :

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

   > **Note**
   >
   > Ce test suppose la présence d'un port 80 par défaut. Si vous souhaitez spécifier un port différent, ajoutez une instruction de port de test, comme suit :
   >
   > ```
   > TEST_IP_ADDRESS=192.168.0.4 TEST_PORT=8000 rake test
   > ```

8. Choisissez Ajouter une action après la création, puis choisissez AWS CodePipeline Publisher. Ne choisissez pas Add.
9. Choisissez Save pour enregistrer votre projet Jenkins.

Création d'une quatrième étape

Pour ajouter une étape à votre pipeline qui inclut l'action de test Jenkins

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Dans Nom, choisissez le nom du pipeline que vous avez créé, MySecondPipeline.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Sur la page Edit (Modifier), choisissez + Stage (+ étape) pour ajouter une étape immédiatement après l'étape de génération.
5. Dans le champ du nom de la nouvelle étape, entrez un nom (par exemple, **`Testing`**), puis choisissez + Add action group (+ Ajouter un groupe d'actions).

6. Dans Nom de l'action, entrez*MyJenkinsTest-Action*. Dans Fournisseur de test, choisissez le nom du fournisseur que vous avez spécifié dans Jenkins (par exemple,*MyJenkinsProviderName*). Dans Nom du projet, entrez le nom du projet que vous avez créé dans Jenkins (par exemple,*MyTestProject*). Dans Artefacts d'entrée, choisissez l'artefact de la version Jenkins dont le nom par défaut est*BuildArtifact*, puis choisissez Terminé.

   > **Note**
   >
   > Comme l'action de test Jenkins fonctionne sur l'application générée dans l'étape de génération Jenkins, utilisez l'artefact de génération pour l'artefact d'entrée vers l'action de test.

   Pour en savoir plus sur les artéfacts d'entrée et de sortie et sur la structure des pipelines, consultez CodePipeline référence de la structure du pipeline.

7. Sur la page Edit choisissez Save pipeline changes. Dans la boîte de dialogue Save pipeline changes, choisissez Save and continue.

8. Bien que la nouvelle étape a été ajoutée à votre pipeline, son état affiche No executions yet pour cette étape, car aucune modification n'a déclenché une autre exécution du pipeline. Pour exécuter l'exemple dans le pipeline révisé, sur la page des détails du pipeline, choisissez Release change.

   Le pipeline affiche ses étapes et ses actions, ainsi que l'état de la révision en cours d'exécution à travers ces quatre étapes. Le temps nécessaire à l'exécution du pipeline dans toutes les étapes dépendra de la taille des artéfacts, de la complexité de votre génération et des actions de test, en plus d'autres facteurs.

Étape 4 : Nettoyer les ressources

Une fois que vous avez terminé ce didacticiel, supprimez le pipeline, ainsi que les ressources qu'il utilise, afin d'éviter d'être facturé pour leur utilisation. Si vous n'avez pas l'intention de continuer à utiliser CodePipeline, supprimez le pipeline, puis l' CodeDeploy application et ses instances Amazon EC2 associées, et enfin le compartiment Amazon S3 utilisé pour stocker les artefacts. Vous devriez également envisager de supprimer d'autres ressources, telles que le GitHub référentiel, si vous n'avez pas l'intention de continuer à les utiliser.

Pour nettoyer les ressources utilisées dans ce didacticiel

1. Ouvrez une session de terminal sur votre machine Linux, macOS ou Unix locale, ou une invite de commande sur votre machine Windows locale, puis exécutez la delete-pipeline commande pour supprimer le pipeline que vous avez créé. Pour **`MySecondPipeline`**, entrez la commande suivante :

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

   Cette commande ne donne aucun résultat.
2. Pour nettoyer vos CodeDeploy ressources, suivez les instructions de la section Nettoyage.
3. Pour nettoyer les ressources de votre instance, supprimez l'instance EC2 où vous avez installé Jenkins. Pour plus d'informations, consultez Nettoyez votre instance.
4. Si vous n'avez pas l'intention de créer d'autres pipelines ou de les CodePipeline réutiliser, supprimez le compartiment Amazon S3 utilisé pour stocker les artefacts de votre pipeline. Pour supprimer le compartiment, veuillez suivre les instructions contenues dans Suppression d'un compartiment.
5. Si vous ne souhaitez pas réutiliser les autres ressources pour ce pipeline, pensez à les supprimer en suivant les instructions propres à chacune d'elles. Par exemple, si vous souhaitez supprimer le GitHub référentiel, suivez les instructions de la section Supprimer un référentiel sur le GitHub site Web.

Tutoriel : Configuration d'une règle d' CloudWatch événements pour recevoir des notifications par e-mail en cas de modification de l'état du pipeline

Après avoir configuré un pipeline dans AWS CodePipeline, vous pouvez configurer une règle d' CloudWatch événements pour envoyer des notifications chaque fois que des modifications sont apportées à l'état d'exécution de vos pipelines, ou au cours des étapes ou des actions de vos pipelines. Pour plus d'informations sur l'utilisation CloudWatch des événements pour configurer des notifications en cas de modification de l'état du pipeline, consultezSurveillance des CodePipeline événements.

Dans ce didacticiel, vous configurez une notification pour envoyer un e-mail lorsque l'état d'un pipeline passe à ÉCHEC. Ce didacticiel utilise une méthode de transformateur d'entrée lors de la

création de la règle CloudWatch Events. Elle transforme les détails de schéma du message pour diffuser le message en texte lisible.

> **Note**
>
> Lorsque vous créez les ressources pour ce didacticiel, telles que la notification Amazon SNS et la règle CloudWatch Events, assurez-vous que les ressources sont créées dans la même AWS région que votre pipeline.

Rubriques

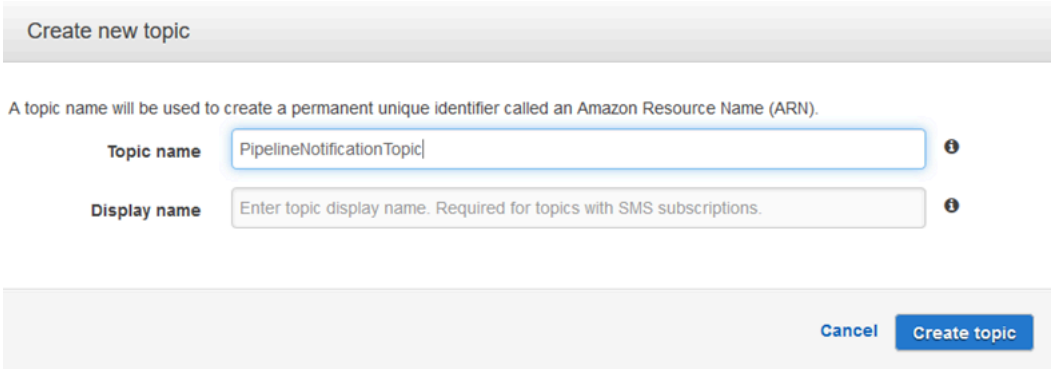
- Étape 1 : configurer une notification par e-mail à l'aide d'Amazon SNS
- Étape 2 : Créer une règle et ajouter la rubrique SNS en tant que cible
- Étape 3 : Nettoyer les ressources

Étape 1 : configurer une notification par e-mail à l'aide d'Amazon SNS

Amazon SNS coordonne l'utilisation des rubriques pour envoyer des messages aux terminaux ou aux clients abonnés. Utilisez Amazon SNS pour créer une rubrique de notification, puis abonnez-vous à cette rubrique à l'aide de votre adresse e-mail. La rubrique Amazon SNS sera ajoutée en tant que cible à votre règle relative aux CloudWatch événements. Pour de plus amples informations, consultez dans le Guide du développeur Amazon Simple Notification Service.

Créez ou identifiez un sujet dans Amazon SNS. CodePipeline utilisera CloudWatch Events pour envoyer des notifications sur cette rubrique via Amazon SNS. Pour créer une rubrique:

1. Ouvrez la console Amazon SNS à l'adresse https://console.aws.amazon.com/sns.
2. Choisissez Créer une rubrique.
3. Dans la boîte de dialogue Créer une rubrique, pour Nom de rubrique, saisissez un nom de rubrique (par exemple **`PipelineNotificationTopic`**).



4. Choisissez Créer une rubrique.

   Pour plus d'informations, consultez la section Créer une rubrique dans le guide du développeur Amazon SNS.

Abonnez un ou plusieurs destinataires à la rubrique pour recevoir des notifications par e-mail. Pour abonner un destinataire à une rubrique :

1. Dans la console Amazon SNS, dans la liste des sujets, cochez la case à côté de votre nouveau sujet. Choisissez Actions, s'abonner à la rubrique.
2. Dans la boîte de dialogue Créer un abonnement, vérifiez que l'ARN s'affiche dans ARN de la rubrique.
3. Pour Protocole, choisissez E-mail.
4. Indiquez l'adresse e-mail complète du destinataire dans Point de terminaison.
5. Choisissez Create Subscription (Créer un abonnement).
6. Amazon SNS envoie un e-mail de confirmation d'abonnement au destinataire. Pour recevoir des notifications par e-mail, un destinataire doit cliquer sur le lien Confirmer l'abonnement dans l'e-mail de confirmation. Une fois que le destinataire a cliqué sur le lien, s'il s'est inscrit avec succès, Amazon SNS affiche un message de confirmation dans le navigateur Web du destinataire.

   Pour plus d'informations, consultez la section S'abonner à une rubrique dans le manuel Amazon SNS Developer Guide.

Étape 2 : Créer une règle et ajouter la rubrique SNS en tant que cible

Créez une règle de notification d' CloudWatch événements avec CodePipeline comme source d'événement.

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le panneau de navigation, sélectionnez Events.
3. Choisissez Créer une règle. Sous Source d'événement, choisissez AWS CodePipeline. Pour Type d'événement, choisissez Changement d'état d'exécution du pipeline.
4. Choisissez État(s) spécifique(s), puis **`FAILED`**.
5. Cliquez sur Modifier pour ouvrir l'éditeur JSON pour le volet Aperçu du modèle d'événement. Ajoutez le paramètre **`pipeline`** avec le nom de votre pipeline comme indiqué dans l'exemple suivant pour un pipeline nommé « myPipeline. »

   Vous pouvez copier le modèle d'événement ici et le coller dans la console :

   ```
   {
     "source": [
       "aws.codepipeline"
     ],
     "detail-type": [
       "CodePipeline Pipeline Execution State Change"
     ],
     "detail": {
       "state": [
         "FAILED"
       ],
       "pipeline": [
         "myPipeline"
       ]
     }
   }
   ```

6. Pour Targets (Cibles), choisissez Add target (Ajouter une cible).
7. Dans la liste des cibles, sélectionnez Rubrique SNS. Pour Rubrique, saisissez la rubrique que vous avez créée.
8. Développez Configurer l'entrée, puis choisissez Transformateur d'entrée.
9. Dans la case Chemin d'entrée, saisissez les paires clé-valeur suivantes.

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

   Dans la case Modèle d'entrée, saisissez ce qui suit :

```
"The Pipeline <pipeline> has failed."
```

10. Choisissez Configure details (Configurer les détails).
11. Sur la page Configure rule details, saisissez un nom et une description facultative. Pour État, gardez la case Activé sélectionnée.
12. Choisissez Créer une règle.
13. Confirmez CodePipeline que les notifications de build sont désormais envoyées. Par exemple, vérifiez si les e-mails de notification de build sont à présent dans votre boîte de réception.
14. Pour modifier le comportement d'une règle, dans la CloudWatch console, choisissez la règle, puis choisissez Actions, Modifier. Modifiez la règle, choisissez Configurer les détails, puis choisissez Mettre à jour la règle.

    Pour arrêter d'utiliser une règle pour envoyer des notifications de build, dans la CloudWatch console, choisissez la règle, puis choisissez Actions, Désactiver.

    Pour supprimer une règle, dans la CloudWatch console, choisissez la règle, puis choisissez Actions, Supprimer.

Étape 3 : Nettoyer les ressources

Une fois que vous avez terminé ce didacticiel, supprimez le pipeline, ainsi que les ressources qu'il utilise, afin d'éviter d'être facturé pour leur utilisation.

Pour plus d'informations sur la façon de nettoyer la notification SNS et de supprimer la règle Amazon CloudWatch Events, consultez la section Nettoyage (désabonnement d'une rubrique Amazon SNS) et consultez la référence `DeleteRule` de l'API Amazon CloudWatch Events.

Tutoriel : Créez un pipeline qui crée et teste votre application Android avec AWS Device Farm

Vous pouvez l'utiliser AWS CodePipeline pour configurer un flux d'intégration continue dans lequel votre application est créée et testée chaque fois qu'un commit est envoyé. Ce didacticiel explique comment créer et configurer un pipeline pour créer et tester votre application Android avec le code source d'un GitHub référentiel. Le pipeline détecte l'arrivée d'un nouveau GitHub commit, puis l'utilise CodeBuildpour créer l'application et Device Farm pour la tester.

> **Important**
>
> Dans le cadre de la création d'un pipeline dans la console, un compartiment d'artefacts S3 sera utilisé par CodePipeline pour les artefacts. (Ceci est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une personne sûre et fiable.

> **Important**
>
> La plupart des actions que vous ajoutez à votre pipeline dans le cadre de cette procédure impliquent AWS des ressources que vous devez créer avant de créer le pipeline. AWS les ressources pour vos actions source doivent toujours être créées dans la même AWS région que celle où vous créez votre pipeline. Par exemple, si vous créez votre pipeline dans la région USA Est (Ohio), votre CodeCommit référentiel doit se trouver dans la région USA Est (Ohio).
>
> Vous pouvez ajouter des actions interrégionales lorsque vous créez votre pipeline. AWS les ressources pour les actions interrégionales doivent se trouver dans la même AWS région que celle où vous prévoyez d'exécuter l'action. Pour de plus amples informations, veuillez consulter Ajouter une action interrégionale dans CodePipeline.

Vous pouvez l'essayer en utilisant votre application Android existante et vos définitions de test, ou vous pouvez utiliser l'exemple d'application et les définitions de test fournis par Device Farm.

Avant de commencer

1. Connectez-vous à la AWS Device Farm console et choisissez Créer un nouveau projet.
2. Choisissez votre projet. Dans le navigateur, copiez l'URL de votre nouveau projet. L'URL contient l'ID de projet.
3. Copiez et conservez cet ID de projet. Vous l'utiliserez lorsque vous créerez votre pipeline dans CodePipeline.

   Voici un exemple d'URL de projet. Pour extraire l'ID de projet, copiez la valeur figurant après `projects/`. Dans cet exemple, l'ID de projet est `eec4905f-98f8-40aa-9afc-4c1cfexample`.

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

Configurez CodePipeline pour utiliser vos tests Device Farm

1. Ajoutez et validez un fichier appelé `buildspec.yml` à la racine du code de votre application, puis envoyez-le dans votre référentiel. CodeBuild utilise ce fichier pour exécuter des commandes et accéder aux artefacts nécessaires à la création de votre application.

   ```
   version: 0.2

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

2. (Facultatif) Si vous utilisez Calabash ou Appium pour tester votre application, ajoutez le fichier de définition de test à votre référentiel. Dans une étape ultérieure, vous pourrez configurer Device Farm pour utiliser les définitions afin de réaliser votre suite de tests.

   Si vous utilisez les tests intégrés à Device Farm, vous pouvez ignorer cette étape.

3. Pour créer votre pipeline et ajouter une étape source, procédez comme suit :

   a. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse https://console.aws.amazon.com/codepipeline/.

   b. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.

   c. Sur la page Étape 1 : Choisir une option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.

   d. Sur la page Étape 2 : Choisir les paramètres du pipeline, dans Nom du pipeline, entrez le nom de votre pipeline.

e. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul type que vous pouvez choisir dans la console. Pour plus d'informations, consultez la section Types de pipelines. Pour plus d'informations sur la tarification CodePipeline, consultez la section Tarification.

f. Dans Rôle de service, laissez Nouveau rôle de service sélectionné et laissez Nom du rôle inchangé. Vous pouvez également choisir d'utiliser un rôle de service si vous en avez déjà un.

> **Note**
>
> Si vous utilisez un rôle de CodePipeline service créé avant juillet 2018, vous devez ajouter des autorisations pour Device Farm. Pour ce faire, ouvrez la console IAM, recherchez le rôle, puis ajoutez les autorisations suivantes à la politique du rôle. Pour de plus amples informations, veuillez consulter Ajouter des autorisations au rôle CodePipeline de service.
>
> ```
> {
>      "Effect": "Allow",
>      "Action": [
>         "devicefarm:ListProjects",
>         "devicefarm:ListDevicePools",
>         "devicefarm:GetRun",
>         "devicefarm:GetUpload",
>         "devicefarm:CreateUpload",
>         "devicefarm:ScheduleRun"
>      ],
>      "Resource": "*"
> }
> ```

g. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.

h. Sur la page Étape 3 : Ajouter une étape source, dans Source provider, sélectionnez GitHub (via GitHub App).

i. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.

j. Dans Référentiel, choisissez le référentiel source.

k. Dans Branche, choisissez la branche que vous souhaitez utiliser.

l. Conservez les valeurs par défaut restantes pour l'action source. Choisissez Suivant.

4. À l'étape 4 : Ajouter une phase de construction, ajoutez une étape de construction :

   a. Dans Fournisseur de build, choisissez Autres fournisseurs de build, puis choisissez AWS CodeBuild. Acceptez la région du pipeline comme Région par défaut.

   b. Sélectionnez Create a project (Créer un projet).

   c. Dans Nom du projet, saisissez un nom pour ce projet de génération.

   d. Dans le champ Image d'environnement, choisissez Image gérée. Pour Système d'exploitation, choisissez Ubuntu.

   e. Pour Runtime (Exécution), sélectionnez Standard. Pour Image, choisissez aws/codebuild/standard : 5.0.

      CodeBuild utilise cette image du système d'exploitation, sur laquelle Android Studio est installé, pour créer votre application.

   f. Pour Rôle de service, choisissez votre rôle CodeBuild de service existant ou créez-en un nouveau.

   g. Pour Build specifications (Spécifications de génération), choisissez Use a buildspec file (Utiliser un fichier buildspec).

   h. Choisissez Continuer vers CodePipeline. Cela revient à la CodePipeline console et crée un CodeBuild projet qui utilise le `buildspec.yml` dans votre référentiel pour la configuration. Le projet de construction utilise un rôle de service pour gérer les Service AWS autorisations. Cette étape peut prendre quelques minutes.

   i. Choisissez Suivant.

5. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer l'étape de test, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer.

   Choisissez Suivant.

6. Sur la page Étape 6 : Ajouter une phase de déploiement, choisissez Ignorer l'étape de déploiement, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer. Choisissez Suivant.

7. À l'étape 7 : Révision, choisissez Créer un pipeline. Vous devez voir un diagramme montrant les étapes source et de génération.

8. Ajoutez une action de test Device Farm à votre pipeline :

   a. Dans le coin supérieur droit, choisissez Modifier.

b. Au bas du diagramme, choisissez + Ajouter une étape. Dans le champ Nom de l'étape, saisissez un nom, tel que **`Test`**.

c. Choisissez + Ajouter un groupe d'actions.

d. Dans Nom de l'action, entrez un nom.

e. Dans Action provider, choisissez AWS Device Farm. Acceptez la région du pipeline comme Région par défaut.

f. Dans Artefacts d'entrée, choisissez l'artefact d'entrée correspondant à l'artefact de sortie de l'étape qui précède l'étape de test, comme `BuildArtifact`.

   Dans la AWS CodePipeline console, vous pouvez trouver le nom de l'artefact de sortie pour chaque étape en survolant l'icône d'information dans le diagramme du pipeline. Si votre pipeline teste votre application directement depuis l'étape Source, choisissez SourceArtifact. Si le pipeline inclut une phase de construction, choisissez BuildArtifact.

g. Entrez ProjectIdl'identifiant de votre projet Device Farm. Suivez les étapes indiquées au début de ce didacticiel pour récupérer votre ID de projet.

h. Dans DevicePoolArn, entrez l'ARN du pool de périphériques. Pour obtenir le pool de périphériques disponible ARNs pour le projet, y compris l'ARN des meilleurs appareils, utilisez la AWS CLI pour entrer la commande suivante :

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

i. Dans AppType, entrez Android.

   Voici une liste de valeurs valides pour AppType:

   - iOS
   - Android
   - Web

j. Dans App, saisissez le chemin d'accès au package de l'application compilée. Ce chemin d'accès se rapporte à la racine de l'artefact d'entrée de votre étape de test. En règle générale, ce chemin est similaire à `app-release.apk`.

k. Dans TestType, entrez votre type de test, puis dans Test, entrez le chemin du fichier de définition du test. Ce chemin dépend de la racine de l'artefact d'entrée de votre test.

   Voici une liste de valeurs valides pour TestType:

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG
- APPIUM_NODE
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- FUZZ INTÉGRÉ
- INSTRUMENTATION
- XCTEST
- XCTEST_UI

> **Note**
>
> Les nœuds d'environnement personnalisés ne sont pas pris en charge.

l. Dans les autres champs, indiquez la configuration appropriée pour votre test et le type d'application.

m. (Facultatif) Dans Avancé, indiquez les informations de configuration pour le test.

n. Choisissez Enregistrer.

o. Dans l'étape que vous modifiez, choisissez Effectué. Dans le volet AWS CodePipeline , choisissez Enregistrer puis Enregistrer dans le message d'avertissement.

p. Pour soumettre vos modifications et lancer la génération d'un pipeline, choisissez Changement de version, puis Publication.

Tutoriel : Créez un pipeline qui teste votre application iOS avec AWS Device Farm

Vous pouvez l'utiliser AWS CodePipeline pour configurer facilement un flux d'intégration continue dans lequel votre application est testée chaque fois que le compartiment source change. Ce didacticiel vous montre comment créer et configurer un pipeline pour tester votre application iOS à partir d'un compartiment S3. Le pipeline détecte l'arrivée d'une modification enregistrée via Amazon CloudWatch Events, puis utilise Device Farm pour tester l'application créée.

Important

Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé CodePipeline pour les artefacts. (Ceci est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient à cette personne sûre et Comptes AWS qu'elle sera fiable.

Important

La plupart des actions que vous ajoutez à votre pipeline dans le cadre de cette procédure impliquent AWS des ressources que vous devez créer avant de créer le pipeline. AWS les ressources pour vos actions source doivent toujours être créées dans la même AWS région que celle où vous créez votre pipeline. Par exemple, si vous créez votre pipeline dans la région USA Est (Ohio), votre CodeCommit référentiel doit se trouver dans la région USA Est (Ohio).

Vous pouvez ajouter des actions interrégionales lorsque vous créez votre pipeline. AWS les ressources pour les actions interrégionales doivent se trouver dans la même AWS région que celle où vous prévoyez d'exécuter l'action. Pour de plus amples informations, veuillez consulter Ajouter une action interrégionale dans CodePipeline.

Vous pouvez tester cela en utilisant votre application iOS existante ou vous pouvez utiliser l'exemple d'application iOS.

Avant de commencer

1. Connectez-vous à la AWS Device Farm console et choisissez Créer un nouveau projet.
2. Choisissez votre projet. Dans le navigateur, copiez l'URL de votre nouveau projet. L'URL contient l'ID de projet.
3. Copiez et conservez cet ID de projet. Vous l'utiliserez lorsque vous créerez votre pipeline dans CodePipeline.

   Voici un exemple d'URL de projet. Pour extraire l'ID de projet, copiez la valeur figurant après `projects/`. Dans cet exemple, l'ID de projet est `eec4905f-98f8-40aa-9afc-4c1cfexample`.

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

Configurer CodePipeline pour utiliser vos tests Device Farm (exemple Amazon S3)

1. Créez ou utilisez un compartiment S3 avec la gestion des versions activée. Suivez les instructions fournies dans Étape 1 : créer un compartiment source S3 pour votre application pour créer un compartiment S3.
2. Dans la console Amazon S3 de votre compartiment, choisissez Upload et suivez les instructions pour télécharger votre fichier .zip.

   Votre exemple d'application doit être empaqueté dans un fichier .zip.
3. Pour créer votre pipeline et ajouter une étape source, procédez comme suit :

   a. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse https://console.aws.amazon.com/codepipeline/.

   b. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.

   c. Sur la page Étape 1 : Choisir une option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.

   d. Sur la page Étape 2 : Choisir les paramètres du pipeline, dans Nom du pipeline, entrez le nom de votre pipeline.

   e. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul type que vous pouvez choisir dans la console. Pour plus

d'informations, consultez la section Types de pipelines. Pour plus d'informations sur la tarification CodePipeline, consultez la section Tarification.

f. Dans Rôle de service, laissez Nouveau rôle de service sélectionné et laissez Nom du rôle inchangé. Vous pouvez également choisir d'utiliser un rôle de service si vous en avez déjà un.

> **Note**
>
> Si vous utilisez un rôle de CodePipeline service créé avant juillet 2018, vous devez ajouter des autorisations pour Device Farm. Pour ce faire, ouvrez la console IAM, recherchez le rôle, puis ajoutez les autorisations suivantes à la politique du rôle. Pour de plus amples informations, veuillez consulter Ajouter des autorisations au rôle CodePipeline de service.
>
> ```
> {
>      "Effect": "Allow",
>      "Action": [
>         "devicefarm:ListProjects",
>         "devicefarm:ListDevicePools",
>         "devicefarm:GetRun",
>         "devicefarm:GetUpload",
>         "devicefarm:CreateUpload",
>         "devicefarm:ScheduleRun"
>      ],
>      "Resource": "*"
> }
> ```

g. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.

h. Sur la page Étape 3 : Ajouter une étape source, dans Source provider, choisissez Amazon S3.

i. Dans l'emplacement Amazon S3, entrez le compartiment, par exemple `my-storage-bucket`, et la clé d'objet, par exemple `s3-ios-test-1.zip` pour votre fichier .zip.

j. Choisissez Suivant.

4. À l'étape 4 : Ajouter une phase de construction, créez une étape de construction fictive pour votre pipeline. Cela vous permet de créer le pipeline dans l'assistant. Une fois que vous avez créé votre pipeline à deux étapes à l'aide de l'assistant, vous n'avez plus besoin de cette

étape de génération d'espace réservé. Une fois le pipeline terminé, cette deuxième étape est supprimée et la nouvelle étape de test est ajoutée à l'étape 5.

a. Dans Fournisseur de génération, choisissez Ajouter Jenkins. Cette sélection de build est uniquement un espace réservé. Elle n'est pas utilisée.

b. Dans Nom du fournisseur, entrez un nom. Ce nom est un espace réservé. Elle n'est pas utilisée.

c. Dans URL du serveur, entrez du texte. Ce texte est un espace réservé. Elle n'est pas utilisée.

d. Dans Nom du projet, entrez un nom. Ce nom est un espace réservé. Elle n'est pas utilisée.

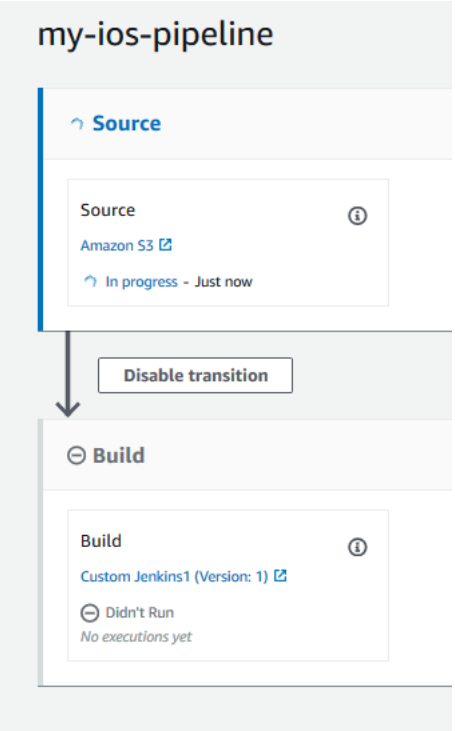
e. Choisissez Suivant.

f. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer l'étape de test, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer.

   Choisissez Suivant.

g. Sur la page Étape 6 : Ajouter une phase de déploiement, choisissez Ignorer l'étape de déploiement, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer.

h. À l'étape 7 : Révision, choisissez Créer un pipeline. Vous devez voir un diagramme montrant les étapes source et de génération.



5. Ajoutez une action de test Device Farm à votre pipeline comme suit :

   a. Dans le coin supérieur droit, choisissez Modifier.

   b. Choisissez Modifier l'étape. Sélectionnez Delete (Supprimer). Cela supprime l'étape d'espace réservé maintenant que vous n'en avez plus besoin pour la création du pipeline.

   c. Au bas du diagramme, choisissez + Ajouter une étape.

   d. Dans le nom de l'étape, saisissez un nom pour l'étape, par exemple Test, puis choisissez Ajouter une étape.

   e. Choisissez + Ajouter un groupe d'actions.

   f. Dans Nom de l'action, entrez un nom, tel que DeviceFarmTest.

   g. Dans Action provider, choisissez AWS Device Farm. Acceptez la région du pipeline comme Région par défaut.

   h. Dans Artefacts d'entrée, choisissez l'artefact d'entrée correspondant à l'artefact de sortie de l'étape qui précède l'étape de test, comme `SourceArtifact`.

      Dans la AWS CodePipeline console, vous pouvez trouver le nom de l'artefact de sortie pour chaque étape en survolant l'icône d'information dans le diagramme du pipeline. Si votre pipeline teste votre application directement depuis l'étape Source, choisissez SourceArtifact. Si le pipeline inclut une phase de construction, choisissez BuildArtifact.

   i. Dans ProjectId, choisissez l'ID de votre projet Device Farm. Suivez les étapes indiquées au début de ce didacticiel pour récupérer votre ID de projet.

   j. Dans DevicePoolArn, entrez l'ARN du pool de périphériques. Pour obtenir le pool de périphériques disponible ARNs pour le projet, y compris l'ARN des meilleurs appareils, utilisez la AWS CLI pour entrer la commande suivante :

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

   k. Dans AppType, entrez iOS.

      Voici une liste de valeurs valides pour AppType:

      - iOS
      - Android
      - Web

l. Dans App, saisissez le chemin d'accès au package de l'application compilée. Ce chemin d'accès se rapporte à la racine de l'artefact d'entrée de votre étape de test. En règle générale, ce chemin est similaire à `ios-test.ipa`.

m. Dans TestType, entrez votre type de test, puis dans Test, entrez le chemin du fichier de définition du test. Ce chemin dépend de la racine de l'artefact d'entrée de votre test.

   Si vous utilisez l'un des tests Device Farm intégrés, entrez le type de test configuré dans votre projet Device Farm, tel que BUILTIN_FUZZ. Dans FuzzEventCount, entrez une durée en millisecondes, par exemple 6 000. Dans FuzzEventThrottle, entrez une durée en millisecondes, par exemple 50.

   Si vous n'utilisez pas l'un des tests Device Farm intégrés, entrez votre type de test, puis dans Test, entrez le chemin du fichier de définition du test. Ce chemin dépend de la racine de l'artefact d'entrée de votre test.

   Voici une liste de valeurs valides pour TestType:

   - APPIUM_JAVA_JUNIT
   - APPIUM_JAVA_TESTNG
   - APPIUM_NODE
   - APPIUM_RUBY
   - APPIUM_PYTHON
   - APPIUM_WEB_JAVA_JUNIT
   - APPIUM_WEB_JAVA_TESTNG
   - APPIUM_WEB_NODE
   - APPIUM_WEB_RUBY
   - APPIUM_WEB_PYTHON
   - FUZZ INTÉGRÉ
   - INSTRUMENTATION
   - XCTEST
   - XCTEST_UI

> **Note**
>
> Les nœuds d'environnement personnalisés ne sont pas pris en charge.

n. Dans les autres champs, indiquez la configuration appropriée pour votre test et le type d'application.

o. (Facultatif) Dans Avancé, indiquez les informations de configuration pour le test.

p. Choisissez Enregistrer.

q. Dans l'étape que vous modifiez, choisissez Effectué. Dans le volet AWS CodePipeline , choisissez Enregistrer puis Enregistrer dans le message d'avertissement.

r. Pour soumettre vos modifications et lancer l'exécution d'un pipeline, choisissez Changement de version, puis Publication.

Tutoriel : Création d'un pipeline qui se déploie vers Service Catalog

Le catalogue de services vous permet de créer et de fournir des produits sur la base AWS CloudFormation de modèles.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Ce didacticiel explique comment créer et configurer un pipeline pour déployer votre modèle de produit dans Service Catalog et appliquer les modifications que vous avez apportées à votre référentiel source (déjà créé dans GitHub CodeCommit, ou Amazon S3).

> **Note**
>
> Lorsqu'Amazon S3 est le fournisseur source de votre pipeline, vous devez télécharger dans votre compartiment tous les fichiers sources regroupés dans un seul fichier .zip. Sinon, l'action source échoue.

Tout d'abord, vous créez un produit dans Service Catalog, puis vous créez un pipeline dans AWS CodePipeline. Ce didacticiel fournit deux options pour définir la configuration de déploiement :

- Créez un produit dans Service Catalog et chargez un fichier modèle dans votre référentiel source. Indiquez la version du produit et la configuration de déploiement dans la CodePipeline console (sans fichier de configuration distinct). Consultez Option 1 : déployer vers Service Catalog sans fichier de configuration.

  > **Note**
  >
  > Le fichier de modèle peut être créé au format YAML ou JSON.

- Créez un produit dans Service Catalog et chargez un fichier modèle dans votre référentiel source. Fournir la version du produit et la configuration de déploiement dans un fichier de configuration distinct. Consultez Option 2 : déployer vers Service Catalog à l'aide d'un fichier de configuration.

Option 1 : déployer vers Service Catalog sans fichier de configuration

Dans cet exemple, vous téléchargez l'exemple de fichier AWS CloudFormation modèle pour un compartiment S3, puis vous créez votre produit dans Service Catalog. Ensuite, vous créez votre pipeline et vous spécifiez la configuration de déploiement dans la CodePipeline console.

Étape 1 : Charger l'exemple de fichier de modèle dans le référentiel source

1. Ouvrez un éditeur de texte. Créez un exemple de modèle en collant les informations suivantes dans le fichier. Enregistrez le fichier sous le nom `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"
    }
  }
}
```

Ce modèle permet AWS CloudFormation de créer un compartiment S3 qui peut être utilisé par Service Catalog.

2. Chargez le fichier `S3_template.json` dans votre référentiel AWS CodeCommit .

Étape 2 : Création d'un produit dans le catalogue de services

1. En tant qu'administrateur informatique, connectez-vous à la console Service Catalog, accédez à la page Produits, puis choisissez Charger un nouveau produit.
2. Sur la page Charger un nouveau produit, procédez comme suit :
   a. Dans Nom du produit, entrez le nom que vous souhaitez utiliser pour votre nouveau produit.
   b. Dans Description, entrez la description du catalogue de produits. Cette description apparaît dans la liste des produits pour aider l'utilisateur à choisir le bon produit.
   c. Dans Fourni par, entrez le nom de votre service ou administrateur informatique.
   d. Choisissez Suivant.
3. (Facultatif) Dans Saisir les informations du Support, saisissez les coordonnées du support produit, puis choisissez Suivant.
4. Dans Détails de la version, procédez comme suit :

a. Choisissez Charger un fichier de modèle. Recherchez votre fichier `S3_template.json` et chargez-le.

b. Dans Nom de la version, entrez le nom de la version du produit (par exemple, **`devops S3 v2`**).

c. Dans Description, entrez les détails qui distinguent cette version des autres.

d. Choisissez Suivant.

5. Dans la page Vérification, vérifiez que les informations sont exactes, puis choisissez Créer.

6. Sur la page Produits, dans le navigateur, copiez l'URL de votre nouveau produit. Elle contient l'ID du produit. Copiez et conservez cet ID de produit. Vous l'utiliserez lorsque vous créerez votre pipeline dans CodePipeline.

   Voici l'URL d'un produit nommé `my-product`. Pour extraire l'ID de produit, copiez la valeur comprise entre le signe égal (=) et le caractère esperluette (&). Dans cet exemple, l'ID de produit est `prod-example123456`.

   ```
   https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
   productCreated=prod-example123456&createdProductTitle=my-product
   ```

   > **Note**
   >
   > Copiez l'URL de votre produit avant de quitter la page. Lorsque vous quittez cette page, vous devez utiliser l'interface de ligne de commande pour obtenir votre ID de produit.

   Après quelques secondes, votre produit s'affiche sur la page Produits. Vous devrez peut-être actualiser votre navigateur pour voir le produit dans la liste.

Étape 3 : Créer votre pipeline

1. Pour nommer votre pipeline et sélectionner ses paramètres, procédez comme suit :

   a. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.

   b. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.

c. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.

d. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez un nom pour votre pipeline.

e. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.

f. Dans Rôle de service, choisissez Nouveau rôle de service pour permettre CodePipeline de créer un rôle de service dans IAM.

g. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.

2. Pour ajouter une étape source sur la page Étape 3 : Ajouter une étape source, procédez comme suit :

   a. Dans Fournisseur de source, choisissez AWS CodeCommit.

   b. Dans Nom du référentiel et Nom de branche, entrez le référentiel et la branche que vous souhaitez utiliser pour votre action source.

   c. Choisissez Suivant.

3. À l'étape 4 : Ajouter une étape de construction, choisissez Ignorer l'étape de génération, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

4. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

   Choisissez Suivant.

5. À l'étape 6 : Ajouter une étape de déploiement, procédez comme suit :

   a. Dans Fournisseur de déploiement, choisissez AWS Service Catalog.

   b. Pour configuration du déploiement, choisissez Entrer la configuration du déploiement.

   c. Dans Product ID, collez l'ID produit que vous avez copié depuis la console Service Catalog.

   d. Dans Template file path (Chemin du fichier de modèle), entrez le chemin d'accès relatif de l'emplacement où le fichier de modèle est stocké.

   e. Dans Type de produit, choisissez CloudFormation un modèle.

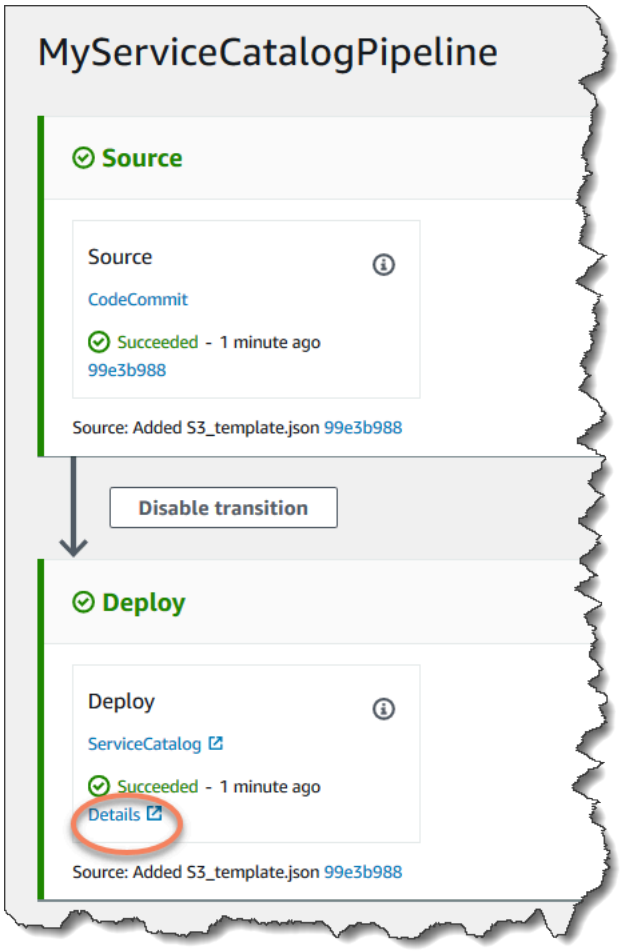
f. Dans Nom de la version du produit, entrez le nom de la version du produit que vous avez spécifiée dans le catalogue de services. Si vous souhaitez que la modification du modèle soit déployée dans une nouvelle version de produit, entrez un nom de version de produit qui n'a été utilisé pour aucune version précédente du même produit.

g. Pour Artefact d'entrée, choisissez l'artefact d'entrée source.

h. Choisissez Suivant.

6. À l'étape 7 : Vérifiez, passez en revue les paramètres de votre pipeline, puis choisissez Créer.

7. Une fois que votre pipeline s'exécute correctement, à l'étape de déploiement, choisissez Détails. Cela ouvre votre produit dans le catalogue de services.



8. Sous les informations sur le produit, choisissez le nom de votre version pour ouvrir le modèle de produit. Affichez le déploiement du modèle.

Étape 4 : Effectuez une modification et vérifiez votre produit dans le catalogue de services

1. Affichez votre pipeline dans la CodePipeline console et, sur votre scène source, choisissez Détails. Votre AWS CodeCommit référentiel source s'ouvre dans la console. Choisissez Modifier et apportez une modification au fichier (par exemple, à la description).

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

2. Validez et envoyez votre modification. Votre pipeline démarre une fois que vous avez envoyé la modification. Lorsque l'exécution du pipeline est terminée, lors de la phase de déploiement, choisissez Détails pour ouvrir votre produit dans Service Catalog.
3. Sous les informations sur le produit, choisissez le nouveau nom de version pour ouvrir le modèle de produit. Affichez la modification du modèle déployé.

Option 2 : déployer vers Service Catalog à l'aide d'un fichier de configuration

Dans cet exemple, vous téléchargez l'exemple de fichier AWS CloudFormation modèle pour un compartiment S3, puis vous créez votre produit dans Service Catalog. Vous pouvez également charger un fichier de configuration distinct qui spécifie votre configuration de déploiement. Ensuite, vous créez votre pipeline et spécifiez l'emplacement de votre fichier de configuration.

Étape 1 : Charger l'exemple de fichier de modèle dans le référentiel source

1. Ouvrez un éditeur de texte. Créez un exemple de modèle en collant les informations suivantes dans le fichier. Enregistrez le fichier sous le nom `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"
    }
  }
}
```

Ce modèle permet AWS CloudFormation de créer un compartiment S3 qui peut être utilisé par Service Catalog.

2. Chargez le fichier `S3_template.json` dans votre référentiel AWS CodeCommit .

Étape 2 : Créer le fichier de configuration du déploiement de votre produit

1. Ouvrez un éditeur de texte. Créez le fichier de configuration de votre produit. Le fichier de configuration est utilisé pour définir le déploiement de votre catalogue de services parameters/ preferences. Vous utilisez ce fichier lorsque vous créez votre pipeline.

   Cet exemple fournit le nom de version de produit (`ProductVersionName`) « devops S3 v2 » et la description de version de produit (`ProductVersionDescription`) `MyProductVersionDescription`. Si vous souhaitez que la modification du modèle soit déployée dans une nouvelle version de produit, entrez simplement un nom de version de produit qui n'a été utilisé pour aucune version précédente du même produit.

   Enregistrez le fichier sous le nom `sample_config.json`.

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

Ce fichier crée automatiquement les informations sur la version de produit à chaque fois que votre pipeline s'exécute.

2. Chargez le fichier `sample_config.json` dans votre référentiel AWS CodeCommit . Veillez à charger ce fichier dans votre référentiel source.

Étape 3 : Création d'un produit dans le catalogue de services

1. En tant qu'administrateur informatique, connectez-vous à la console Service Catalog, accédez à la page Produits, puis choisissez Charger un nouveau produit.
2. Sur la page Charger un nouveau produit, procédez comme suit :
   a. Dans Nom du produit, entrez le nom que vous souhaitez utiliser pour votre nouveau produit.
   b. Dans Description, entrez la description du catalogue de produits. Cette description apparaît dans la liste des produits pour aider l'utilisateur à choisir le bon produit.
   c. Dans Fourni par, entrez le nom de votre service ou administrateur informatique.
   d. Choisissez Suivant.
3. (Facultatif) Dans Saisir les informations du Support, saisissez les coordonnées du support produit, puis choisissez Suivant.
4. Dans Détails de la version, procédez comme suit :
   a. Choisissez Charger un fichier de modèle. Recherchez votre fichier `S3_template.json` et chargez-le.
   b. Dans Nom de la version, entrez le nom de la version du produit (par exemple, « devops S3 v2 »).
   c. Dans Description, entrez les détails qui distinguent cette version des autres.
   d. Choisissez Suivant.
5. Sur la page Vérification, vérifiez que les informations sont correctes, puis choisissez Confirmer et charger.
6. Sur la page Produits, dans le navigateur, copiez l'URL de votre nouveau produit. Elle contient l'ID du produit. Copiez et conservez cet ID de produit. Vous l'utiliserez lorsque vous créerez votre pipeline dans CodePipeline.

Voici l'URL d'un produit nommé `my-product`. Pour extraire l'ID de produit, copiez la valeur comprise entre le signe égal (=) et le caractère esperluette (&). Dans cet exemple, l'ID de produit est `prod-example123456`.

```
https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
productCreated=prod-example123456&createdProductTitle=my-product
```

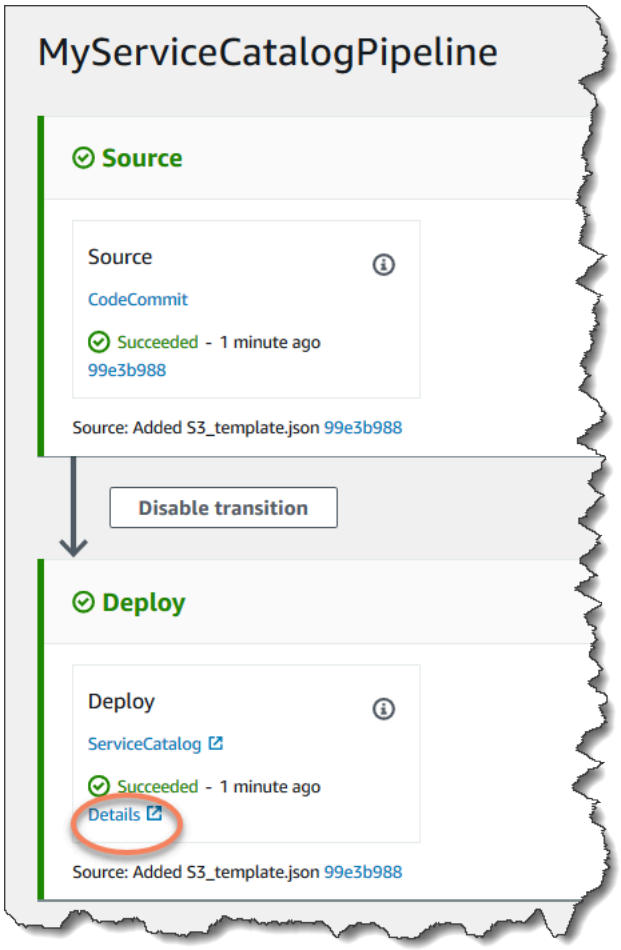
> **Note**
>
> Copiez l'URL de votre produit avant de quitter la page. Lorsque vous quittez cette page, vous devez utiliser l'interface de ligne de commande pour obtenir votre ID de produit.

Après quelques secondes, votre produit s'affiche sur la page Produits. Vous devrez peut-être actualiser votre navigateur pour voir le produit dans la liste.

Étape 4 : Créer votre pipeline

1. Pour nommer votre pipeline et sélectionner ses paramètres, procédez comme suit :
   a. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
   b. Choisissez Mise en route. Choisissez Créer un pipeline, puis entrez un nom pour votre pipeline.
   c. Dans Rôle de service, choisissez Nouveau rôle de service pour permettre CodePipeline de créer un rôle de service dans IAM.
   d. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
2. Pour ajouter une étape source, procédez comme suit :
   a. Dans Fournisseur de source, choisissez AWS CodeCommit.
   b. Dans Nom du référentiel et Nom de branche, entrez le référentiel et la branche que vous souhaitez utiliser pour votre action source.
   c. Choisissez Suivant.

3. Dans Add build stage (Ajouter une étape de génération), choisissez Skip build stage (Ignorer l'étape de génération) et acceptez le message d'avertissement en choisissant à nouveau Skip (Ignorer).
4. Dans Add deploy stage (Ajouter une étape de déploiement), procédez comme suit :
   a. Dans Fournisseur de déploiement, choisissez AWS Service Catalog.
   b. Choisissez Use configuration file (Utiliser un fichier de configuration).
   c. Dans Product ID, collez l'ID produit que vous avez copié depuis la console Service Catalog.
   d. Dans Configuration file path (Chemin du fichier de configuration), entrez le chemin d'accès du fichier de configuration dans votre référentiel.
   e. Choisissez Suivant.
5. Dans Vérification, vérifiez les paramètres de votre pipeline, puis choisissez Créer.
6. Une fois votre pipeline exécuté avec succès, lors de la phase de déploiement, choisissez Détails pour ouvrir votre produit dans Service Catalog.



7. Sous les informations sur le produit, choisissez le nom de votre version pour ouvrir le modèle de produit. Affichez le déploiement du modèle.

Étape 5 : Envoyer (push) une modification et vérifier votre produit dans Service Catalog

1. Affichez votre pipeline dans la CodePipeline console et, sur la scène source, choisissez Détails. Votre AWS CodeCommit référentiel source s'ouvre dans la console. Choisissez Modifier, puis apportez une modification au fichier (par exemple, à la description).

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

2. Validez et envoyez votre modification. Votre pipeline démarre une fois que vous avez envoyé la modification. Lorsque l'exécution du pipeline est terminée, lors de la phase de déploiement, choisissez Détails pour ouvrir votre produit dans Service Catalog.
3. Sous les informations sur le produit, choisissez le nouveau nom de version pour ouvrir le modèle de produit. Affichez la modification du modèle déployé.

Tutoriel : Création d'un pipeline avec AWS CloudFormation

Les exemples fournissent des exemples de modèles que vous pouvez utiliser AWS CloudFormation pour créer un pipeline qui déploie votre application sur vos instances chaque fois que le code source est modifié. L'exemple de modèle crée un pipeline que vous pouvez afficher dans AWS CodePipeline. Le pipeline détecte l'arrivée d'une modification enregistrée via Amazon CloudWatch Events.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Rubriques

- Exemple 1 : créer un AWS CodeCommit pipeline avec AWS CloudFormation
- Exemple 2 : créer un pipeline Amazon S3 avec AWS CloudFormation

Exemple 1 : créer un AWS CodeCommit pipeline avec AWS CloudFormation

Cette procédure pas à pas vous montre comment utiliser la AWS CloudFormation console pour créer une infrastructure qui inclut un pipeline connecté à un référentiel CodeCommit source. Dans ce didacticiel, vous allez utiliser l'exemple de fichier modèle fourni pour créer votre pile de ressources, qui inclut votre magasin d'artefacts, votre pipeline et vos ressources de détection des modifications, telles que votre règle Amazon CloudWatch Events. Après avoir créé votre pile de ressources dans AWS CloudFormation, vous pouvez afficher votre pipeline dans la AWS CodePipeline console.
Le pipeline est un gazoduc en deux étapes avec un étage CodeCommit source et un étage de CodeDeploy déploiement.

Prérequis :

Vous devez avoir créé les ressources suivantes pour pouvoir les utiliser avec l' AWS CloudFormation exemple de modèle :

- Vous devez avoir créé un référentiel source. Vous pouvez utiliser le AWS CodeCommit référentiel que vous avez créé dansTutoriel : Création d'un pipeline simple (CodeCommit référentiel).
- Vous devez avoir créé une CodeDeploy application et un groupe de déploiement. Vous pouvez utiliser les ressources CodeDeploy que vous avez créées dans Tutoriel : Création d'un pipeline simple (CodeCommit référentiel).
- Cliquez sur l'un de ces liens pour télécharger l'exemple de fichier AWS CloudFormation modèle permettant de créer un pipeline : YAML | JSON samples/codepipeline-codecommit-events-json.zip

  Décompressez le fichier et placez-le sur votre ordinateur local.
- Téléchargez le fichier Linux.zip d'application SampleApp _ exemple.

Créez votre pipeline dans AWS CloudFormation

1. Décompressez les fichiers depuis SampleApp _ Linux.zip et téléchargez-les dans votre AWS CodeCommit référentiel. Vous devez charger les fichiers décompressés dans le répertoire

racine de votre référentiel. Vous pouvez suivre les instructions figurant dans Étape 2 : ajouter un exemple de code à votre CodeCommit référentiel pour transmettre les fichiers à votre référentiel.

2. Ouvrez la AWS CloudFormation console et choisissez Create Stack. Choisissez Avec de nouvelles ressources (standard).
3. Sous Spécifier le modèle, choisissez Charger un modèle. Sélectionnez Choisir un fichier, puis choisissez le fichier modèle sur votre ordinateur local. Choisissez Suivant.
4. Dans Nom de la pile, entrez un nom pour votre pipeline. Les paramètres spécifiés par l'exemple de modèle s'affichent. Entrez les paramètres suivants :
   a. Dans ApplicationName, saisissez le nom de votre CodeDeploy application.
   b. Dans BetaFleet, entrez le nom de votre groupe de CodeDeploy déploiement.
   c. Dans BranchName, entrez la branche du référentiel que vous souhaitez utiliser.
   d. Dans RepositoryName, entrez le nom de votre référentiel CodeCommit source.
5. Choisissez Suivant. Acceptez les valeurs par défaut de la page suivante, puis choisissez Suivant.
6. Dans Fonctionnalités, sélectionnez Je reconnais que cela AWS CloudFormation peut créer des ressources IAM, puis choisissez Créer une pile.
7. Une fois la création de la pile terminée, affichez la liste des événements pour rechercher les erreurs éventuelles.

Résolution des problèmes

L'utilisateur IAM qui crée le pipeline dans AWS CloudFormation peut avoir besoin d'autorisations supplémentaires pour créer des ressources pour le pipeline. Les autorisations suivantes sont requises dans la politique pour permettre AWS CloudFormation de créer les ressources Amazon CloudWatch Events requises pour le CodeCommit pipeline :

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

```
    "Resource": "resource_ARN"
}
```

8. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.

   Sous Pipelines, choisissez votre pipeline et choisissez Afficher. Le schéma illustre les étapes source et de déploiement de votre pipeline.

   > **Note**
   >
   > Pour afficher le pipeline qui a été créé, recherchez la colonne ID logique sous l'onglet Ressources correspondant à votre pile CloudFormation. Notez le nom du pipeline dans la colonne Physical ID. Dans CodePipeline, vous pouvez afficher le pipeline avec le même identifiant physique (nom du pipeline) dans la région où vous avez créé votre pile.

9. Dans votre référentiel source, validez et envoyez une modification. Vos ressources de détection de modification récupèrent la modification, et votre pipeline démarre.

Exemple 2 : créer un pipeline Amazon S3 avec AWS CloudFormation

Cette procédure pas à pas vous montre comment utiliser la AWS CloudFormation console pour créer une infrastructure qui inclut un pipeline connecté à un compartiment source Amazon S3. Dans ce didacticiel, vous utilisez l'exemple de fichier modèle fourni pour créer votre pile de ressources, qui comprend votre compartiment source, votre magasin d'artefacts, votre pipeline et des ressources de détection des modifications, telles que votre règle et votre historique Amazon CloudWatch Events. CloudTrail Après avoir créé votre pile de ressources dans AWS CloudFormation, vous pouvez afficher votre pipeline dans la AWS CodePipeline console. Le pipeline est un pipeline en deux étapes avec une étape source Amazon S3 et une étape de CodeDeploy déploiement.

Prérequis :

Vous devez disposer des ressources suivantes pour utiliser l' AWS CloudFormation exemple de modèle :

- Vous devez avoir créé les instances Amazon EC2, sur lesquelles vous avez installé l' CodeDeploy agent. Vous devez avoir créé une CodeDeploy application et un groupe de déploiement. Utilisez Amazon EC2 et les CodeDeploy ressources que vous avez créées dansTutoriel : Création d'un pipeline simple (CodeCommit référentiel).

- Cliquez sur les liens suivants pour télécharger les exemples de fichiers AWS CloudFormation modèles permettant de créer un pipeline avec une source Amazon S3 :
  - Téléchargez l'exemple de modèle pour votre pipeline : YAML | JSON
  - Téléchargez l'exemple de modèle pour votre CloudTrail bucket et votre trail : YAML | JSON samples/codepipeline-s3-cloudtrail-json.zip
  - Décompressez les fichiers et placez-les sur votre ordinateur local.
- Téléchargez l'exemple d'application sur le SampleApp site _ Linux.zip.

  Enregistrez le fichier .zip sur votre ordinateur local. Vous chargez le fichier .zip après la création de la pile.

Créez votre pipeline dans AWS CloudFormation

1. Ouvrez la AWS CloudFormation console et choisissez Create Stack. Choisissez Avec de nouvelles ressources (standard).
2. Dans Choisir un modèle, choisissez Charger un modèle. Sélectionnez Choisir un fichier, puis choisissez le fichier modèle sur votre ordinateur local. Choisissez Suivant.
3. Dans Nom de la pile, entrez un nom pour votre pipeline. Les paramètres spécifiés par l'exemple de modèle s'affichent. Entrez les paramètres suivants :
   a. Dans ApplicationName, saisissez le nom de votre CodeDeploy application. Vous pouvez remplacer le nom par défaut `DemoApplication`.
   b. Dans BetaFleet, entrez le nom de votre groupe de CodeDeploy déploiement. Vous pouvez remplacer le nom par défaut `DemoFleet`.
   c. Dans le champ SourceObjectKey, saisissez `SampleApp_Linux.zip`. Vous chargez ce fichier dans votre compartiment une fois que le modèle a créé le compartiment et le pipeline.
4. Choisissez Suivant. Acceptez les valeurs par défaut de la page suivante, puis choisissez Suivant.
5. Dans Fonctionnalités, sélectionnez Je reconnais que cela AWS CloudFormation peut créer des ressources IAM, puis choisissez Créer une pile.
6. Une fois la création de la pile terminée, affichez la liste des événements pour rechercher les erreurs éventuelles.

   Résolution des problèmes

L'utilisateur IAM qui crée le pipeline dans AWS CloudFormation peut avoir besoin d'autorisations supplémentaires pour créer des ressources pour le pipeline. Les autorisations suivantes sont requises dans la politique pour permettre AWS CloudFormation de créer les ressources Amazon CloudWatch Events requises pour le pipeline Amazon S3 :

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

7. Dans CloudFormation l'onglet Ressources de votre pile, visualisez les ressources qui ont été créées pour votre pile.

> **Note**
>
> Pour afficher le pipeline qui a été créé, recherchez la colonne ID logique sous l'onglet Ressources correspondant à votre pile CloudFormation. Notez le nom du pipeline dans la colonne Physical ID. Dans CodePipeline, vous pouvez afficher le pipeline avec le même identifiant physique (nom du pipeline) dans la région où vous avez créé votre pile.

Choisissez le compartiment S3 dont le nom comporte une étiquette `sourcebucket`, par exemple `s3-cfn-codepipeline-sourcebucket-y04EXAMPLE.` Ne choisissez pas le compartiment d'artefacts de pipeline.

Le compartiment source est vide, car la ressource est nouvellement créée par CloudFormation. Ouvrez la console Amazon S3 et localisez votre `sourcebucket` compartiment. Choisissez Charger, puis suivez les instructions pour charger votre fichier .zip `SampleApp_Linux.zip`.

> **Note**
>
> Lorsqu'Amazon S3 est le fournisseur source de votre pipeline, vous devez télécharger dans votre compartiment tous les fichiers sources regroupés dans un seul fichier .zip. Sinon, l'action source échoue.

8. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.

   Sous Pipelines, choisissez votre pipeline, puis choisissez Afficher. Le schéma illustre les étapes source et de déploiement de votre pipeline.

9. Effectuez les étapes de la procédure suivante pour créer vos ressources AWS CloudTrail .

Créez votre AWS CloudTrail ressources en AWS CloudFormation

1. Ouvrez la AWS CloudFormation console et choisissez Create Stack.
2. Dans Choisir un modèle, choisissez Télécharger un modèle sur Amazon S3. Choisissez Parcourir, puis sélectionnez le fichier modèle pour les AWS CloudTrail ressources sur votre ordinateur local. Choisissez Suivant.
3. Dans Nom de la pile, entrez le nom de votre pile de ressources. Les paramètres spécifiés par l'exemple de modèle s'affichent. Entrez les paramètres suivants :

   - Dans SourceObjectKey, acceptez la valeur par défaut pour le fichier zip de l'exemple d'application.

4. Choisissez Suivant. Acceptez les valeurs par défaut de la page suivante, puis choisissez Suivant.
5. Dans Fonctionnalités, sélectionnez Je reconnais que cela AWS CloudFormation peut créer des ressources IAM, puis choisissez Créer.
6. Une fois la création de la pile terminée, affichez la liste des événements pour rechercher les erreurs éventuelles.

   Les autorisations suivantes sont requises dans la politique pour AWS CloudFormation permettre de créer les CloudTrail ressources requises pour le pipeline Amazon S3 :

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

```
      "Action": [
         "cloudtrail:CreateTrail",
         "cloudtrail:DeleteTrail",
         "cloudtrail:StartLogging",
         "cloudtrail:StopLogging",
         "cloudtrail:PutEventSelectors"
      ],
      "Resource": "resource_ARN"
}
```

7. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.

   Sous Pipelines, choisissez votre pipeline, puis choisissez Afficher. Le schéma illustre les étapes source et de déploiement de votre pipeline.

8. Dans votre compartiment source, validez et envoyez une modification. Vos ressources de détection de modification récupèrent la modification, et votre pipeline démarre.

Tutoriel : Création d'un pipeline qui utilise des variables provenant de AWS CloudFormation actions de déploiement

Dans ce didacticiel, vous allez utiliser la AWS CodePipeline console pour créer un pipeline avec une action de déploiement. Lorsque le pipeline s'exécute, le modèle crée une pile, ainsi qu'un fichier `outputs`. Les sorties générées par le modèle de pile sont les variables générées par l' AWS CloudFormation action dans CodePipeline.

Dans l'action où vous créez la pile à partir du modèle, vous définissez un espace de noms de variables. Les variables produites par le fichier `outputs` peuvent ensuite être consommées par les actions suivantes. Dans cet exemple, vous créez un ensemble de modifications basé sur la `StackName` variable produite par l' AWS CloudFormation action. Après une approbation manuelle, vous exécutez le jeu de modifications, puis créez une action de suppression de pile qui supprime la pile en fonction de la variable `StackName`.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un

compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Rubriques

- Prérequis : créer un AWS CloudFormation rôle de service et CodeCommit référentiel
- Étape 1 : Téléchargez, modifiez et chargez l'échantillon AWS CloudFormation modèle
- Étape 2 : Créer votre pipeline
- Étape 3 : Ajoutez un CloudFormation action de déploiement pour créer l'ensemble de modifications
- Étape 4 : Ajouter une action d'approbation manuelle
- Étape 5 : Ajouter une action CloudFormation de déploiement pour exécuter l'ensemble de modifications
- Étape 6 : Ajouter une action CloudFormation de déploiement pour supprimer la pile

Prérequis : créer un AWS CloudFormation rôle de service et CodeCommit référentiel

Vous devez déjà disposer des éléments suivants :

- Un CodeCommit référentiel. Vous pouvez utiliser le AWS CodeCommit référentiel que vous avez créé dansTutoriel : Création d'un pipeline simple (CodeCommit référentiel).
- Cet exemple crée une pile Amazon DocumentDB à partir d'un modèle. Vous devez utiliser Gestion des identités et des accès AWS (IAM) pour créer un rôle de AWS CloudFormation service avec les autorisations suivantes pour Amazon DocumentDB.

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

Étape 1 : Téléchargez, modifiez et chargez l'échantillon AWS CloudFormation modèle

Téléchargez l'exemple AWS CloudFormation de fichier modèle et chargez-le dans votre CodeCommit référentiel.

1. Accédez à l'exemple de modèle pour votre région. Par exemple, utilisez le tableau https://docs.aws.amazon.com/documentdb/latest/developerguide/quick_start_cfn.html#quick_start_cfn-launch_stack pour choisir la région et télécharger le modèle. Téléchargez le modèle pour un cluster Amazon DocumentDB. Le nom du fichier est `documentdb_full_stack.yaml`.
2. Décompressez le fichier `documentdb_full_stack.yaml` et ouvrez-le dans un éditeur de texte. Effectuez les modifications suivantes.

   a. Pour cet exemple, ajoutez le paramètre `Purpose:` suivant à votre section `Parameters` du modèle.

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

   b. Pour cet exemple, ajoutez la sortie `StackName` suivante à votre section `Outputs:` du modèle.

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

3. Téléchargez le fichier modèle dans votre AWS CodeCommit référentiel. Vous devez charger le fichier de modèle décompressé et modifié dans le répertoire racine de votre référentiel.

   Pour télécharger vos fichiers à l' CodeCommit aide de la console, procédez comme suit :

   a. Ouvrez la CodeCommit console et choisissez votre référentiel dans la liste des référentiels.

   b. Choisissez Ajouter un fichier, puis choisissez Charger le fichier.

c. Sélectionnez Choose file (Choisir un fichier), puis naviguez vers votre fichier. Validez la modification en entrant votre nom d'utilisateur et votre adresse e-mail. Choisissez Valider les modifications.

Votre fichier doit se présenter comme suit au niveau racine de votre référentiel :

```
documentdb_full_stack.yaml
```

Étape 2 : Créer votre pipeline

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Stage source comportant une CodeCommit action dans laquelle l'artefact source est votre fichier modèle.
- Une étape de déploiement avec une action CloudFormation de déploiement.

Chaque action des étapes source et de déploiement créées par l'assistant se voit attribuer un espace de noms variable, `SourceVariables` et `DeployVariables`, respectivement. Comme un espace de noms est attribué aux actions, les variables configurées dans cet exemple sont disponibles pour les actions en aval. Pour de plus amples informations, veuillez consulter Référence des variables.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyCFNDeployPipeline`**.
5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.
6. Dans Rôle du service, sélectionnez l'une des options suivantes :

- Choisissez Nouveau rôle de service pour permettre CodePipeline de créer un rôle de service dans IAM.
- Choisissez Existing service role (Rôle de service existant). Dans Role name (Nom du rôle), choisissez votre rôle de service à partir de la liste.

7. Dans Magasin d'artefacts :

   a. Choisissez Emplacement par défaut pour utiliser le magasin d'artefacts par défaut, tel que le compartiment d'artefacts Amazon S3 désigné par défaut, pour votre pipeline dans la région que vous avez sélectionnée pour votre pipeline.

   b. Choisissez Emplacement personnalisé si vous possédez déjà un magasin d'artefacts, tel qu'un compartiment d'artefacts Amazon S3, dans la même région que votre pipeline.

> **Note**
>
> Il ne s'agit pas du compartiment source de votre code source. Il s'agit du magasin d'artefacts pour votre pipeline. Un magasin d'artefacts distinct, tel qu'un compartiment S3, est nécessaire pour chaque pipeline. Lorsque vous créez ou modifiez un pipeline, vous devez disposer d'un compartiment d'artefacts dans la région du pipeline et d'un compartiment d'artefacts par AWS région dans laquelle vous exécutez une action. Pour plus d'informations, consultez Artefacts d'entrée et de sortie et CodePipeline référence de la structure du pipeline.

   Choisissez Suivant.

8. À l'étape 3 : ajouter une étape source :

   a. Dans Fournisseur de source, choisissez AWS CodeCommit.

   b. Dans Nom du référentiel, choisissez le nom du CodeCommit référentiel dans lequel vous avez créé Étape 1 : Création d'un CodeCommit référentiel.

   c. Dans Nom de branche, choisissez le nom de la branche qui contient votre dernière mise à jour du code.

   Une fois que vous avez sélectionné le nom du référentiel et la branche, la règle Amazon CloudWatch Events à créer pour ce pipeline s'affiche.

Choisissez Suivant.

9. À l'étape 4 : Ajouter une étape de construction, choisissez Ignorer l'étape de génération, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

   Choisissez Suivant.

10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.

11. À l'étape 6 : ajouter une étape de déploiement :

    a. Dans Nom de l'action, choisissez Déployer. Dans Fournisseur de déploiement, choisissez CloudFormation.

    b. Dans Mode d'action, choisissez Créer ou mettre à jour une pile.

    c. Dans Nom de la pile, attribuez un nom à la pile. Il s'agit du nom de la pile que le modèle va créer.

    d. Dans Nom du fichier de sortie, entrez un nom pour le fichier de sorties, par exemple **`outputs`**. Il s'agit du nom du fichier qui sera créé par l'action après la création de la pile.

    e. Développez Avancé. Sous Remplacements de paramètres, entrez les remplacements de modèles en tant que paires clé-valeur. Par exemple, ce modèle nécessite les remplacements suivants.

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

    Si vous ne saisissez pas de valeurs de remplacement, le modèle crée une pile avec les valeurs par défaut.

    f. Choisissez Suivant.

    g. À l'étape 7 : Révision, choisissez Créer un pipeline. Vous devriez voir un schéma qui montre les étapes du pipeline. Autorisez votre pipeline s'exécuter. Votre pipeline en deux étapes est terminé et prêt pour les étapes supplémentaires à ajouter.

Étape 3 : Ajoutez un CloudFormation action de déploiement pour créer l'ensemble de modifications

Créez une action suivante dans votre pipeline qui permettra de CloudFormation créer l'ensemble de modifications avant l'action d'approbation manuelle.

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.

   Sous Pipelines, choisissez votre pipeline et choisissez Afficher. Le schéma illustre les étapes source et de déploiement de votre pipeline.
2. Choisissez de modifier le pipeline ou continuez à l'afficher en mode Edition.
3. Choisissez de modifier l'étape de déploiement.
4. Ajoutez une action de déploiement qui créera un ensemble de modifications pour la pile créée lors de l'action précédente. Vous ajoutez cette action après l'action existante dans l'étape.

   a. Dans Nom de l'action, entrez Change_Set. Dans Action provider, sélectionnez AWS CloudFormation .

   b. Dans Input artefact, choisissez SourceArtifact.

   c. Dans Mode d'action, choisissez Créer ou remplacer un jeu de modifications.

   d. Dans Nom de la pile, entrez la syntaxe de la variable comme indiqué. Il s'agit du nom de la pile pour laquelle le jeu de modifications est créé, où l'espace de noms par défaut `DeployVariables` est affecté à l'action.

   ```
   #{DeployVariables.StackName}
   ```

   e. Dans Nom du jeu de modifications, entrez le nom du jeu de modifications.

   ```
   my-changeset
   ```

   f. Dans Remplacements de paramètres, modifiez le paramètre `Purpose` de `testing` en `production`.

   ```
   {
   "DBClusterName": "MyDBCluster",
   "DBInstanceName": "MyDBInstance",
   "MasterUser": "UserName",
   "MasterPassword": "Password",
   ```

```
"DBInstanceClass": "db.r4.large",
"Purpose": "production"}
```

g. Choisissez Terminé pour enregistrer l'action.

Étape 4 : Ajouter une action d'approbation manuelle

Créez une action d'approbation manuelle dans votre pipeline.

1. Choisissez de modifier le pipeline ou continuez à l'afficher en mode Edition.
2. Choisissez de modifier l'étape de déploiement.
3. Ajoutez une action d'approbation manuelle après l'action de déploiement qui crée le jeu de modifications. Cette action vous permet de vérifier l'ensemble de modifications de ressources créé CloudFormation avant que le pipeline n'exécute l'ensemble de modifications.

Étape 5 : Ajouter une action CloudFormation de déploiement pour exécuter l'ensemble de modifications

Créez une action suivante dans votre pipeline qui permet d' CloudFormation exécuter l'ensemble de modifications après l'action d'approbation manuelle.

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.

   Sous Pipelines, choisissez votre pipeline et choisissez Afficher. Le schéma illustre les étapes source et de déploiement de votre pipeline.
2. Choisissez de modifier le pipeline ou continuez à l'afficher en mode Edition.
3. Choisissez de modifier l'étape de déploiement.
4. Ajoutez une action de déploiement qui exécutera l'ensemble de modifications approuvé lors de l'action manuelle précédente :

   a. Dans Nom de l'action, entrez Execute_Change_Set. Dans Action provider, sélectionnez AWS CloudFormation.

   b. Dans Input artefact, choisissez SourceArtifact.

   c. Dans Action mode (Mode d'action), choisissez Execute a change set (Exécuter un jeu de modifications).

d. Dans Nom de la pile, entrez la syntaxe de la variable comme indiqué. Il s'agit du nom de la pile pour laquelle le jeu de modifications est créé.

```
#{DeployVariables.StackName}
```

e. Dans Nom du jeu de modifications, entrez le nom du jeu de modifications que vous avez créé lors de l'action précédente.

```
my-changeset
```

f. Choisissez Terminé pour enregistrer l'action.

g. Continuez l'exécution du pipeline.

Étape 6 : Ajouter une action CloudFormation de déploiement pour supprimer la pile

Créez une action finale dans votre pipeline qui permet d' CloudFormation obtenir le nom de la pile à partir de la variable dans le fichier de sorties et de supprimer la pile.

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.

   Sous Pipelines, choisissez votre pipeline et choisissez Afficher. Le schéma illustre les étapes source et de déploiement de votre pipeline.
2. Choisissez de modifier le pipeline.
3. Choisissez de modifier l'étape de déploiement.
4. Ajoutez une action de déploiement qui supprimera la pile :

   a. Dans Nom de l'action, choisissez DeleteStack. Dans Fournisseur de déploiement, choisissez CloudFormation.

   b. Dans Mode d'action, choisissez Supprimer une pile.

   c. Dans Nom de la pile, entrez la syntaxe de la variable comme indiqué. Il s'agit du nom de la pile que l'action va supprimer.

   d. Choisissez Terminé pour enregistrer l'action.

   e. Choisissez Save (Enregistrer) pour enregistrer la stratégie.

Le pipeline s'exécute lorsqu'il est enregistré.

Tutoriel : Déploiement standard d'Amazon ECS avec CodePipeline

Ce didacticiel vous aide à créer un pipeline de déploiement continu (CD) complet de bout en bout avec Amazon ECS avec CodePipeline.

Important

Dans le cadre de la création d'un pipeline dans la console, un compartiment d'artefacts S3 sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Note

Ce didacticiel concerne l'action de déploiement standard d'Amazon ECS pour CodePipeline. Pour un didacticiel qui utilise Amazon ECS pour CodeDeploy blue/green déployer des actions CodePipeline, consultezTutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR.

Note

Ce didacticiel concerne l'action de déploiement standard d'Amazon ECS pour CodePipeline une action source. Pour un didacticiel qui utilise l'action de déploiement standard d'Amazon ECS ainsi ECRBuildAndPublish que l'action d' CodePipeline intégration pour diffuser votre image, consultezTutoriel : Créez et transférez une image Docker vers Amazon ECR avec CodePipeline (type V2).

Conditions préalables

Avant de pouvoir créer votre pipeline de déploiement continu à l'aide de ce didacticiel, vous devez avoir mis en place certaines ressources. Voici ce dont vous avez besoin pour commencer :

> **Note**
>
> Toutes ces ressources devraient être créées au sein de la même AWS région.

- Un référentiel de contrôle de source (utilisé par ce didacticiel CodeCommit) avec votre Dockerfile et la source de votre application. Pour plus d'informations, voir Créer un CodeCommit référentiel dans le Guide de AWS CodeCommit l'utilisateur.
- Un référentiel d'images Docker (ce didacticiel utilise Amazon ECR) qui contient une image que vous avez créée à partir de votre Dockerfile et de la source de votre application. Pour plus d'informations, consultez la section Création d'un référentiel et diffusion d'une image dans le guide de l'utilisateur d'Amazon Elastic Container Registry.
- Définition de tâche Amazon ECS qui fait référence à l'image Docker hébergée dans votre référentiel d'images. Pour plus d'informations, consultez la section Création d'une définition de tâche dans le guide du développeur Amazon Elastic Container Service.

> **Important**
>
> L'action de déploiement standard Amazon ECS pour CodePipeline crée sa propre révision de la définition de tâche en fonction de la révision utilisée par le service Amazon ECS. Si vous créez de nouvelles révisions pour la définition de tâche sans mettre à jour le service Amazon ECS, l'action de déploiement ignorera ces révisions.

Vous trouverez ci-dessous un exemple de définition de tâche utilisée pour ce didacticiel. La valeur que vous utilisez pour `name` et `family` qui sera utilisée à l'étape suivante pour votre fichier de spécifications de construction.

```
{
  "ipcMode": null,
  "executionRoleArn": "role_ARN",
  "containerDefinitions": [
    {
```

```
      "dnsSearchDomains": null,
      "environmentFiles": null,
      "logConfiguration": {
        "logDriver": "awslogs",
        "secretOptions": null,
        "options": {
          "awslogs-group": "/ecs/hello-world",
          "awslogs-region": "us-west-2",
          "awslogs-stream-prefix": "ecs"
        }
      },
      "entryPoint": null,
      "portMappings": [
        {
          "hostPort": 80,
          "protocol": "tcp",
          "containerPort": 80
        }
      ],
      "command": null,
      "linuxParameters": null,
      "cpu": 0,
      "environment": [],
      "resourceRequirements": null,
      "ulimits": null,
      "dnsServers": null,
      "mountPoints": [],
      "workingDirectory": null,
      "secrets": null,
      "dockerSecurityOptions": null,
      "memory": null,
      "memoryReservation": 128,
      "volumesFrom": [],
      "stopTimeout": null,
      "image": "image_name",
      "startTimeout": null,
      "firelensConfiguration": null,
      "dependsOn": null,
      "disableNetworking": null,
      "interactive": null,
      "healthCheck": null,
      "essential": true,
      "links": null,
      "hostname": null,
```

```
      "extraHosts": null,
      "pseudoTerminal": null,
      "user": null,
      "readonlyRootFilesystem": null,
      "dockerLabels": null,
      "systemControls": null,
      "privileged": null,
      "name": "hello-world"
    }
  ],
  "placementConstraints": [],
  "memory": "2048",
  "taskRoleArn": null,
  "compatibilities": [
    "EC2",
    "FARGATE"
  ],
  "taskDefinitionArn": "ARN",
  "family": "hello-world",
  "requiresAttributes": [],
  "pidMode": null,
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "networkMode": "awsvpc",
  "cpu": "1024",
  "revision": 1,
  "status": "ACTIVE",
  "inferenceAccelerators": null,
  "proxyConfiguration": null,
  "volumes": []
}
```

- Un cluster Amazon ECS qui exécute un service qui utilise la définition de tâche que vous avez mentionnée précédemment. Pour plus d'informations, consultez Création d'un cluster et création d'un service dans le guide du développeur Amazon Elastic Container Service.

Dès lors que ces prérequis sont respectés, vous pouvez commencer à créer votre pipeline de déploiement continu à l'aide du didacticiel.

Étape 1 : Ajout d'un fichier de spécification de génération dans votre référentiel source

Ce didacticiel vous CodeBuild permet de créer votre image Docker et de la transmettre à Amazon ECR. Ajoutez un fichier `buildspec.yml` à votre référentiel de code source pour indiquer à CodeBuild comment effectuer cette opération. L'exemple de spécification de génération ci-dessous prévoit les opérations suivantes :

- Pre-build étape :
  - Connectez-vous à Amazon ECR.
  - Définition de l'URI du référentiel en fonction de votre image ECR et ajout d'une balise d'image avec les sept premiers caractères de l'ID de validation Git de la source.
- Phase de génération :
  - Création de l'image Docker et balisage de l'image avec `latest` et l'ID de validation Git.
- Post-build étape :
  - Transmission de l'image à votre référentiel ECR avec les deux balises.
  - Écrivez un fichier appelé `imagedefinitions.json` à la racine de compilation contenant le nom du conteneur de votre service Amazon ECS, ainsi que l'image et la balise. La phase de déploiement de votre pipeline de déploiement continu s'appuie sur ces informations pour créer une révision de la définition de tâche de votre service avant de mettre à jour ce dernier afin qu'il utilise la nouvelle définition de tâche. Le fichier `imagedefinitions.json` est obligatoire pour l'exécutant de tâches ECS.

Collez cet exemple de texte pour créer votre `buildspec.yml` fichier et remplacez les valeurs de votre image et de votre définition de tâche. Ce texte utilise l'exemple d'ID de compte 111122223333.

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --
username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
      - REPOSITORY_URI=012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
```

```
      - IMAGE_TAG=${COMMIT_HASH:=latest}
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:latest
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - printf '[{"name":"hello-world","imageUri":"%s"}]' $REPOSITORY_URI:$IMAGE_TAG >
 imagedefinitions.json
artifacts:
    files: imagedefinitions.json
```

La spécification de génération a été écrite pour l'exemple de définition de tâche fourni dansConditions préalables, utilisé par le service Amazon ECS pour ce didacticiel. La valeur `REPOSITORY_URI` correspond au référentiel `image` (sans aucune balise d'image), tandis que la valeur *`hello-world`* située vers la fin du fichier correspond au nom de conteneur dans la définition de tâche du service.

Pour ajouter un **`buildspec.yml`** fichier dans votre référentiel source

1. Ouvrez un éditeur de texte, puis copiez et collez la spécification de génération précédente dans un nouveau fichier.
2. Remplacez la `REPOSITORY_URI` valeur (*`012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world`*) par l'URI de votre référentiel Amazon ECR (sans aucune balise d'image) pour votre image Docker. Remplacez *`hello-world`* par le nom de conteneur figurant dans la définition de tâche de votre service qui fait référence à votre image Docker.
3. Validez votre fichier `buildspec.yml` et transmettez-le à votre référentiel source.

   a. Ajoutez le fichier.

   ```
   git add .
   ```

   b. Validez la modification.

```
git commit -m "Adding build specification."
```

c. Transmettez la validation.

```
git push
```

Étape 2 : Création de votre pipeline de déploiement continu

Utilisez l' CodePipeline assistant pour créer les étapes de votre pipeline et connecter votre référentiel source à votre service ECS.

Pour créer le pipeline

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Welcome (Bienvenue), choisissez Créer un pipeline.

   S'il s'agit de votre première utilisation CodePipeline, une page d'introduction s'affiche à la place de Bienvenue. Choisir Get Started Now (Démarrer maintenant).
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, tapez le nom de votre pipeline. Pour ce didacticiel, le nom du pipeline est hello-world.
5. Dans Type de pipeline, conservez la sélection par défaut sur V2. Les types de pipelines diffèrent en termes de caractéristiques et de prix. Pour de plus amples informations, veuillez consulter Types de canalisations. Choisissez Suivant.
6. Sur la page Étape 3 : Ajouter une étape source, pour Fournisseur source, choisissez AWS CodeCommit.

   a. Pour Nom du référentiel, choisissez le nom du référentiel CodeCommit à utiliser comme emplacement source pour votre pipeline.

   b. Pour Nom de branche, choisissez la branche à utiliser, puis Étape suivante.
7. Sur la page Étape 4 : Ajouter une étape de construction, pour Build provider AWS CodeBuild, choisissez, puis choisissez Create project.

   a. Dans Nom du projet, attribuez un nom unique à votre projet de génération. Pour ce didacticiel, le nom de projet est hello-world.

b. Pour Image d'environnement, choisissez Managed image (Image gérée).

c. Pour Operating system (Système d'exploitation), choisissez Amazon Linux 2.

d. Pour Runtime(s) (Exécution(s)), sélectionnez Standard.

e. Pour Image, choisissez **`aws/codebuild/amazonlinux2-x86_64-standard:3.0`**.

f. Pour la Image version (Version d'image) et le Environment type (Type d'environnement), utilisez les valeurs par défaut.

g. Sélectionnez Enable this flag if you want to build Docker images or want your builds to get elevated privileges (Activer cet indicateur si vous souhaitez créer des images Docker ou que vos builds reçoivent des privilèges élevés).

h. Désélectionnez les CloudWatch journaux. Il se peut que vous deviez développer la fonctionnalité Avancé.

i. Choisissez Continuer vers CodePipeline.

j. Choisissez Suivant.

> **Note**
>
> L'assistant crée un rôle CodeBuild de service pour votre projet de build, appelé codebuild- ***`build-project-name`*** -service-role. Notez le nom de ce rôle lorsque vous y ajouterez des autorisations Amazon ECR ultérieurement.

8. Sur la page Étape 5 : Ajouter une phase de déploiement, pour Fournisseur de déploiement, choisissez Amazon ECS.

   a. Pour Nom du cluster, choisissez le cluster Amazon ECS dans lequel votre service est exécuté. Pour ce didacticiel, le cluster est default.

   b. Pour Nom du service, choisissez le service à mettre à jour, puis Étape suivante. Pour ce didacticiel, le nom de service est hello-world.

9. Sur la page Étape 6 : Vérification, vérifiez la configuration de votre pipeline, puis choisissez Créer un pipeline pour créer le pipeline.

> **Note**
>
> Maintenant que le pipeline est créé, il essaie d'exécuter les différentes phases. Cependant, le CodeBuild rôle par défaut créé par l'assistant n'est pas autorisé à exécuter toutes les commandes contenues dans le `buildspec.yml` fichier, de sorte que la

phase de génération échoue. Les autorisations nécessaires à la phase de génération sont ajoutées dans la section suivante.

Étape 3 : ajouter des autorisations Amazon ECR au rôle CodeBuild

L' CodePipeline assistant a créé un rôle IAM pour le projet de génération, appelé CodeBuild codebuild- ***`build-project-name`*** -service-role. Pour ce didacticiel, le nom est codebuild-hello-world-service-role. Étant donné que le `buildspec.yml` fichier appelle les opérations de l'API Amazon ECR, le rôle doit disposer d'une politique qui autorise les autorisations nécessaires pour effectuer ces appels Amazon ECR. La procédure suivante vous aide à attacher les autorisations adéquates au rôle.

Pour ajouter des autorisations Amazon ECR au rôle CodeBuild

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Dans la zone de recherche, tapez codebuild- et choisissez le rôle créé par l' CodePipeline assistant. Pour ce didacticiel, le nom de rôle est code-build-hello-world-service-role.
4. Sur la page Summary (Récapitulatif), choisissez Attach policies (Attacher les stratégies).
5. Cochez la case située à gauche de la AmazonEC2ContainerRegistryPowerUser politique, puis choisissez Joindre la politique.

Étape 4 : Test de votre pipeline

Votre pipeline doit disposer de tous les éléments nécessaires pour exécuter un déploiement AWS continu natif de bout en bout. À présent, testez sa fonctionnalité en transmettant une modification de code à votre référentiel source.

Pour tester votre pipeline

1. Apportez une modification de code au référentiel source que vous avez configuré, validez-la, puis transmettez-la.
2. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
3. Choisissez votre pipeline dans la liste.

4. Observez la progression du pipeline dans ses différentes phases. Votre pipeline devrait être terminé et votre service Amazon ECS exécute l'image Docker créée à partir de votre modification de code.

Tutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR

Dans ce didacticiel, vous allez configurer un pipeline AWS CodePipeline qui déploie des applications de conteneur à l'aide d'un blue/green déploiement qui prend en charge les images Docker. Lors d'un blue/green déploiement, vous pouvez lancer la nouvelle version de votre application parallèlement à l'ancienne version et tester la nouvelle version avant de rediriger le trafic. Vous pouvez également surveiller le processus de déploiement et revenir rapidement en arrière en cas de problème.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

> **Note**
>
> Ce didacticiel est destiné à l'action de CodeDeploy blue/green déploiement d'Amazon ECS CodePipeline. Pour un didacticiel utilisant l'action de déploiement standard d'Amazon ECS dans CodePipeline, consultezTutoriel : Déploiement standard d'Amazon ECS avec CodePipeline.

Le pipeline terminé détecte les modifications apportées à votre image, qui est stockée dans un référentiel d'images tel qu'Amazon ECR, et utilisée CodeDeploy pour acheminer et déployer le trafic vers un cluster Amazon ECS et un équilibreur de charge. CodeDeploy utilise un écouteur pour rediriger le trafic vers le port du conteneur mis à jour spécifié dans le AppSpec fichier. Pour plus d'informations sur la manière dont l'équilibreur de charge, l'écouteur de production, les groupes cibles

et votre application Amazon ECS sont utilisés dans un blue/green déploiement, consultez Tutoriel : Déployer un service Amazon ECS.

Le pipeline est également configuré pour utiliser un emplacement source, tel que CodeCommit l'endroit où votre définition de tâche Amazon ECS est stockée. Dans ce didacticiel, vous configurez chacune de ces AWS ressources, puis vous créez votre pipeline avec des étapes contenant des actions pour chaque ressource.

Votre pipeline de diffusion continue créera et déploiera automatiquement des images de conteneurs chaque fois que le code source est modifié ou qu'une nouvelle image de base est chargée sur Amazon ECR.

Ce flux utilise les artefacts suivants :

- Fichier image Docker qui spécifie le nom du conteneur et l'URI du référentiel de votre référentiel d'images Amazon ECR.
- Une définition de tâche Amazon ECS qui répertorie le nom de votre image Docker, le nom du conteneur, le nom du service Amazon ECS et la configuration de l'équilibreur de charge.
- CodeDeploy AppSpec Fichier qui spécifie le nom du fichier de définition des tâches Amazon ECS, le nom du conteneur de l'application mise à jour et le port de conteneur sur lequel CodeDeploy redirige le trafic de production. Il peut également spécifier la configuration du réseau et les fonctions Lambda facultatifs que vous pouvez exécuter durant les hooks d'événement de cycle de vie de déploiement.

> **Note**
>
> Lorsque vous validez une modification dans votre référentiel d'images Amazon ECR, l'action source du pipeline crée un `imageDetail.json` fichier pour cette validation. Pour de plus amples informations concernant le fichier `imageDetail.json`, veuillez consulter Detail.json fichier image pour les actions de blue/green déploiement d'Amazon ECS.

Lorsque vous créez ou modifiez votre pipeline et mettez à jour ou spécifiez des artefacts source pour votre étape de déploiement, assurez-vous pour qu'il pointe vers la source des artefacts avec le nom et la version les plus récents que vous souhaitez utiliser. Une fois que vous avez configuré votre pipeline, au fur et à mesure que vous apportez des modifications à votre image ou à une définition de tâche, il se peut que vous ayez besoin de mettre à jour vos fichiers d'artefact source dans vos référentiels, puis de modifier l'étape de déploiement dans votre pipeline.

Rubriques

- Conditions préalables
- Étape 1 : Création d'une image et transfert vers un référentiel Amazon ECR
- Étape 2 : créer une définition de tâche et des fichiers AppSpec sources et les transférer vers un CodeCommit référentiel
- Étape 3 : Créer votre équilibreur de charge d'application et vos groupes cibles
- Étape 4 : Création de votre cluster et de votre service Amazon ECS
- Étape 5 : Création de votre CodeDeploy application et de votre groupe de déploiement (plateforme de calcul ECS)
- Étape 6 : Créer le pipeline
- Étape 7 : Modifier le pipeline et vérifier le déploiement

Conditions préalables

Vous devez avoir déjà créé les ressources suivantes :

- Un CodeCommit référentiel. Vous pouvez utiliser le AWS CodeCommit référentiel que vous avez créé dansTutoriel : Création d'un pipeline simple (CodeCommit référentiel).
- Lancez une instance Amazon EC2 Linux et installez Docker pour créer une image, comme indiqué dans ce didacticiel. Si vous avez déjà une image que vous souhaitez utiliser, vous pouvez ignorer cette condition préalable.

Étape 1 : Création d'une image et transfert vers un référentiel Amazon ECR

Dans cette section, vous utilisez Docker pour créer une image, puis utilisez le AWS CLI pour créer un référentiel Amazon ECR et envoyer l'image vers le référentiel.

> **Note**
>
> Si vous avez déjà une image que vous souhaitez utiliser, vous pouvez ignorer cette étape.

Pour créer une image

1. Connectez-vous à votre instance Linux dans laquelle Docker est installé.

Retirez une image pour `nginx`. Cette commande fournit l'`nginx:latest`image :

```
docker pull nginx
```

2. Exécutez docker images. Vous devez voir l'image dans la liste.

```
docker images
```

Pour créer un référentiel Amazon ECR et diffuser votre image

1. Créez un référentiel Amazon ECR pour stocker votre image . Notez l'`repositoryUri` dans la sortie.

```
aws ecr create-repository --repository-name nginx
```

Sortie :

```
{
    "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. Balisez l'image avec la valeur `repositoryUri` de l'étape précédente.

```
docker tag nginx:latest aws_account_id.dkr.ecr.us-east-1.amazonaws.com/nginx:latest
```

3. Exécutez la aws ecr get-login-password commande, comme indiqué dans cet exemple pour la `us-west-2` région et l'ID de compte 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. Transférez l'image vers Amazon ECR à l'aide `repositoryUri` de l'étape précédente.

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

Étape 2 : créer une définition de tâche et des fichiers AppSpec sources et les transférer vers un CodeCommit référentiel

Dans cette section, vous allez créer un fichier JSON de définition de tâche et l'enregistrer auprès d'Amazon ECS. Vous créez ensuite un AppSpec fichier pour CodeDeploy et utilisez votre client Git pour transférer les fichiers vers votre CodeCommit référentiel.

Pour créer une définition de tâche pour votre image

1. Créez un fichier nommé `taskdef.json` avec les contenus suivants. Pour `image`, saisissez votre nom d'image, par exemple nginx. Cette valeur est mise à jour lorsque votre pipeline est exécuté.

   > **Note**
   >
   > Assurez-vous que le rôle d'exécution spécifié dans la définition de tâche contient la stratégie `AmazonECSTaskExecutionRolePolicy`. Pour plus d'informations, consultez Amazon ECS Task Execution IAM Role dans le manuel Amazon ECS Developer Guide.

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "name": "sample-website",
            "image": "nginx",
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
```

```
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512",
    "family": "ecs-demo"
}
```

2. Enregistrez votre définition de tâche dans le fichier `taskdef.json`.

```
aws ecs register-task-definition --cli-input-json file://taskdef.json
```

3. Une fois la définition de tâche enregistrée, modifiez votre fichier afin de supprimer le nom et d'inclure le texte d'espace réservé <IMAGE1_NAME> dans le champ de l'image.

```
{
    "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"
}
```

Pour créer un AppSpec fichier

- Le AppSpec fichier est utilisé pour les CodeDeploy déploiements. Le fichier, qui inclut des champs facultatifs, utilise ce format :

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "task-definition-ARN"
        LoadBalancerInfo:
          ContainerName: "container-name"
          ContainerPort: container-port-number
# Optional properties
        PlatformVersion: "LATEST"
        NetworkConfiguration:
            AwsvpcConfiguration:
              Subnets: ["subnet-name-1", "subnet-name-2"]
              SecurityGroups: ["security-group"]
              AssignPublicIp: "ENABLED"
Hooks:
- BeforeInstall: "BeforeInstallHookFunctionName"
- AfterInstall: "AfterInstallHookFunctionName"
- AfterAllowTestTraffic: "AfterAllowTestTrafficHookFunctionName"
- BeforeAllowTraffic: "BeforeAllowTrafficHookFunctionName"
- AfterAllowTraffic: "AfterAllowTrafficHookFunctionName"
```

Pour plus d'informations sur le AppSpec fichier, y compris des exemples, voir Référence de CodeDeploy AppSpec fichier.

Créez un fichier nommé `appspec.yaml` avec les contenus suivants. Pour `TaskDefinition`, ne modifiez pas le texte d'espace réservé `<TASK_DEFINITION>`. Cette valeur est mise à jour lorsque votre pipeline est exécuté.

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: <TASK_DEFINITION>
        LoadBalancerInfo:
```

```
          ContainerName: "sample-website"
          ContainerPort: 80
```

Pour transférer des fichiers vers votre CodeCommit référentiel

1. Transférez ou chargez les fichiers dans votre CodeCommit référentiel. Ces fichiers constituent l'artefact source créé par l'assistant Create Pipeline (Création de pipeline) pour votre action de déploiement dans CodePipeline. Vos fichiers doivent être similaires à ce qui suit dans votre répertoire local :

```
/tmp
  |my-demo-repo
    |-- appspec.yaml
    |-- taskdef.json
```

2. Choisissez la méthode que vous souhaitez utiliser pour charger vos fichiers :

   a. Pour utiliser la ligne de commande Git à partir d'un référentiel cloné sur votre ordinateur local :

      i. Modifiez les répertoires vers votre référentiel local :

```
(For Linux, macOS, or Unix) cd /tmp/my-demo-repo
(For Windows) cd c:\temp\my-demo-repo
```

      ii. Exécutez la commande suivante pour organiser tous vos fichiers à la fois :

```
git add -A
```

      iii. Exécutez la commande suivante pour valider les fichiers avec un message de validation :

```
git commit -m "Added task definition files"
```

      iv. Exécutez la commande suivante pour transférer les fichiers de votre dépôt local vers votre CodeCommit référentiel :

```
git push
```

   b. Pour télécharger vos fichiers à l' CodeCommit aide de la console, procédez comme suit :

i. Ouvrez la CodeCommit console et choisissez votre référentiel dans la liste des référentiels.

ii. Choisissez Ajouter un fichier, puis choisissez Charger le fichier.

iii. Sélectionnez Choisir un fichier, puis recherchez votre fichier. Validez la modification en entrant votre nom d'utilisateur et votre adresse e-mail. Choisissez Valider les modifications.

iv. Répétez cette étape pour chaque fichier que vous souhaitez charger.

Étape 3 : Créer votre équilibreur de charge d'application et vos groupes cibles

Dans cette section, vous allez créer un équilibreur de charge d'application Amazon EC2. Vous utilisez les noms de sous-réseaux et les valeurs de groupes cibles que vous créez avec votre équilibreur de charge ultérieurement, lorsque vous créez votre service Amazon ECS. Vous pouvez créer un équilibreur de charge d'application ou un Network Load Balancer. L'équilibreur de charge doit utiliser un VPC avec deux sous-réseaux publics dans différentes zones de disponibilité. Au cours de ces étapes, vous devez confirmer votre VPC par défaut, créer un équilibreur de charge, puis créer deux groupes cible pour votre équilibreur de charge. Pour plus d'informations, consultez Groupes cibles pour vos équilibreurs de charge du réseau.

Pour vérifier votre VPC par défaut et les sous-réseaux publics

1. Connectez-vous au Console de gestion AWS et ouvrez la console Amazon VPC à https://console.aws.amazon.com/vpc/ l'adresse.
2. Vérifiez le VPC par défaut à utiliser. Dans le panneau de navigation, sélectionnez Your VPCs (Vos VPC). Notez que le VPC affiche Oui dans la colonne VPC par défaut. Il s'agit du VPC par défaut. Il contient des sous-réseaux par défaut que vous pouvez sélectionner.
3. Choisissez Sous-réseaux. Choisissez deux sous-réseaux qui contiennent Oui dans la colonne Sous-réseau par défaut.

> **Note**
>
> Prenez note de vos ID de sous-réseaux. Vous en aurez besoin ultérieurement dans ce didacticiel.

4. Choisissez les sous-réseaux, puis choisissez l'onglet Description. Vérifiez que les sous-réseaux que vous souhaitez utiliser sont dans des zones de disponibilité différentes.
5. Choisissez les sous-réseaux, puis choisissez l'onglet Table de routage. Pour vérifier que chaque sous-réseau que vous souhaitez utiliser est un sous-réseau public, vérifiez qu'une ligne de passerelle est incluse dans la table de routage.

Pour créer un équilibreur de charge d'application Amazon EC2

1. Connectez-vous à Console de gestion AWS et ouvrez la console Amazon EC2 à https://console.aws.amazon.com/ec2/ l'adresse.
2. Dans le volet de navigation, choisissez Load Balancers.
3. Sélectionnez Create Load Balancer (Créer un équilibreur de charge).
4. Choisissez Équilibreur de charge d'application, puis choisissez Créer.
5. Dans Nom, entrez le nom de votre équilibreur de charge.
6. Dans Méthode, choisissez Accessible sur Internet.
7. Dans Type d'adresse IP, choisissez ipv4.
8. Configurez deux ports d'écoute pour votre équilibreur de charge :
    a. Sous Protocole d'écoute de l'équilibreur de charge, choisissez HTTP. Sous Port de l'équilibreur de charge, entrez **`80`**.
    b. Choisissez Add listener (Ajouter un écouteur).
    c. Sous Protocole de l'équilibreur de charge pour le deuxième écouteur, choisissez HTTP. Sous Port de l'équilibreur de charge, entrez **`8080`**.
9. Sous Zones de disponibilité, dans VPC, choisissez le VPC par défaut. Ensuite, choisissez deux sous-réseaux par défaut que vous souhaitez utiliser.
10. Choisissez Next: Configure Security Settings (Suivant : Configurer les paramètres de sécurité).
11. Choisissez Next: Configure Security Groups (Suivant : Configurer des groupes de sécurité).
12. Choisissez Sélectionner un groupe de sécurité existant et prenez note de l'ID du groupe de sécurité.
13. Choisissez Next: Configure Routing (Suivant :Configurer le routage).
14. Dans Groupe cible, choisissez Nouveau groupe cible et configurez votre premier groupe cible :
    a. Dans Nom, entrez un nom de groupe cible (par exemple, **`target-group-1`**).
    b. Dans Type cible, choisissez IP.

c. Dans Protocole, choisissez HTTP. Dans Port, entrez **80**.

d. Choisissez Next: Register Targets (Suivant : Enregistrer des cibles).

15. Choisissez Suivant : Vérification, puis Créer.

Pour créer un deuxième groupe cible pour votre équilibreur de charge

1. Une fois votre équilibreur de charge configuré, ouvrez la console Amazon EC2. Dans le volet de navigation, sélectionnez Groupes cibles.
2. Sélectionnez Créer un groupe cible.
3. Dans Nom, entrez un nom de groupe cible (par exemple, **target-group-2**).
4. Dans Type cible, choisissez IP.
5. Dans Protocole, choisissez HTTP. Dans Port, entrez **8080**.
6. Dans VPC, choisissez le VPC par défaut.
7. Choisissez Créer.

> **Note**
>
> Vous devez avoir deux groupes cibles créés pour votre équilibreur de charge afin de permettre l'exécution du déploiement. Vous devez uniquement noter l'ARN de votre premier groupe cible. Cet ARN est utilisé dans le fichier JSON `create-service` au cours de l'étape suivante.

Pour mettre à jour votre équilibreur de charge de façon à inclure votre deuxième groupe cible

1. Ouvrez la console Amazon EC2. Dans le volet de navigation, choisissez Load Balancers.
2. Choisissez l'équilibreur de charge, puis choisissez l'onglet Ecouteurs. Choisissez l'écouteur avec le port 8080, puis choisissez Modifier.
3. Choisissez l'icône en forme de crayon en regard de Réacheminer vers. Choisissez votre deuxième groupe cible, puis choisissez la coche. Choisissez Mettre à jour pour mettre à jour les mises à jour.

Étape 4 : Création de votre cluster et de votre service Amazon ECS

Dans cette section, vous allez créer un cluster et un service Amazon ECS qui CodeDeploy achemine le trafic pendant le déploiement (vers un cluster Amazon ECS plutôt que vers des instances EC2). Pour créer votre service Amazon ECS, vous devez utiliser les noms de sous-réseaux, le groupe de sécurité et la valeur du groupe cible que vous avez créés avec votre équilibreur de charge pour créer votre service.

> **Note**
>
> Lorsque vous suivez ces étapes pour créer votre cluster Amazon ECS, vous utilisez le modèle de cluster Networking only, qui provisionne les conteneurs AWS Fargate. AWS Fargate est une technologie qui gère votre infrastructure d'instances de conteneurs pour vous. Il n'est pas nécessaire de choisir ou de créer manuellement des instances Amazon EC2 pour votre cluster Amazon ECS.

Pour créer un nouveau cluster Amazon ECS

1. Ouvrez la console classique Amazon ECS à l'adresse https://console.aws.amazon.com/ecs/.
2. Dans le panneau de navigation, choisissez Clusters.
3. Choisissez Créer un cluster.
4. Choisissez le modèle de cluster réseau uniquement qui utilise AWS Fargate, puis choisissez Étape suivante.
5. Saisissez le nom d'un cluster sur la page Configure cluster (Configurer le cluster). Vous pouvez ajouter une balise facultative pour votre ressource. Choisissez Créer.

Pour créer un service Amazon ECS

Utilisez le AWS CLI pour créer votre service dans Amazon ECS.

1. Créez un fichier JSON et nommez-le `create-service.json`. Collez ce qui suit dans le fichier JSON.

   Pour `taskDefinition` ce champ, lorsque vous enregistrez une définition de tâche dans Amazon ECS, vous lui attribuez une famille. Il s'agit d'un élément similaire à un nom pour plusieurs versions de la définition de tâche, spécifié avec un numéro de révision. Dans cet exemple, utilisez « `ecs-demo:1` » comme famille et numéro de révision dans votre fichier.

Utilisez les noms des sous-réseau, le groupe de sécurité et la valeur du groupe cible créés avec l'équilibreur de charge dans Étape 3 : Créer votre équilibreur de charge d'application et vos groupes cibles.

> **Note**
>
> Vous devez inclure votre ARN de groupe cible dans ce fichier. Ouvrez la console Amazon EC2 et, dans le volet de navigation, sous LOAD BALANCING, sélectionnez Target Groups. Choisissez votre premier groupe cible. Copiez votre ARN à partir de l'onglet Description.

```
{
    "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. Exécutez la commande create-service, en spécifiant le fichier JSON :

> **Important**
>
> N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

Cet exemple crée un service nommé `my-service`.

> **Note**
>
> Cet exemple de commande crée un service nommé my-service. Si vous avez déjà un service portant ce nom, la commande renvoie une erreur.

```
aws ecs create-service --service-name my-service --cli-input-json file://create-service.json
```

La sortie renvoie les champs de description pour votre service.

3. Exécutez la commande describe-services pour vérifier que le service a été créé.

```
aws ecs describe-services --cluster cluster-name --services service-name
```

Étape 5 : Création de votre CodeDeploy application et de votre groupe de déploiement (plateforme de calcul ECS)

Lorsque vous créez une CodeDeploy application et un groupe de déploiement pour la plate-forme de calcul Amazon ECS, l'application est utilisée lors d'un déploiement pour référencer le groupe de déploiement, les groupes cibles, les auditeurs et le comportement de redirection du trafic appropriés.

Pour créer une CodeDeploy application

1. Ouvrez la CodeDeploy console et choisissez Créer une application.
2. Dans Nom de l'application, entrez le nom que vous souhaitez utiliser.
3. Dans Plateforme de calcul, choisissez Amazon ECS.

4. Choisissez Créer une application.

Pour créer un groupe CodeDeploy de déploiement

1. Sur la page de votre application, dans l'onglet Groupe de déploiement, choisissez Créer un groupe de déploiement.
2. Dans Nom du groupe de déploiement, entrez un nom décrivant le groupe de déploiement.
3. Dans Rôle de service, choisissez un rôle de service qui accorde CodeDeploy l'accès à Amazon ECS. Pour créer un rôle de service, procédez comme suit :
   a. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
   b. Dans le tableau de bord de la console, choisissez Rôles.
   c. Choisissez Créer un rôle.
   d. Sous Sélectionner le type d'entité de confiance, sélectionnez Service AWS. Sous Choisir un cas d'utilisation, sélectionnez CodeDeploy. Dans Sélectionnez votre cas d'utilisation, sélectionnez CodeDeploy - ECS. Choisissez Suivant : Autorisations. La stratégie gérée `AWSCodeDeployRoleForECS` est déjà attachée au rôle.
   e. Choisissez Suivant : Balises, puis Suivant : Vérification.
   f. Entrez un nom pour le rôle (par exemple, **`CodeDeployECSRole`**), puis choisissez Créer un rôle.
4. Dans Configuration de l'environnement, choisissez le nom de votre cluster Amazon ECS et le nom du service.
5. Dans Équilibreurs de charge, choisissez le nom de l'équilibreur de charge qui achemine le trafic vers votre service Amazon ECS.
6. Dans Port du récepteur de production, choisissez le port et le protocole du récepteur qui achemine le trafic de production vers votre service Amazon ECS. Dans Test listener port (Port de l'écouteur de test), choisissez le port et le protocole de l'écouteur de test.
7. À partir de Nom du groupe cible 1 et Nom du groupe cible 2, choisissez les groupes cible utilisés pour acheminer le trafic au cours de votre déploiement. Assurez-vous que ce sont les groupes cibles que vous avez créés pour votre équilibreur de charge.
8. Choisissez Réacheminer le trafic immédiatement pour déterminer combien de temps après un déploiement réussi vous devez rediriger le trafic vers votre tâche Amazon ECS mise à jour.
9. Choisissez Créer un groupe de déploiement.

Étape 6 : Créer le pipeline

Dans cette section, vous créez un pipeline avec les actions suivantes :

- CodeCommit Action dont les artefacts sources sont la définition de la tâche et le AppSpec fichier.
- Stage source avec une action source Amazon ECR où l'artefact source est le fichier image.
- Étape de déploiement avec une action de déploiement Amazon ECS au cours de laquelle le déploiement s'exécute avec une CodeDeploy application et un groupe de déploiement.

Pour créer un pipeline à deux étapes avec l'assistant

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyImagePipeline`**.
5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.
6. Dans Rôle de service, choisissez Nouveau rôle de service pour permettre CodePipeline de créer un rôle de service dans IAM.
7. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
8. À l'étape 3 : Ajouter une étape source, dans Source provider, choisissez AWS CodeCommit. Dans Nom du référentiel, choisissez le nom du référentiel CodeCommit que vous avez créé dans Étape 1 : Création d'un CodeCommit référentiel. Dans Nom de branche, choisissez le nom de la branche qui contient votre dernière mise à jour du code.

   Choisissez Suivant.
9. À l'étape 4 : Ajouter une étape de construction, choisissez Ignorer l'étape de génération, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer. Choisissez Suivant.

10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.

11. À l'étape 6 : ajouter une étape de déploiement :

    a. Dans Deploy provider, choisissez Amazon ECS (Blue/Green). Dans Nom de l'application, entrez ou choisissez le nom de l'application dans la liste, comme `codedeployapp`. Dans Groupe de déploiement, saisissez ou choisissez le nom du groupe de déploiement dans la liste, comme `codedeploydeplgroup`.

    > **Note**
    >
    > « Déploiement » est le nom donné par défaut à l'étape du pipeline créée lors de l'Étape 4 : Déploiement tout comme « Source » est le nom donné à la première étape du pipeline.

    b. Dans Définition de tâche Amazon ECS, choisissez SourceArtifact. Dans le champ, entrez **`taskdef.json`**.

    c. Dans AWS CodeDeploy AppSpec Fichier, sélectionnez SourceArtifact. Dans le champ, entrez **`appspec.yaml`**.

    > **Note**
    >
    > À ce stade, ne renseignez aucune information sous Mettre à jour l'image de la définition de tâche de manière dynamique.

    d. Choisissez Suivant.

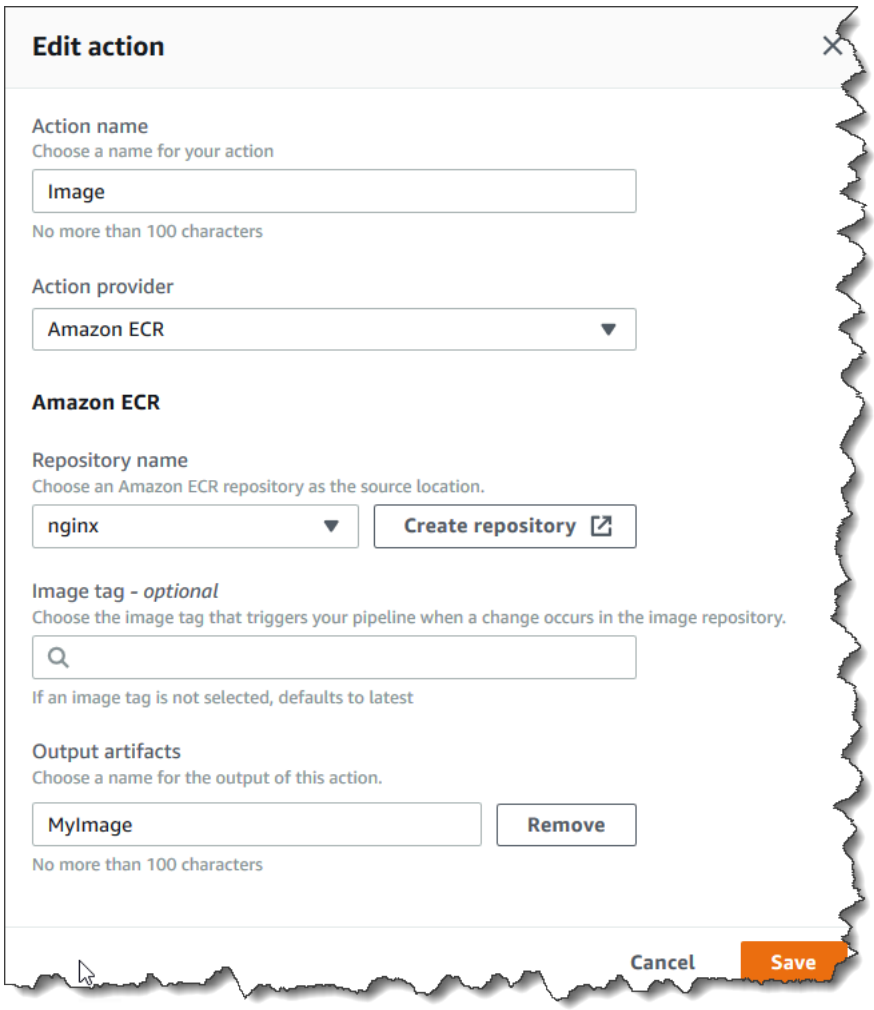
12. À l'étape 7 : Passez en revue, passez en revue les informations, puis choisissez Créer un pipeline.

Pour ajouter une action source Amazon ECR à votre pipeline

Consultez votre pipeline et ajoutez une action source Amazon ECR à votre pipeline.

1. Choisissez votre pipeline. Dans le coin supérieur gauche, choisissez Modifier.
2. Dans la phase source, choisissez Modifier l'étape.

3. Ajoutez une action parallèle en choisissant + Ajouter une action à côté de votre action CodeCommit source.
4. Dans Nom d'action, saisissez un nom pour le pipeline (par exemple, **`Image`**).
5. Dans Fournisseur d'action, choisissez Amazon ECR.



6. Dans Nom du référentiel, choisissez le nom de votre référentiel Amazon ECR.
7. Dans Balise d'image, spécifiez le nom et la version de l'image, s'ils sont différents des précédents.
8. Dans Artefacts de sortie, choisissez l'artefact de sortie par défaut (par exemple, `MyImage`) qui contient les informations de nom d'image et d'URI de référentiel que l'étape suivante doit utiliser.
9. Choisissez Save (Enregistrer) sur l'écran de l'action. Choisissez Done (Terminé) sur l'écran de l'étape. Choisissez Save (Enregistrer) sur le pipeline. Un message indique la règle Amazon CloudWatch Events à créer pour l'action source Amazon ECR.

Pour relier vos artefacts source à l'action de déploiement

1. Choisissez Modifier sur votre phase de déploiement et choisissez l'icône pour modifier l'action Amazon ECS (Blue/Green).
2. Faites défiler jusqu'au bas du panneau. Dans Artefacts d'entrée, choisissez Ajouter. Ajoutez l'artefact source à partir de votre nouveau référentiel Amazon ECR (par exemple,`MyImage`).
3. Dans Définition de la tâche, choisissez SourceArtifact, puis vérifiez que **`taskdef.json`** c'est saisi.
4. Dans AWS CodeDeploy AppSpec Fichier, choisissez SourceArtifact, puis vérifiez si la saisie **`appspec.yaml`** est effectuée.
5. Dans Mettre à jour dynamiquement l'image de définition de tâche, dans Artefact d'entrée avec URI d'image MyImage, choisissez, puis entrez le texte de l'espace réservé utilisé dans le `taskdef.json` fichier :. **`IMAGE1_NAME`** Choisissez Enregistrer.
6. Dans le AWS CodePipeline volet, choisissez Enregistrer la modification du pipeline, puis sélectionnez Enregistrer la modification. Affichez votre pipeline mis à jour.

   Une fois cet exemple de pipeline créé, la configuration d'action pour les entrées de la console s'affiche dans la structure du pipeline, comme suit :

   ```
   "configuration": {
     "AppSpecTemplateArtifact": "SourceArtifact",
     "AppSpecTemplatePath": "appspec.yaml",
     "TaskDefinitionTemplateArtifact": "SourceArtifact",
     "TaskDefinitionTemplatePath": "taskdef.json",
     "ApplicationName": "codedeployapp",
     "DeploymentGroupName": "codedeploydeplgroup",
     "Image1ArtifactName": "MyImage",
     "Image1ContainerName": "IMAGE1_NAME"
   },
   ```
7. Pour soumettre vos modifications et lancer la génération d'un pipeline, choisissez Changement de version, puis Publication.
8. Choisissez l'action de déploiement dans laquelle vous souhaitez l'afficher CodeDeploy et suivre la progression du transfert de trafic.

> **Note**
>
> Il se peut qu'une étape de déploiement affiche un temps d'attente facultatif. Par défaut, il CodeDeploy attend une heure après un déploiement réussi avant de mettre fin à l'ensemble de tâches d'origine. Vous pouvez utiliser ce temps pour revenir en arrière ou mettre fin à la tâche. Sinon, votre déploiement prend fin lorsque l'ensemble de tâches est terminé.

Étape 7 : Modifier le pipeline et vérifier le déploiement

Apportez une modification à votre image, puis transférez la modification à votre référentiel Amazon ECR. Cela déclenche l'exécution de votre pipeline. Vérifiez que la modification de la source de l'image est déployée.

Didacticiel : Création d'un pipeline qui déploie un kit Amazon Alexa Skill

Dans ce didacticiel, vous configurez un pipeline qui diffuse en continu votre compétence Alexa à l'aide du kit Alexa Skills en tant que fournisseur de déploiement dans votre étape de déploiement. Le pipeline terminé détecte les modifications apportées à votre compétence lorsque vous modifiez les fichiers source dans votre référentiel source. Le pipeline utilise ensuite le kit Alexa Skills pour procéder au déploiement sur l'étape de développement de la compétence Alexa.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé CodePipeline pour les artefacts. (Ceci est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient à cette personne sûre et Comptes AWS qu'elle sera fiable.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans la région Asie-Pacifique (Hong Kong) ou en Europe (Milan). Pour utiliser les autres actions de déploiement disponibles dans cette région, consultezIntégrations d'actions de déploiement.

Pour créer votre compétence personnalisée en tant que fonction Lambda, voir Héberger une compétence personnalisée en tant que fonction Lambda AWS. Vous pouvez également créer un pipeline qui utilise des fichiers source Lambda et un CodeBuild projet pour déployer les modifications apportées à Lambda en fonction de vos compétences.

Conditions préalables

Vous devez déjà disposer des éléments suivants :

- Un CodeCommit référentiel. Vous pouvez utiliser le AWS CodeCommit référentiel dans lequel vous l'avez crééTutoriel : Création d'un pipeline simple (CodeCommit référentiel).
- Un compte de développeur Amazon. Ce compte est propriétaire de vos compétences Alexa. Vous pouvez créer un compte gratuitement depuis la page kit Alexa Skills.
- Une compétence Alexa. Vous pouvez créer un exemple de compétence à l'aide du didacticiel Obtention d'exemple de code de compétence personnalisé.
- Installez la CLI ASK et configurez-la à l'`ask init`aide de vos AWS informations d'identification. Veuillez consulter Installation et initialisation de l'interface de ligne de commande ASK.

Étape 1 : Créer un profil de sécurité LWA des services de développement Alexa

Dans cette section, vous allez créer un profil de sécurité à utiliser avec Login with Amazon (LWA). Si vous disposez déjà d'un profil, vous pouvez ignorer cette étape.

- Utilisez les étapes de generate-lwa-tokens pour créer un profil de sécurité.
- Après avoir créé le profil, notez l'ID client et la clé secrète du client.
- Assurez-vous de saisir le retour autorisé URLs comme indiqué dans les instructions. Ils URLs permettent à la commande ASK CLI de rediriger les demandes de jeton d'actualisation.

Étape 2 : créer des fichiers source de compétences Alexa et les transférer vers votre CodeCommit référentiel

Dans cette section, vous allez créer et transmettre vos fichiers source de compétence Alexa au référentiel que le pipeline utilise pour votre étape source. Pour la compétence que vous avez créée dans la console du développeur Amazon, vous créez et transmettez les éléments suivants :

- Un fichier `skill.json`.
- Un dossier `interactionModel/custom`.

> **Note**
>
> Cette structure de répertoires est conforme aux exigences de format du package Kit Alexa Skills, comme décrit dans Format de package de compétence. Si votre structure de répertoire n'utilise pas le bon format de package de compétences, les modifications ne sont pas déployées correctement dans la console du kit Alexa Skills.

Pour créer des fichiers source pour votre compétence

1. Récupérez votre ID de compétence à partir de la console du développeur du kit Alexa Skills. Utilisez cette commande :

   ```
   ask api list-skills
   ```

   Recherchez votre compétence par son nom, puis copiez l'ID associé dans le champ `skillId`.
2. Générez un fichier `skill.json` contenant les détails de votre compétence. Utilisez cette commande :

   ```
   ask api get-skill -s skill-ID > skill.json
   ```
3. (Facultatif) Créez un dossier `interactionModel/custom`.

   Utilisez cette commande pour générer le fichier de modèle d'interaction dans le dossier. Pour les paramètres régionaux, ce didacticiel utilise en-US comme paramètres régionaux dans le nom de fichier.

   ```
   ask api get-model --skill-id skill-ID --locale locale >
   ```

```
    ./interactionModel/custom/locale.json
```

Pour transférer des fichiers vers votre CodeCommit dépôt

1. Envoyez ou téléchargez les fichiers dans votre CodeCommit dépôt. Ces fichiers constituent l'artefact source créé par l'assistant Create Pipeline (Création de pipeline) pour votre action de déploiement dans AWS CodePipeline. Vos fichiers doivent être similaires à ce qui suit dans votre répertoire local :

```
skill.json
/interactionModel
  /custom
    |en-US.json
```

2. Choisissez la méthode que vous souhaitez utiliser pour charger vos fichiers :

   a. Pour utiliser la ligne de commande Git à partir d'un référentiel cloné sur votre ordinateur local :

      i. Exécutez la commande suivante pour organiser tous vos fichiers à la fois :

```
git add -A
```

      ii. Exécutez la commande suivante pour valider les fichiers avec un message de validation :

```
git commit -m "Added Alexa skill files"
```

      iii. Exécutez la commande suivante pour transférer les fichiers de votre dépôt local vers votre CodeCommit dépôt :

```
git push
```

   b. Pour télécharger vos fichiers à l' CodeCommit aide de la console, procédez comme suit :

      i. Ouvrez la CodeCommit console et choisissez votre dépôt dans la liste des référentiels.

      ii. Choisissez Ajouter un fichier, puis choisissez Charger le fichier.

iii. Sélectionnez Choisir un fichier, puis recherchez votre fichier. Validez la modification en entrant votre nom d'utilisateur et votre adresse e-mail. Choisissez Valider les modifications.

iv. Répétez cette étape pour chaque fichier que vous souhaitez charger.

Étape 3 : Utiliser les commandes CLI ASK pour créer un jeton d'actualisation

CodePipeline utilise un jeton d'actualisation basé sur l'ID client et le secret de votre compte de développeur Amazon pour autoriser les actions qu'il effectue en votre nom. Dans cette section, vous utilisez l'interface de ligne de commande ASK pour créer le jeton. Vous utilisez ces informations d'identification lorsque vous utilisez l'assistant Create Pipeline (Création de pipeline).

Pour créer un jeton d'actualisation avec vos informations d'identification de compte de développeur Amazon

1. Utilisez la commande suivante :

```
ask util generate-lwa-tokens
```

2. Lorsque vous y êtes invité, saisissez votre ID client et votre clé secrète, comme illustré dans l'exemple suivant :

```
? Please type in the client ID:
amzn1.application-client.example112233445566
? Please type in the client secret:
example112233445566
```

3. La page de connexion s'affiche. Connectez-vous avec les informations d'identification de votre compte de développeur Amazon.
4. Revenez à l'écran de ligne de commande. Le jeton d'accès et le jeton d'actualisation sont générés dans la sortie. Copiez le jeton d'actualisation renvoyé dans la sortie.

Étape 4 : Créer votre pipeline

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Une étape source avec une CodeCommit action dans laquelle les artefacts sources sont les fichiers de compétences Alexa qui soutiennent votre compétence.
- Une étape de déploiement avec une action de déploiement du kit Alexa Skills.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Choisissez la AWS région dans laquelle vous souhaitez créer le projet et ses ressources. L'exécution de la compétence Alexa est uniquement disponible dans les régions suivantes :
   - Asie-Pacifique (Tokyo)
   - Europe (Irlande)
   - USA Est (Virginie du Nord)
   - USA Ouest (Oregon)
3. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
4. Sur la page Étape 1 : Choisir une option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
5. À l'étape 2 : Choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyAlexaPipeline`**.
6. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul type que vous pouvez choisir dans la console. Pour plus d'informations, consultez la section Types de pipelines. Pour plus d'informations sur la tarification CodePipeline, consultez la section Tarification.
7. Dans Rôle de service, choisissez Nouveau rôle de service pour autoriser CodePipeline la création d'un rôle de service dans IAM.
8. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
9. À l'étape 3 : Ajouter une étape source, dans Source provider, choisissez AWS CodeCommit. Dans Nom du référentiel, choisissez le nom du CodeCommit référentiel dans lequel vous l'avez créé Étape 1 : Création d'un CodeCommit référentiel. Dans Nom de branche, choisissez le nom de la branche qui contient votre dernière mise à jour du code.

   Après avoir sélectionné le nom du référentiel et la branche, un message indique la règle Amazon CloudWatch Events à créer pour ce pipeline.

Choisissez Suivant.

10. À l'étape 4 : Ajouter une phase de construction, choisissez Ignorer l'étape de génération, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer.

    Choisissez Suivant.

11. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer l'étape de test, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer.

    Choisissez Suivant.

12. À l'étape 6 : Ajouter une phase de déploiement :

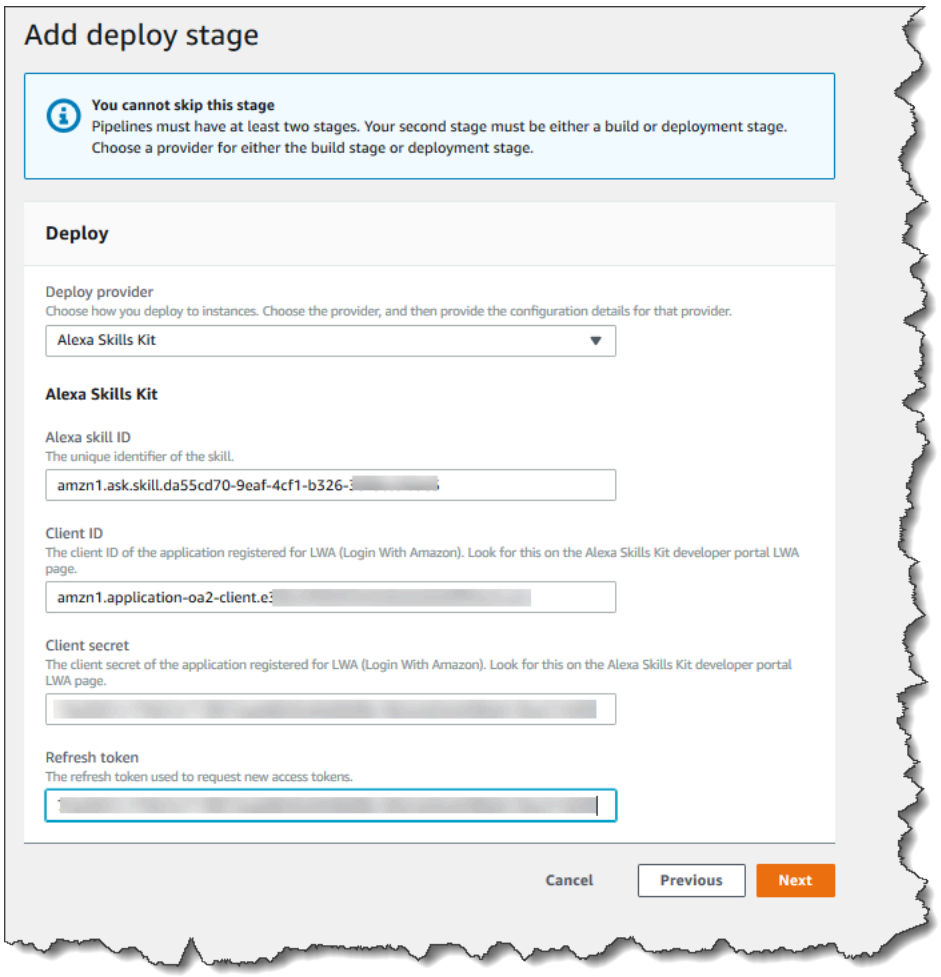
    a. Dans Deploy provider (Déployer le fournisseur), choisissez Alexa Skills Kit (Kit Alexa Skills).

    b. Dans Alexa skill ID (ID de compétence Alexa), saisissez l'ID de compétence affecté à votre compétence dans la console du développeur du kit Alexa Skills.

    c. Dans Client ID (ID client), saisissez l'ID de l'application que vous avez enregistrée.

    d. Dans Client secret (Clé secrète du client), saisissez la clé secrète que vous avez choisie lors de l'enregistrement.

    e. Dans Refresh token (Jeton d'actualisation), entrez le jeton que vous avez généré à l'étape 3.



f. Choisissez Suivant.

13. À l'étape 7 : Passez en revue, passez en revue les informations, puis choisissez Créer un pipeline.

Étape 5 : Modifier un fichier source et vérifier le déploiement

Apportez une modification à votre compétence, puis transmettez le changement à votre référentiel. Cela déclenche l'exécution de votre pipeline. Vérifiez que votre compétence est mise à jour dans la console de développeur du kit Alexa Skills.

Tutoriel : Création d'un pipeline qui utilise Amazon S3 comme fournisseur de déploiement

Dans ce didacticiel, vous allez configurer un pipeline qui fournit des fichiers en continu en utilisant Amazon S3 comme fournisseur d'actions de déploiement dans votre phase de déploiement. Le pipeline terminé détecte les modifications apportées lorsque vous modifiez les fichiers source dans votre référentiel source. Le pipeline utilise ensuite Amazon S3 pour déployer les fichiers dans votre compartiment. Chaque fois que vous modifiez ou ajoutez les fichiers de votre site Web dans votre emplacement source, le déploiement crée le site Web avec vos derniers fichiers.

Important

Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé CodePipeline pour les artefacts. (Ceci est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient à cette personne sûre et Comptes AWS qu'elle sera fiable.

Note

Même si vous supprimez des fichiers du référentiel source, l'action de déploiement S3 ne supprime pas les objets S3 correspondant aux fichiers supprimés.

Ce didacticiel fournit deux options :

- Création d'un pipeline qui déploie un site web statique dans votre compartiment S3 public. Cet exemple crée un pipeline avec une action AWS CodeCommit source et une action de déploiement Amazon S3. Consultez Option 1 : déployer des fichiers de site Web statiques sur Amazon S3.
- Créez un pipeline qui compile un exemple de TypeScript code JavaScript et déploie l'artefact de CodeBuild sortie dans votre compartiment S3 à des fins d'archivage. Cet exemple crée un pipeline avec une action source Amazon S3, une action de CodeBuild génération et une action de déploiement Amazon S3. Consultez Option 2 : déployer des fichiers d'archive créés sur Amazon S3 à partir d'un compartiment source S3.

> **Important**
>
> La plupart des actions que vous ajoutez à votre pipeline dans le cadre de cette procédure impliquent AWS des ressources que vous devez créer avant de créer le pipeline. AWS les ressources pour vos actions source doivent toujours être créées dans la même AWS région que celle où vous créez votre pipeline. Par exemple, si vous créez votre pipeline dans la région USA Est (Ohio), votre CodeCommit référentiel doit se trouver dans la région USA Est (Ohio).
>
> Vous pouvez ajouter des actions interrégionales lorsque vous créez votre pipeline. AWS les ressources pour les actions interrégionales doivent se trouver dans la même AWS région que celle où vous prévoyez d'exécuter l'action. Pour de plus amples informations, veuillez consulter Ajouter une action interrégionale dans CodePipeline.

Option 1 : déployer des fichiers de site Web statiques sur Amazon S3

Dans cet exemple, vous téléchargez l'exemple de fichier de modèle de site Web statique, vous chargez les fichiers dans votre AWS CodeCommit référentiel, vous créez votre compartiment et vous le configurez pour l'hébergement. Ensuite, vous utilisez la AWS CodePipeline console pour créer votre pipeline et spécifier une configuration de déploiement Amazon S3.

Conditions préalables

Vous devez déjà disposer des éléments suivants :

- Un CodeCommit référentiel. Vous pouvez utiliser le AWS CodeCommit référentiel dans lequel vous l'avez créé Tutoriel : Création d'un pipeline simple (CodeCommit référentiel).
- Fichiers source pour votre site web statique. Utilisez ce lien pour télécharger un exemple de site web statique. Le téléchargement du fichier sample-website.zip génère les fichiers suivants :
  - Un fichier `index.html`
  - Un fichier `main.css`
  - Un fichier `graphic.jpg`
- Un compartiment S3 configuré pour l'hébergement de site web. Veuillez consulter Hébergement d'un site web statique sur Amazon S3. Vérifiez que vous avez créé votre compartiment dans la même région que le pipeline.

> **Note**
>
> Pour héberger un site Web, le compartiment doit avoir un accès en lecture public, ce qui donne un accès en lecture à tout le monde. À l'exception de l'hébergement de site web, vous devez conserver les paramètres d'accès par défaut qui bloquent l'accès public aux compartiments S3.

Étape 1 : Transférez les fichiers source vers votre CodeCommit dépôt

Dans cette section, vous allez transmettre vos fichiers source au référentiel que le pipeline utilise pour votre étape source.

Pour transférer des fichiers vers votre CodeCommit dépôt

1. Extrayez les exemples de fichier téléchargés. Ne chargez pas le fichier ZIP dans votre référentiel.
2. Envoyez ou téléchargez les fichiers dans votre CodeCommit dépôt. Ces fichiers constituent l'artefact source créé par l'assistant Create Pipeline (Création de pipeline) pour votre action de déploiement dans CodePipeline. Vos fichiers doivent être similaires à ce qui suit dans votre répertoire local :

   ```
   index.html
   main.css
   graphic.jpg
   ```

3. Vous pouvez utiliser Git ou la CodeCommit console pour télécharger vos fichiers :

   a. Pour utiliser la ligne de commande Git à partir d'un référentiel cloné sur votre ordinateur local :

      i. Exécutez la commande suivante pour organiser tous vos fichiers à la fois :

      ```
      git add -A
      ```

      ii. Exécutez la commande suivante pour valider les fichiers avec un message de validation :

```
git commit -m "Added static website files"
```

iii. Exécutez la commande suivante pour transférer les fichiers de votre dépôt local vers votre CodeCommit dépôt :

```
git push
```

b. Pour télécharger vos fichiers à l' CodeCommit aide de la console, procédez comme suit :

i. Ouvrez la CodeCommit console et choisissez votre dépôt dans la liste des référentiels.

ii. Choisissez Ajouter un fichier, puis choisissez Charger le fichier.

iii. Sélectionnez Choose file (Choisir un fichier), puis naviguez vers votre fichier. Validez la modification en entrant votre nom d'utilisateur et votre adresse e-mail. Choisissez Valider les modifications.

iv. Répétez cette étape pour chaque fichier que vous souhaitez charger.

Étape 2 : Créer votre pipeline

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Un stage source avec une CodeCommit action dans laquelle les artefacts source sont les fichiers de votre site Web.
- Une étape de déploiement avec une action de déploiement Amazon S3.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Sur la page Bienvenue, Démarrez ou Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisir une option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : Choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyS3DeployPipeline`**.
5. Dans Type de pipeline, choisissez V2. Pour de plus amples informations, veuillez consulter Types de canalisations. Choisissez Suivant.

6. Dans Rôle de service, choisissez Nouveau rôle de service pour autoriser CodePipeline la création d'un rôle de service dans IAM.
7. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
8. À l'étape 3 : Ajouter une étape source, dans Source provider, choisissez AWS CodeCommit. Dans Nom du référentiel, choisissez le nom du référentiel CodeCommit que vous avez créé dans Étape 1 : Création d'un CodeCommit référentiel. Dans Nom de branche, choisissez le nom de la branche qui contient votre dernière mise à jour du code. À moins d'avoir créé une autre branche vous-même, seul `main` est disponible.

   Après avoir sélectionné le nom du référentiel et la branche, la règle Amazon CloudWatch Events à créer pour ce pipeline s'affiche.

   Choisissez Suivant.
9. À l'étape 4 : Ajouter une phase de construction, choisissez Ignorer l'étape de génération, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer.

   Choisissez Suivant.
10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer l'étape de test, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer.

    Choisissez Suivant.
11. À l'étape 6 : Ajouter une phase de déploiement :

    a. Dans Deploy provider (Fournisseur de déploiement), choisissez Amazon S3.

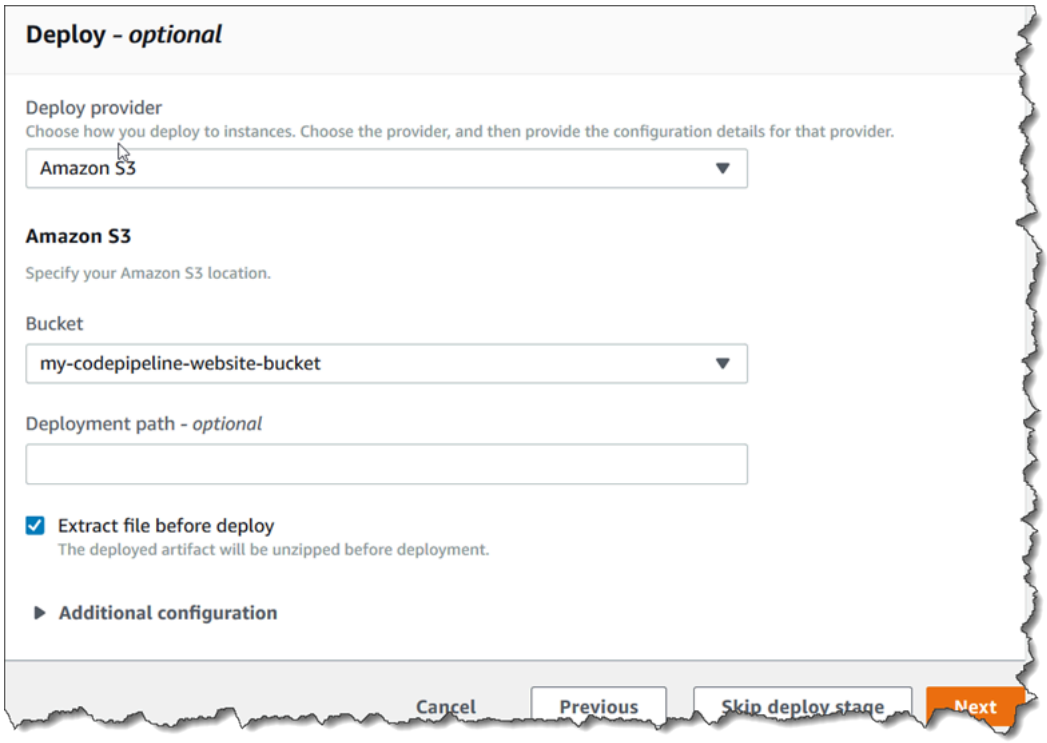
    b. Dans Bucket (Compartiment), indiquez le nom du compartiment public.

    c. Sélectionnez Extract file before deploy (Extraire le fichier avant de déployer).

    > **Note**
    >
    > Le déploiement échoue si vous ne sélectionnez pas Veuillez extraire le fichier avant le déploiement. Cela est dû au fait que l' AWS CodeCommit action de votre pipeline compresse les artefacts source et que votre fichier est un fichier ZIP.

    Lorsque Extract file before deploy (Extraire le fichier avant de déployer) est sélectionné, Deployment path (Chemin de déploiement) s'affiche. Entrez le nom du chemin que vous

souhaitez utiliser. Cela crée une structure de dossiers dans Amazon S3 dans laquelle les fichiers sont extraits. Dans le cadre de ce didacticiel, laissez ce champ vide.



d. (Facultatif) Dans Canned ACL (Liste ACL prête à l'emploi), vous pouvez appliquer un ensemble d'autorisations prédéfinies, appelées liste ACL prête à l'emploi, aux artefacts chargés.

e. (Facultatif) Dans Cache control (Contrôle de cache), entrez les paramètres de mise en cache. Vous pouvez définir cela pour contrôler le comportement de mise en cache pour les demandes/réponses. Pour connaître les valeurs valides, consultez le champ d'en-tête `Cache-Control` pour les opérations HTTP.

f. Choisissez Suivant.

12. À l'étape 7 : Passez en revue, passez en revue les informations, puis choisissez Créer un pipeline.

13. Une fois votre pipeline exécuté avec succès, ouvrez la console Amazon S3 et vérifiez que vos fichiers apparaissent dans votre compartiment public, comme indiqué :

```
index.html
main.css
graphic.jpg
```

14. Accédez à votre point de terminaison pour tester le site web. Votre point de terminaison est conforme au format suivant : `http://bucket-name.s3-website-region.amazonaws.com/`.

    Exemple de point de terminaison : `http://my-bucket.s3-website-us-west-2.amazonaws.com/`.

    L'exemple de page Web apparaît.

Étape 3 : Modifier un fichier source et vérifier le déploiement

Apportez une modification à vos fichiers source, puis transmettez le changement à votre référentiel. Cela déclenche l'exécution de votre pipeline. Vérifiez que votre site Web est mis à jour.

Option 2 : déployer des fichiers d'archive créés sur Amazon S3 à partir d'un compartiment source S3

Dans cette option, les commandes de construction de votre phase de construction compilent le TypeScript code en JavaScript code et déploient la sortie dans votre compartiment cible S3 dans un dossier horodaté distinct. Vous devez d'abord créer TypeScript du code et un fichier buildspec.yml. Après avoir combiné les fichiers source dans un fichier ZIP, vous téléchargez le fichier ZIP source dans votre compartiment source S3 et utilisez une CodeBuild étape pour déployer un fichier ZIP d'application intégré dans votre compartiment cible S3. Le code compilé est conservé en tant qu'archive dans votre compartiment cible.

Conditions préalables

Vous devez déjà disposer des éléments suivants :

- Un compartiment source S3. Vous pouvez utiliser le compartiment que vous avez créé dans Didacticiel : Création d'un pipeline simple (compartiment S3).
- Un compartiment cible S3. Veuillez consulter Hébergement d'un site web statique sur Amazon S3. Assurez-vous de créer votre bucket de la même manière Région AWS que le pipeline que vous souhaitez créer.

> **Note**
>
> Cet exemple illustre le déploiement de fichiers sur un compartiment privé. N'activez pas votre compartiment cible pour l'hébergement de site Web ou n'attachez aucune stratégie qui rendrait le compartiment public.

Étape 1 : Créer et charger des fichiers source sur votre compartiment source S3

Dans cette section, vous allez créer et charger vos fichiers source sur le compartiment que le pipeline utilise pour votre étape source. Cette section fournit des instructions pour créer les fichiers source suivants :

- Un `buildspec.yml` fichier, qui est utilisé pour les projets de CodeBuild construction.
- Un fichier `index.ts`.

Pour créer un fichier buildspec.yml

- Créez un fichier nommé `buildspec.yml` avec les contenus suivants. Ces commandes de compilation installent TypeScript et utilisent le TypeScript compilateur pour réécrire le code `index.ts` en JavaScript code.

```
version: 0.2

phases:
  install:
    commands:
      - npm install -g typescript
  build:
    commands:
      - tsc index.ts
artifacts:
  files:
    - index.js
```

Pour créer un fichier index.ts

- Créez un fichier nommé `index.ts` avec les contenus suivants.

```
interface Greeting {
    message: string;
}

class HelloGreeting implements Greeting {
    message = "Hello!";
}

function greet(greeting: Greeting) {
    console.log(greeting.message);
}

let greeting = new HelloGreeting();

greet(greeting);
```

Pour charger des fichiers sur votre compartiment source S3

1. Vos fichiers doivent être similaires à ce qui suit dans votre répertoire local :

   ```
   buildspec.yml
   index.ts
   ```

   Compressez les fichiers et nommez le fichier `source.zip`.
2. Dans la console Amazon S3, pour votre compartiment source, choisissez Upload. Choisissez Ajouter des fichiers, puis recherchez le fichier ZIP que vous avez créé.
3. Choisissez Charger. Ces fichiers constituent l'artefact source créé par l'assistant Create Pipeline (Création de pipeline) pour votre action de déploiement dans CodePipeline. Votre fichier doit être similaire à ce qui suit dans votre compartiment :

   ```
   source.zip
   ```

Étape 2 : Créer votre pipeline

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Une étape source avec une action Amazon S3 dans laquelle les artefacts source sont les fichiers de votre application téléchargeable.
- Une étape de déploiement avec une action de déploiement Amazon S3.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Sur la page Bienvenue, Démarrez ou Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisir une option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : Choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyS3DeployPipeline`**.
5. Dans Rôle de service, choisissez Nouveau rôle de service pour autoriser CodePipeline la création d'un rôle de service dans IAM.
6. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
7. À l'étape 3 : Ajouter une étape source, dans Source provider, choisissez Amazon S3. Dans Bucket (Compartiment), choisissez le nom de votre compartiment source. Dans S3 object key (Clé d'objet S3), entrez le nom de votre fichier ZIP source. Assurez-vous d'inclure l'extension de fichier .zip.

   Choisissez Suivant.
8. À l'étape 4 : Ajouter une étape de construction :

   a. Dans le champ Fournisseur de génération, choisissez CodeBuild.

   b. Choisissez Créer un projet de génération. Sur la page Create project (Créer un projet) :

   c. Dans Nom du projet, saisissez un nom pour ce projet de génération.

   d. Dans Environnement, choisissez Image gérée. Pour Système d'exploitation, choisissez Ubuntu.

   e. Pour Runtime (Exécution), sélectionnez Standard. Pour la version Runtime, choisissez:1.0aws/codebuild/standard.

   f. Dans Version d'image, choisissez Utilisez Always use the latest image for this runtime version (Toujours utiliser la dernière image pour cette version d'exécution).

g. Pour Rôle de service, choisissez votre rôle CodeBuild de service ou créez-en un.

h. Pour Build specifications (Spécifications de génération), choisissez Use a buildspec file (Utiliser un fichier buildspec).

i. Choisissez Continuer vers CodePipeline. Un message s'affiche si le projet est créé avec succès.

j. Choisissez Suivant.

9. À l'étape 5 : Ajouter une phase de déploiement :

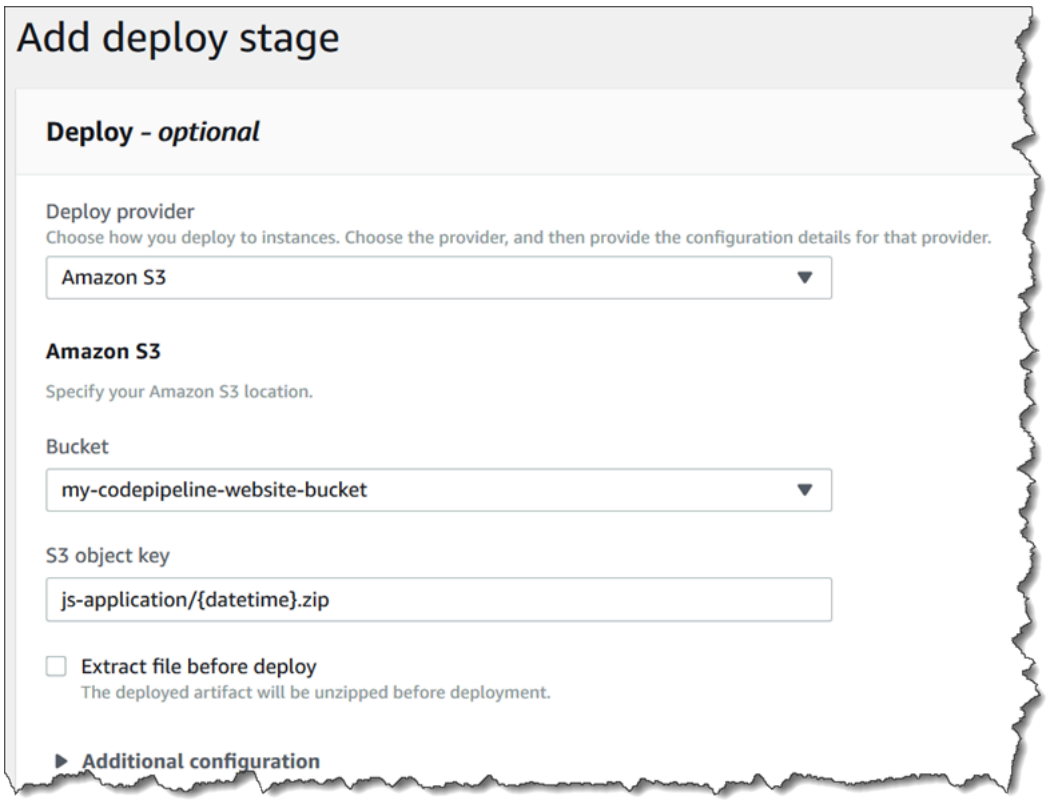
a. Dans Deploy provider (Fournisseur de déploiement), choisissez Amazon S3.

b. Dans Bucket (Compartiment), entrez le nom de votre compartiment cible S3.

c. Assurez-vous que Extract file before deploy (Extraire le fichier avant de déployer) est décochée.

Lorsque Extract file before deploy (Extraire le fichier avant de déployer) est décochée, S3 object key (Clé d'objet S3) est affiché. Entrez le nom du chemin que vous souhaitez utiliser : `js-application/{datetime}.zip`.

Cela crée un `js-application` dossier dans Amazon S3 dans lequel les fichiers sont extraits. Dans ce dossier, la variable `{datetime}` crée un horodatage sur chaque fichier de sortie lorsque votre pipeline s'exécute.



d. (Facultatif) Dans Canned ACL (Liste ACL prête à l'emploi), vous pouvez appliquer un ensemble d'autorisations prédéfinies, appelées liste ACL prête à l'emploi, aux artefacts chargés.

e. (Facultatif) Dans Cache control (Contrôle de cache), entrez les paramètres de mise en cache. Vous pouvez définir cela pour contrôler le comportement de mise en cache pour les demandes/réponses. Pour connaître les valeurs valides, consultez le champ d'en-tête `Cache-Control` pour les opérations HTTP.

f. Choisissez Suivant.

10. Dans Step 6: Review, vérifiez les informations puis choisissez Create pipeline.

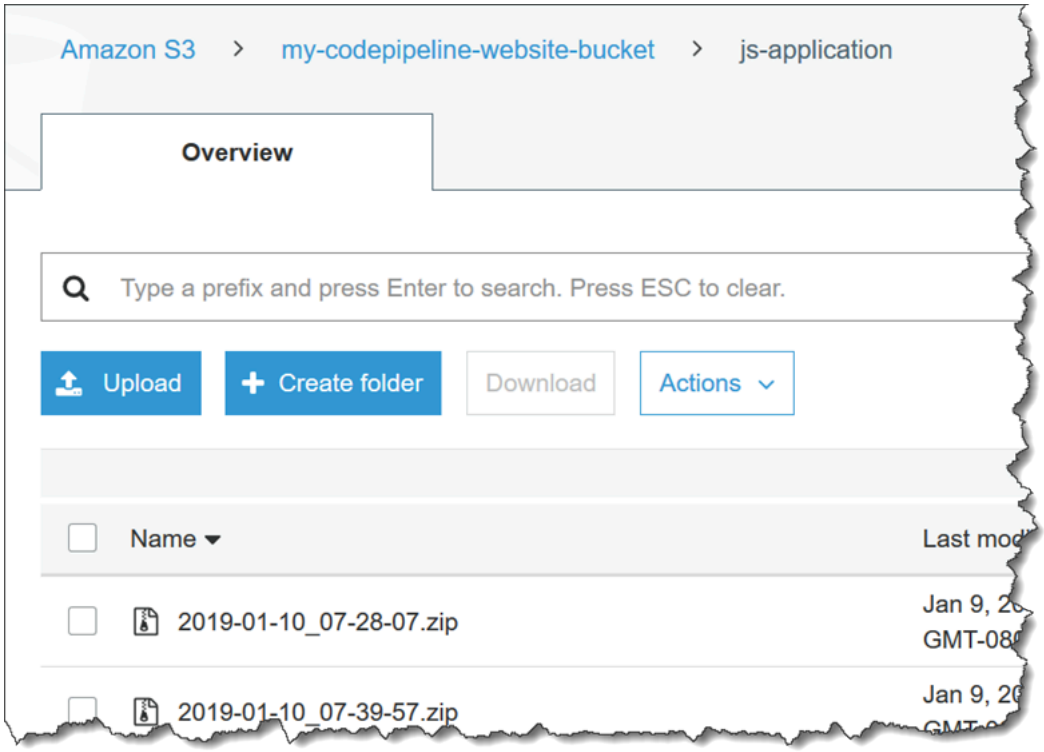
11. Une fois votre pipeline exécuté avec succès, consultez votre compartiment dans la console Amazon S3. Vérifiez que votre fichier ZIP déployé s'affiche dans votre compartiment cible sous le dossier `js-application`. Le JavaScript fichier contenu dans le fichier ZIP doit êtreindex.js. Le fichier `index.js` contient la sortie suivante :

```
var HelloGreeting = /** @class */ (function () {
    function HelloGreeting() {
        this.message = "Hello!";
    }
    return HelloGreeting;
```

```
}());
function greet(greeting) {
    console.log(greeting.message);
}
var greeting = new HelloGreeting();
greet(greeting);
```

Étape 3 : Modifier un fichier source et vérifier le déploiement

Apportez une modification à vos fichiers source, puis chargez-les dans votre compartiment source. Cela déclenche l'exécution de votre pipeline. Affichez votre compartiment cible et vérifiez que les fichiers de sortie déployés sont disponibles dans le dossier `js-application`, comme indiqué ci-après :



Tutoriel : Créez un pipeline qui publie votre application sans serveur sur AWS Serverless Application Repository

Vous pouvez l'utiliser AWS CodePipeline pour fournir en continu votre application AWS SAM sans serveur au AWS Serverless Application Repository.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé CodePipeline pour les artefacts. (Ceci est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient à cette personne sûre et Comptes AWS qu'elle sera fiable.

Ce didacticiel explique comment créer et configurer un pipeline pour créer votre application sans serveur hébergée GitHub et publiée AWS Serverless Application Repository automatiquement sur le. Le pipeline est utilisé GitHub comme fournisseur de source et CodeBuild comme fournisseur de build. Pour publier votre application sans serveur sur le AWS Serverless Application Repository, vous déployez une application (à partir du AWS Serverless Application Repository) et associez la fonction Lambda créée par cette application en tant que fournisseur d'actions Invoke dans votre pipeline. Vous pouvez ensuite fournir des mises à jour d'applications en continu sur le AWS Serverless Application Repository, sans écrire de code.

> **Important**
>
> La plupart des actions que vous ajoutez à votre pipeline dans le cadre de cette procédure impliquent AWS des ressources que vous devez créer avant de créer le pipeline. AWS les ressources pour vos actions source doivent toujours être créées dans la même AWS région que celle où vous créez votre pipeline. Par exemple, si vous créez votre pipeline dans la région USA Est (Ohio), votre CodeCommit référentiel doit se trouver dans la région USA Est (Ohio).
>
> Vous pouvez ajouter des actions interrégionales lorsque vous créez votre pipeline. AWS les ressources pour les actions interrégionales doivent se trouver dans la même AWS région que celle où vous prévoyez d'exécuter l'action. Pour de plus amples informations, veuillez consulter Ajouter une action interrégionale dans CodePipeline.

Avant de commencer

Dans ce didacticiel, nous supposons ce qui suit :

- Vous connaissez AWS Serverless Application Model (AWS SAM) et AWS Serverless Application Repository.

- Vous avez une application sans serveur hébergée dans GitHub laquelle vous avez publié à l' AWS Serverless Application Repository aide de la AWS SAM CLI. Pour publier un exemple d'application sur le AWS Serverless Application Repository, voir Démarrage rapide : publication d'applications dans le guide du AWS Serverless Application Repository développeur. Pour publier votre propre application sur le AWS Serverless Application Repository, consultez la section Publication d'applications à l'aide de la AWS SAM CLI dans le guide du AWS Serverless Application Model développeur.

Étape 1 : Créer un fichier buildspec.yml

Créez un `buildspec.yml` fichier avec le contenu suivant et ajoutez-le au GitHub référentiel de votre application sans serveur. *`template.yml`* Remplacez-le par le AWS SAM modèle de votre application et *`bucketname`* par le compartiment S3 dans lequel votre application packagée est stockée.

```
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
```

Étape 2 : Créer et configurer votre pipeline

Suivez ces étapes pour créer votre pipeline à l' Région AWS endroit où vous souhaitez publier votre application sans serveur.

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Si nécessaire, passez à l' Région AWS endroit où vous souhaitez publier votre application sans serveur.
3. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.

4. Sur la page Étape 1 : Choisir une option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
5. Choisissez Créer un pipeline. Sur la page Étape 2 : Choisir les paramètres du pipeline, dans Nom du pipeline, entrez le nom de votre pipeline.
6. Dans Type de pipeline, choisissez V2. Pour de plus amples informations, veuillez consulter Types de canalisations. Choisissez Suivant.
7. Dans Rôle de service, choisissez Nouveau rôle de service pour autoriser CodePipeline la création d'un rôle de service dans IAM.
8. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
9. Sur la page Étape 3 : Ajouter une étape source, dans Source provider, sélectionnez GitHub.
10. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.
11. Dans Repository, choisissez votre référentiel GitHub source.
12. Dans Branche, choisissez votre GitHub succursale.
13. Conservez les valeurs par défaut restantes pour l'action source. Choisissez Suivant.
14. Sur la page Étape 4 : Ajouter une phase de construction, ajoutez une étape de construction :

    a. Dans le champ Fournisseur de génération, choisissez AWS CodeBuild. Pour Région, utilisez la région du pipeline.

    b. Sélectionnez Create a project (Créer un projet).

    c. Dans Nom du projet, saisissez un nom pour ce projet de génération.

    d. Dans le champ Image d'environnement, choisissez Image gérée. Pour Système d'exploitation, choisissez Ubuntu.

    e. Pour Exécution et Version d'exécution, choisissez l'exécution et la version d'exécution requises pour votre application sans serveur.

    f. Pour Rôle de service, choisissez Nouveau rôle de service.

    g. Pour Build specifications (Spécifications de génération), choisissez Use a buildspec file (Utiliser un fichier buildspec).

    h. Choisissez Continuer vers CodePipeline. Cela ouvre la CodePipeline console et crée un CodeBuild projet qui utilise le `buildspec.yml` dans votre référentiel pour la configuration. Le projet de construction utilise un rôle de service pour gérer les Service AWS autorisations. Cette étape peut prendre quelques minutes.

i. Choisissez Suivant.

15. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer l'étape de test, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer.

    Choisissez Suivant.

16. Sur la page Étape 6 : Ajouter une phase de déploiement, choisissez Ignorer l'étape de déploiement, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer. Choisissez Suivant.

17. À l'étape 7 : Révision, choisissez Créer un pipeline. Vous devriez voir un schéma illustrant les étapes.

18. Accordez au rôle de CodeBuild service l'autorisation d'accéder au compartiment S3 dans lequel votre application packagée est stockée.

    a. À l'étape Génération de votre nouveau pipeline, choisissez CodeBuild.

    b. Choisissez l'onglet Détails de la génération.

    c. Dans Environment, choisissez le rôle CodeBuild de service pour ouvrir la console IAM.

    d. Développez la sélection pour `CodeBuildBasePolicy`, puis choisissez Modifier la stratégie.

    e. Choisissez JSON.

    f. Ajoutez un nouvel énoncé de stratégie avec le contenu suivant. L'instruction permet CodeBuild de placer des objets dans le compartiment S3 dans lequel votre application empaquetée est stockée. *`bucketname`*Remplacez-le par le nom de votre compartiment S3.

```
{
    "Effect": "Allow",
    "Resource": [
        "arn:aws:s3:::bucketname/*"
    ],
    "Action": [
        "s3:PutObject"
    ]
}
```

    g. Choisissez Examiner une politique.

    h. Sélectionnez Enregistrer les modifications.

Étape 3 : Déployer l'application de publication

Procédez comme suit pour déployer l'application qui contient la fonction Lambda qui effectue la publication sur. AWS Serverless Application Repository Cette application s'appelle aws-serverless-codepipeline-serverlessrepo-publish.

> **Note**
>
> Vous devez déployer l'application sur le même site Région AWS que votre pipeline.

1. Accédez à la page de l'application et choisissez Déployer.
2. Sélectionnez Je comprends que cette application crée des rôles IAM personnalisés.
3. Choisissez Déployer.
4. Choisissez View CloudFormation Stack pour ouvrir la CloudFormation console.
5. Développez la section Ressources. Vous voyez ServerlessRepoPublish, laquelle est du genre AWS::Lambda::Function. Prenez note de l'ID physique de cette ressource pour l'étape suivante. Utilisez cet ID physique lorsque vous créez la nouvelle action de publication dans CodePipeline.

Étape 4 : Créer l'action de publication

Procédez comme suit pour créer l'action de publication dans votre pipeline.

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Dans le volet de navigation de gauche, choisissez le pipeline à modifier.
3. Choisissez Modifier.
4. Après la dernière étape de votre pipeline, choisissez +Ajouter une étape. Dans Nom de l'étape, saisissez un nom, tel que **`Publish`**, puis choisissez Ajouter une étape.
5. Dans la nouvelle étape, choisissez + Add action group (+ Ajouter un groupe d'actions).
6. Saisissez un nom d'action. Dans Fournisseur d'action, dans Invoquer, choisissez AWS Lambda.
7. Dans Artefacts d'entrée, sélectionnez BuildArtifact.
8. Dans Nom de la fonction, choisissez l'ID physique de la fonction Lambda que vous avez noté à l'étape précédente.
9. Choisissez Enregistrer pour l'action.
10. Choisissez Effectué pour l'étape.

11. Dans le coin supérieur droit, choisissez Enregistrer.
12. Pour vérifier votre pipeline, apportez une modification à votre application dans GitHub. Par exemple, modifiez la description de l'application dans la `Metadata` section de votre fichier AWS SAM modèle. Validez le changement et envoyez-le à votre GitHub agence. Cela déclenche l'exécution de votre pipeline. Une fois le pipeline terminé, vérifiez que votre application a été mise à jour avec vos modifications dans le AWS Serverless Application Repository.

Tutoriel : Déploiements de fonctions Lambda avec CodePipeline

Ce didacticiel vous aide à créer une action de déploiement CodePipeline qui déploie votre code sur la fonction que vous avez configurée dans Lambda. Vous allez créer un exemple de fonction Lambda dans lequel vous allez créer un alias et une version, ajouter la fonction Lambda compressée à votre emplacement source et exécuter l'action Lambda dans votre pipeline.

> **Note**
>
> Dans le cadre de la création d'un pipeline dans la console, un compartiment d'artefacts S3 sera utilisé par CodePipeline pour les artefacts. (Ceci est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une personne sûre et fiable.

> **Note**
>
> L'action de `Lambda` déploiement n'est disponible que pour les pipelines de type V2.

Conditions préalables

Avant de pouvoir créer votre pipeline de déploiement continu à l'aide de ce didacticiel, vous devez avoir mis en place certaines ressources. Voici ce dont vous avez besoin pour commencer :

> **Note**
>
> Toutes ces ressources devraient être créées au sein de la même AWS région.

- Un référentiel de contrôle de source, tel que GitHub, ou un compartiment source S3 (ce didacticiel utilise S3) dans lequel vous stockerez un `.zip` fichier que vous créez pour votre fonction Lambda.
- Vous devez utiliser un rôle de CodePipeline service existant qui a été mis à jour avec les autorisations nécessaires pour cette action. Pour mettre à jour votre rôle de service, consultezAutorisations relatives à la politique de rôle de service pour l'action de déploiement Lambda.

Dès lors que ces prérequis sont respectés, vous pouvez commencer à créer votre pipeline de déploiement continu à l'aide du didacticiel.

Étape 1 : Création de votre exemple de fonction Lambda

Au cours de cette étape, vous créez la fonction Lambda que vous allez déployer.

Pour créer votre fonction Lambda

1. Accédez à la console Lambda et suivez les étapes du didacticiel suivant pour créer un exemple de fonction Lambda : link.
2. Dans le menu de navigation supérieur, choisissez Créer, puis sélectionnez Commencer à zéro en haut de la page.
3. Pour Name (Nom), entrez **`MyLambdaFunction`**.
4. Publiez une nouvelle version. Il s'agira de la version vers laquelle l'alias pointera.
    a. Sélectionnez votre fonction.
    b. Choisissez le menu déroulant Actions.
    c. Choisissez Publish new version (Publier une nouvelle version).
    d. (Facultatif) Ajoutez à la description dans Description.
    e. Choisissez Publier.
5. Créez un alias pour votre fonction, par exemple`aliasV1`.
6. Assurez-vous que l'alias pointe vers la version que vous venez de créer (1, par exemple).

> **Note**
>
> Si vous choisissez $LATEST, vous ne pouvez pas utiliser les fonctionnalités de transfert de trafic car Lambda ne prend pas en charge $LATEST pour un alias pointant vers plusieurs versions.

Étape 2 : Téléchargez le fichier de fonctions dans votre référentiel

Téléchargez la fonction et enregistrez-la sous forme de fichier zip. Téléchargez le fichier compressé dans votre compartiment S3 en suivant les étapes ci-dessous.

Pour ajouter un fichier **`.zip`** à votre référentiel source

1. Ouvrez votre compartiment S3.
2. Choisissez Charger.
3. Téléchargez le fichier zip contenant votre `sample_lambda_source.zip` fichier dans votre compartiment source.

   Notez le chemin.

   ```
   object key
   ```

Étape 3 : Création de votre pipeline

Utilisez l' CodePipeline assistant pour créer les étapes de votre pipeline et connecter votre référentiel source.

Pour créer le pipeline

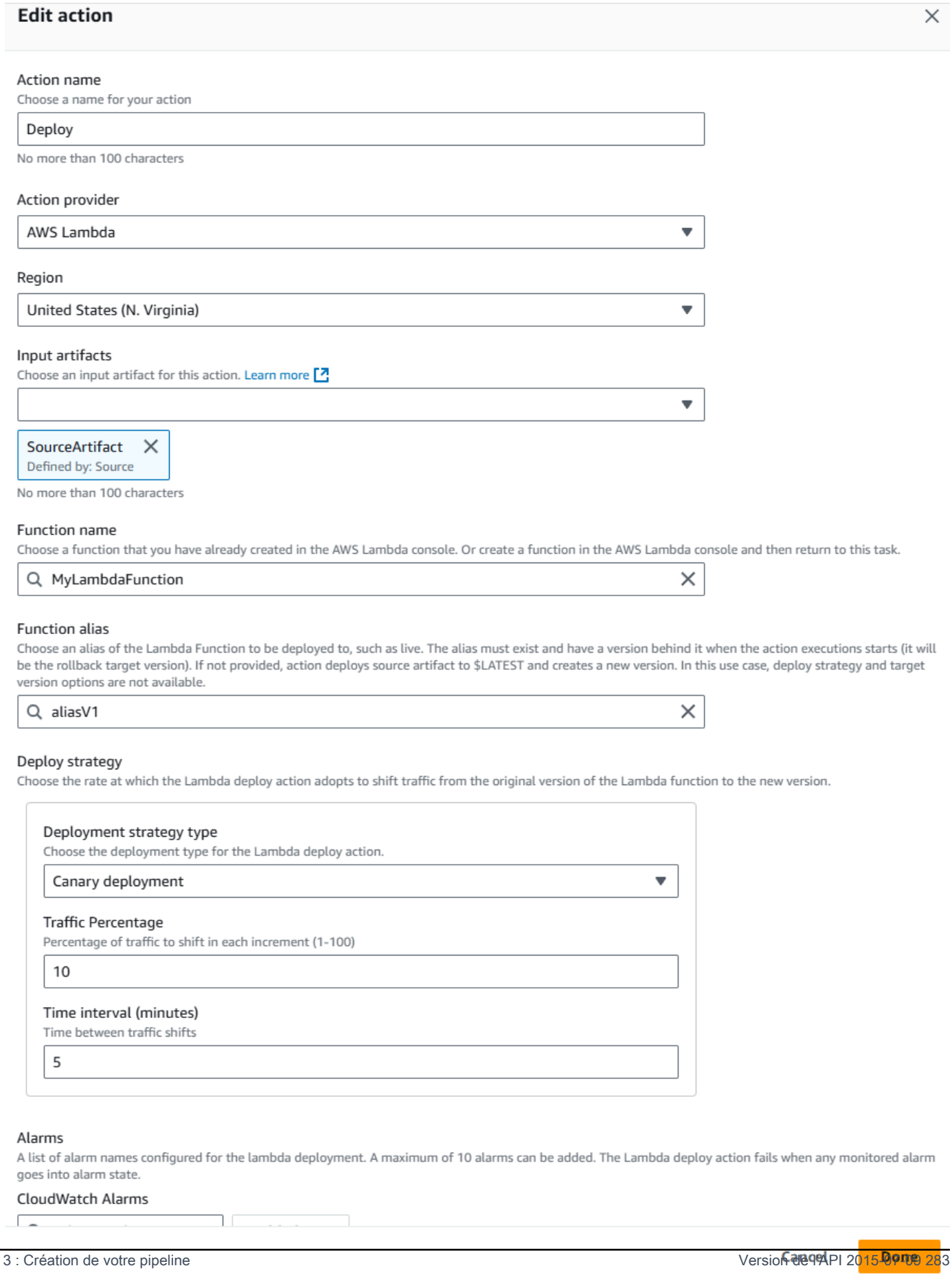
1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisir une option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : Choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyPipeline`**.

5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul type que vous pouvez choisir dans la console. Pour plus d'informations, consultez la section Types de pipelines. Pour plus d'informations sur la tarification CodePipeline, consultez la section Tarification.
6. Dans Rôle de service, choisissez Utiliser le rôle de service existant, puis choisissez le rôle de CodePipeline service qui a été mis à jour avec les autorisations requises pour cette action. Pour configurer votre rôle CodePipeline de service pour cette action, consultezAutorisations relatives à la politique de rôle de service pour l'action de déploiement Lambda.
7. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
8. Sur la page Étape 3 : Ajouter un stage source, ajoutez un stage source :

   a. Dans Fournisseur de source, choisissez Amazon S3.

   b. Dans Clé d'objet, ajoutez le nom de votre fichier .zip, y compris son extension, telle que`sample_lambda_source.zip`.

   Choisissez Suivant.
9. Sur la page Étape 4 : Ajouter une phase de construction, choisissez Ignorer.
10. Sur la page Étape 5 : Ajouter une phase de test, choisissez Ignorer.
11. Sur la page Étape 6 : Ajouter une étape de déploiement, sélectionnez Lambda.

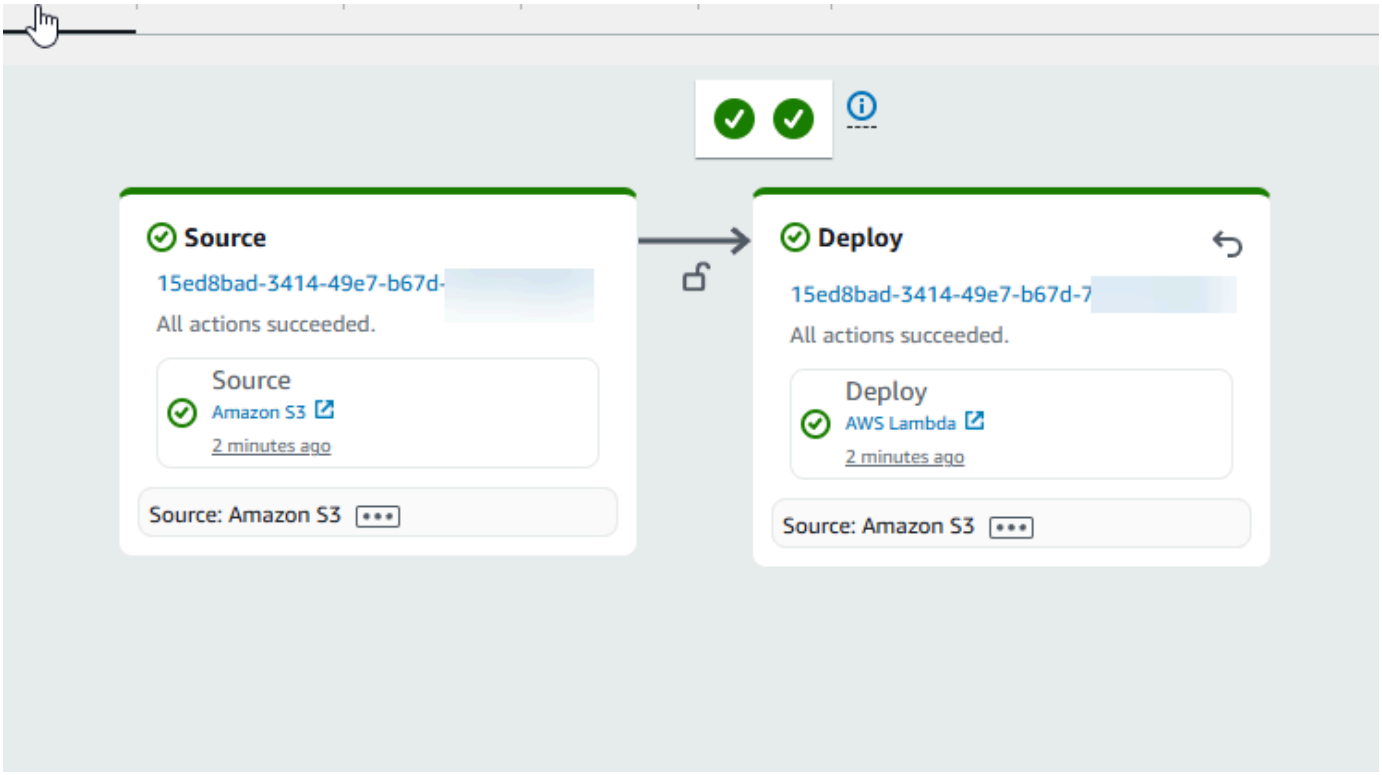

Edit action
Action name
Choose a name for your action
Deploy
No more than 100 characters
Action provider
AWS Lambda
Region
United States (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
Defined by: Source
No more than 100 characters
Function name
Choose 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. Ajoutez le nom et l'alias de votre fonction.

b. Choisissez votre stratégie de déploiement.

c. Choisissez Suivant.

12. Sur la page Étape 7 : Révision, passez en revue la configuration de votre pipeline et choisissez Create pipeline pour créer le pipeline.



Étape 4 : Test de votre pipeline

Votre pipeline doit disposer de tous les éléments nécessaires pour exécuter un déploiement AWS continu end-to-end natif. À présent, testez sa fonctionnalité en transmettant une modification de code à votre référentiel source.

Pour tester votre pipeline

1. Apportez une modification de code au référentiel source que vous avez configuré, validez-la, puis transmettez-la.
2. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
3. Choisissez votre pipeline dans la liste.

4. Observez la progression du pipeline dans ses différentes phases. Votre pipeline doit être terminé et votre action doit être déployée sur votre fonction Lambda.

En savoir plus

L'action de déploiement Lambda permet deux méthodes de déploiement. L'une des méthodes consiste à déplacer le trafic uniquement sans artéfact d'entrée provenant de l'action source. L'autre méthode consiste à mettre à jour le code de fonction à l'aide d'un artefact d'entrée provenant de l'action source, puis à publier une nouvelle version basée sur le code mis à jour. Pour la deuxième méthode, si l'alias est fourni, le trafic CodePipeline sera également transféré. Ce didacticiel d'action de déploiement Lambda explique comment mettre à jour votre fonction à l'aide d'un artefact source.

Pour en savoir plus sur l'action, consultez la page de référence de l'action à l'adresseAWS Lambda référence des actions de déploiement.

Tutoriel : Utilisation de variables avec des actions d'invocation Lambda

Une action d'invocation Lambda peut utiliser des variables d'une autre action dans le cadre de son entrée et renvoyer de nouvelles variables avec sa sortie. Pour plus d'informations sur les variables pour les actions dans CodePipeline, consultezRéférence des variables.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

À la fin de ce tutoriel, vous aurez :

- Une action d'invocation Lambda qui :
  - Consomme la `CommitId` variable à partir d'une action CodeCommit source
  - Émet trois nouvelles variables : `dateTime`, `testRunId` et `region`

- Action d'approbation manuelle qui utilise les nouvelles variables de votre action d'appel Lambda pour fournir une URL de test et un ID d'exécution du test
- Pipeline mis à jour avec les nouvelles actions

Rubriques

- Conditions préalables
- Étape 1 : créer une fonction Lambda
- Étape 2 : ajoutez une action d'appel Lambda et une action d'approbation manuelle à votre pipeline

Conditions préalables

Avant de commencer, les prérequis suivants doivent être remplis :

- Vous pouvez créer ou utiliser le pipeline avec la CodeCommit source dansTutoriel : Création d'un pipeline simple (CodeCommit référentiel).
- Modifiez votre pipeline existant afin que l'action CodeCommit source possède un espace de noms. Affectez l'espace de noms `SourceVariables` à l'action.

Étape 1 : créer une fonction Lambda

Procédez comme suit pour créer une fonction Lambda et un rôle d'exécution Lambda. Vous ajoutez l'action Lambda à votre pipeline après avoir créé la fonction Lambda.

Pour créer une fonction Lambda et un rôle d'exécution

1. Connectez-vous à Console de gestion AWS et ouvrez la AWS Lambda console à l'adresse https://console.aws.amazon.com/lambda/.
2. Choisissez Créer une fonction. Laissez l'option Créer à partir de zéro sélectionnée.
3. Dans Nom de la fonction, entrez le nom de votre fonction, par exemple **`myInvokeFunction`**. Dans Runtime, laissez l'option par défaut sélectionnée.
4. Développez Choose or create an execution role (Choisir ou créer un rôle d'exécution). Choisissez Create a new role with basic Lambda permissions (Créer un rôle avec les autorisations Lambda standard).
5. Choisissez Créer une fonction.

6. Pour utiliser une variable d'une autre action, elle devra être passée à `UserParameters` dans la configuration de l'action d'appel (Invoke) Lambda. Vous allez configurer l'action dans votre pipeline plus tard dans le didacticiel, mais vous ajouterez le code en supposant que la variable sera transmise.

   Pour créer de nouvelles variables, définissez une propriété appelée `outputVariables` sur l'entrée avec la valeur `putJobSuccessResult`. Notez que vous ne pouvez pas produire de variables dans le cadre d'un fichier `putJobFailureResult`.

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

   Dans votre nouvelle fonction, dans l'onglet Code, collez l'exemple de code suivant ci-dessous`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. Autorisez la fonction à enregistrer automatiquement.
8. Copiez le nom de ressource Amazon (ARN) contenu dans le champ ARN de la fonction en haut de l'écran.
9. Enfin, ouvrez la console Gestion des identités et des accès AWS (IAM) à https://console.aws.amazon.com/iam/ l'adresse. Modifiez le rôle d'exécution Lambda pour ajouter la politique suivante : AWSCodePipelineCustomActionAccess. Pour connaître les étapes à suivre pour créer un rôle d'exécution Lambda ou modifier la stratégie de rôle, veuillez consulter Étape 2 : Création de la fonction Lambda.

Étape 2 : ajoutez une action d'appel Lambda et une action d'approbation manuelle à votre pipeline

Au cours de cette étape, vous ajoutez une action d'appel Lambda à votre pipeline. Vous ajoutez l'action dans le cadre d'une étape nommée Test. Le type d'action est une action d'appel. Vous ajoutez ensuite une action d'approbation manuelle après l'action d'appel.

Pour ajouter une action Lambda et une action d'approbation manuelle au pipeline

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés. Choisissez le pipeline dans lequel vous souhaitez ajouter l'action.
2. Ajoutez l'action de test Lambda à votre pipeline.

   a. Pour modifier votre pipeline, choisissez Modifier. Ajoutez une étape après l'action source dans le pipeline existant. Entrez un nom pour l'étape, par exemple **`Test`**.

   b. Dans la nouvelle étape, choisissez Ajouter un groupe d'actions pour ajouter une action. Dans Nom de l'action, entrez le nom de l'action d'appel, par exemple **`Test_Commit`**.

   c. Dans Action provider, sélectionnez AWS Lambda.

   d. Dans Artefacts d'entrée, choisissez le nom de l'artefact de sortie de votre action source, tel que `SourceArtifact`.

   e. Dans FunctionName, ajoutez l'ARN de la fonction Lambda que vous avez créée.

f. Dans Variable namespace (Espace de noms de variables), ajoutez le nom de l'espace de noms, par exemple **`TestVariables`**.

g. Dans Artefacts de sortie, ajoutez le nom de l'artefact de sortie, tel que **`LambdaArtifact`**.

h. Sélectionnez Exécuté.

3. Ajoutez l'action d'approbation manuelle à votre pipeline.

a. Votre pipeline étant toujours en mode d'édition, ajoutez une étape après l'action d'appel. Entrez un nom pour l'étape, par exemple **`Approval`**.

b. Dans cette nouvelle étape, choisissez l'icône pour ajouter une action. Dans Nom de l'action, entrez le nom de l'action d'approbation, par exemple **`Change_Approval`**.

c. Dans Fournisseur d'action, choisissez Approbation manuelle.

d. Dans URL pour la révision, élaborez l'URL en ajoutant la syntaxe de variable pour la variable `region` et la variable `CommitId`. Veillez à utiliser les espaces de noms affectés aux actions qui fournissent les variables de sortie.

Dans cet exemple, l'URL dont la syntaxe est variable pour une CodeCommit action possède l'espace de noms `SourceVariables` par défaut. La variable de sortie de région Lambda a l'espace de noms `TestVariables`. L'URL se présente comme suit.

```
https://#{TestVariables.region}.console.aws.amazon.com/codesuite/codecommit/
repositories/MyDemoRepo/commit/#{SourceVariables.CommitId}
```

Dans Commentaires, élaborez le texte du message d'approbation en ajoutant la syntaxe de variable pour la variable `testRunId`. Pour cet exemple, l'URL avec la syntaxe de variable pour la variable de sortie `testRunId` Lambda a l'espace de noms `TestVariables`. Entrez le message suivant.

```
Make sure to review the code before approving this action. Test Run ID:
 #{TestVariables.testRunId}
```

4. Choisissez Effectué pour fermer l'écran d'édition de l'action, puis choisissez Effectué pour fermer l'écran d'édition de l'étape. Pour enregistrer le pipeline, choisissez Effectué. Le pipeline terminé contient désormais une structure avec les étapes source, de test, d'approbation et de déploiement.

Choisissez Publier une modification pour exécuter la dernière modification via la structure du pipeline.

5. Lorsque le pipeline atteint l'étape d'approbation manuelle, choisissez Vérification. Les variables résolues apparaissent comme l'URL de l'ID de validation. Votre approbateur peut choisir cette URL pour afficher la validation.
6. Une fois le pipeline exécuté avec succès, vous pouvez également afficher les valeurs des variables sur la page de l'historique d'exécution des actions.

Tutoriel : Utiliser une AWS Step Functions action d'appel dans un pipeline

Vous pouvez l'utiliser AWS Step Functions pour créer et configurer des machines d'état. Ce didacticiel vous montre comment ajouter une action d'appel à un pipeline qui active les exécutions de machines d'état à partir de votre pipeline.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé CodePipeline pour les artefacts. (Ceci est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient à cette personne sûre et Comptes AWS qu'elle sera fiable.

Dans ce didacticiel, vous effectuez les tâches suivantes :

- Créez une machine à états standard dans AWS Step Functions.
- Entrez directement le JSON d'entrée de la machine d'état. Vous pouvez également télécharger le fichier d'entrée de la machine à états dans un compartiment Amazon Simple Storage Service (Amazon S3).
- Mettez à jour votre pipeline en ajoutant l'action de la machine d'état.

Rubriques

- Condition préalable : créer ou choisir un pipeline simple
- Étape 1 : Créer l'exemple de machine d'état
- Étape 2 : Ajouter une action d'appel Step Functions à votre pipeline

Condition préalable : créer ou choisir un pipeline simple

Dans ce didacticiel, vous ajoutez une action d'appel à un pipeline existant. Vous pouvez utiliser le pipeline que vous avez créé dans Didacticiel : Création d'un pipeline simple (compartiment S3) ou Tutoriel : Création d'un pipeline simple (CodeCommit référentiel).

Vous utilisez un pipeline existant avec une action source et au moins une structure en deux étapes, mais vous n'utilisez pas d'artefacts source pour cet exemple.

> **Note**
>
> Vous devrez peut-être mettre à jour le rôle de service utilisé par votre pipeline avec les autorisations supplémentaires requises pour exécuter cette action. Pour ce faire, ouvrez la console Gestion des identités et des accès AWS (IAM), recherchez le rôle, puis ajoutez les autorisations à la politique du rôle. Pour de plus amples informations, veuillez consulter Ajouter des autorisations au rôle CodePipeline de service.

Étape 1 : Créer l'exemple de machine d'état

Dans la console Step Functions, créez une machine à états à l'aide du modèle `HelloWorld` d'exemple. Pour obtenir des instructions, consultez la section Créer une machine à états dans le guide du AWS Step Functions développeur.

Étape 2 : Ajouter une action d'appel Step Functions à votre pipeline

Ajoutez une action d'appel Step Functions à votre pipeline comme suit :

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés.
2. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier. Une vue détaillée du pipeline s'affiche alors, laquelle indique notamment l'état de chaque action, dans chaque étape du pipeline.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Sur la deuxième étape de votre pipeline simple, choisissez Modifier l'étape. Sélectionnez Delete (Supprimer). Ceci supprime la deuxième étape maintenant que vous n'en avez plus besoin.

5. Au bas du diagramme, choisissez + Ajouter une étape.
6. Dans Nom d'étape, saisissez un nom pour l'étape, par exemple **`Invoke`**, puis choisissez Ajouter une étape.
7. Choisissez + Ajouter un groupe d'actions.
8. Dans Nom de l'action, saisissez un nom, par exemple **`Invoke`**.
9. Dans Action provider, sélectionnez AWS Step Functions. Acceptez la région du pipeline comme Région par défaut.
10. Dans Artefacts d'entrée, choisissez `SourceArtifact`.
11. Dans ARN de la machine d'état, choisissez l'ARN (Amazon Resource Name) de la machine d'état que vous avez créée précédemment.
12. (Facultatif) Dans Préfixe du nom d'exécution, saisissez un préfixe à ajouter à l'ID d'exécution de la machine d'état.
13. Dans Type d'entrée, choisissez Littéral.
14. Dans Entrée, saisissez le JSON d'entrée que l'exemple de machine d'état `HelloWorld` attend.

    > **Note**
    >
    > L'entrée dans l'exécution de la machine à états est différente du terme utilisé CodePipeline pour décrire les artefacts d'entrée pour les actions.

    Pour cet exemple, saisissez le JSON suivant :

    ```
    {"IsHelloWorldExample": true}
    ```

15. Sélectionnez Exécuté.
16. Sur l'étape que vous modifiez, choisissez Terminé. Dans le volet AWS CodePipeline , choisissez Enregistrer puis Enregistrer dans le message d'avertissement.
17. Pour soumettre vos modifications et lancer l'exécution d'un pipeline, choisissez Changement de version, puis Publication.
18. Une fois votre pipeline terminé, choisissez AWS Step Functions dans votre action d'invocation. Dans la AWS Step Functions console, consultez l'ID d'exécution de votre machine à états. L'ID affiche le nom de votre machine d'état `HelloWorld` et l'ID d'exécution de la machine d'état avec le préfixe `my-prefix`.

```
arn:aws:states:us-west-2:account-ID:execution:HelloWorld:my-
prefix-0d9a0900-3609-4ebc-925e-83d9618fcca1
```

Tutoriel : Création d'un pipeline utilisé AWS AppConfig comme fournisseur de déploiement

Dans ce didacticiel, vous allez configurer un pipeline qui fournit en continu des fichiers de configuration en AWS AppConfig tant que fournisseur d'actions de déploiement dans votre phase de déploiement.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé CodePipeline pour les artefacts. (Ceci est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient à cette personne sûre et Comptes AWS qu'elle sera fiable.

Rubriques

- Conditions préalables
- Étape 1 : Créez vos AWS AppConfig ressources
- Étape 2 : télécharger des fichiers dans votre compartiment source S3
- Étape 3 : Créer votre pipeline
- Étape 4 : apporter une modification à n'importe quel fichier source et vérifier le déploiement

Conditions préalables

Avant de commencer, vous devez effectuer les opérations suivantes :

- Cet exemple utilise une source S3 pour votre pipeline. Créez ou utilisez un compartiment Amazon S3 avec le contrôle de version activé. Suivez les instructions fournies dans Étape 1 : créer un compartiment source S3 pour votre application pour créer un compartiment S3.

Étape 1 : Créez vos AWS AppConfig ressources

Dans cette section, vous allez créer les ressources suivantes :

- Une application in AWS AppConfig est une unité logique de code qui fournit des fonctionnalités à vos clients.
- Un environnement dans AWS AppConfig est un groupe de déploiement logique de AppConfig cibles, telles que des applications dans un environnement bêta ou de production.
- Un profil de configuration est un ensemble de paramètres qui influencent le comportement de votre application. Le profil de configuration permet AWS AppConfig d'accéder à votre configuration dans son emplacement enregistré.
- (Facultatif) Une stratégie de déploiement AWS AppConfig définit le comportement d'un déploiement de configuration, par exemple le pourcentage de clients devant recevoir la nouvelle configuration déployée à un moment donné au cours d'un déploiement.

Pour créer une application, un environnement, un profil de configuration et une stratégie de déploiement

1. Connectez-vous au Console de gestion AWS.
2. Suivez les étapes décrites dans les rubriques suivantes pour créer vos ressources dans AWS AppConfig.
   - Créez une application.
   - Créez un environnement.
   - Créez un profil AWS CodePipeline de configuration.
   - (Facultatif) Choisissez une stratégie de déploiement prédéfinie ou créez la vôtre.

Étape 2 : télécharger des fichiers dans votre compartiment source S3

Dans cette section, créez votre ou vos fichiers de configuration. Ensuite, compressez et transférez vos fichiers source vers le compartiment que le pipeline utilise pour votre étape source.

Pour créer des fichiers de configuration

1. Créez un `configuration.json` fichier pour chaque configuration dans chaque région. Incluez le contenu suivant :

```
Hello World!
```

2. Suivez les étapes ci-dessous pour compresser et télécharger vos fichiers de configuration.

Pour compresser et télécharger des fichiers source

1. Créez un fichier .zip avec vos fichiers et nommez-le. `configuration-files.zip` Par exemple, votre fichier .zip peut utiliser la structure suivante :

```
.
### appconfig-configurations
    ### MyConfigurations
        ### us-east-1
        #   ### configuration.json
        ### us-west-2
            ### configuration.json
```

2. Dans la console Amazon S3 de votre compartiment, choisissez Upload et suivez les instructions pour télécharger votre fichier .zip.

Étape 3 : Créer votre pipeline

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Un stage source avec une action Amazon S3 où les artefacts source sont les fichiers de votre configuration.
- Une phase de déploiement avec une action AppConfig de déploiement.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Sur la page Bienvenue, la page Démarrez ou la page Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisir une option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : Choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyAppConfigPipeline`**.

5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul type que vous pouvez choisir dans la console. Pour plus d'informations, consultez la section Types de pipelines. Pour plus d'informations sur la tarification CodePipeline, consultez la section Tarification.
6. Dans Rôle de service, choisissez Nouveau rôle de service pour autoriser CodePipeline la création d'un rôle de service dans IAM.
7. Laissez les paramètres sous Advanced settings (Paramètres avancés) à leurs valeurs par défaut, puis choisissez Suivant.
8. À l'étape 3 : Ajouter une étape source, dans Source provider, choisissez Amazon S3. Dans Bucket, choisissez le nom de votre compartiment source S3.

   Dans la clé d'objet S3, entrez le nom de votre fichier .zip :`configuration-files.zip`.

   Choisissez Suivant.
9. À l'étape 4 : Ajouter une phase de construction, choisissez Ignorer l'étape de génération, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer.

   Choisissez Suivant.
10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer l'étape de test, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer.

    Choisissez Suivant.
11. À l'étape 6 : Ajouter une phase de déploiement :

    a. Dans Deploy provider, sélectionnez AWS AppConfig.

    b. Dans Application, choisissez le nom de l'application dans laquelle vous l'avez créée AWS AppConfig. Le champ indique l'ID de votre candidature.

    c. Dans Environnement, choisissez le nom de l'environnement dans lequel vous l'avez créé AWS AppConfig. Le champ indique l'identifiant de votre environnement.

    d. Dans Profil de configuration, choisissez le nom du profil de configuration que vous avez créé dans AWS AppConfig. Le champ indique l'ID de votre profil de configuration.

    e. Dans Stratégie de déploiement, choisissez le nom de votre stratégie de déploiement. Il peut s'agir d'une stratégie de déploiement que vous avez créée dans AppConfig ou d'une stratégie que vous avez choisie parmi les stratégies de déploiement prédéfinies dans AppConfig. Le champ indique l'identifiant de votre stratégie de déploiement.

f. Dans Chemin de configuration de l'artefact en entrée, entrez le chemin du fichier. Assurez-vous que le chemin de configuration de votre artefact d'entrée correspond à la structure de répertoire du fichier .zip de votre compartiment S3. Pour cet exemple, entrez le chemin de fichier suivant :`appconfig-configurations/MyConfigurations/us-west-2/configuration.json`.

g. Choisissez Suivant.

12. À l'étape 7 : Passez en revue, passez en revue les informations, puis choisissez Créer un pipeline.

Étape 4 : apporter une modification à n'importe quel fichier source et vérifier le déploiement

Apportez une modification à vos fichiers source et chargez-la dans votre bucket. Cela déclenche l'exécution de votre pipeline. Vérifiez que votre configuration est disponible en consultant la version.

Tutoriel : Utiliser un clone complet avec une source de GitHub pipeline

Vous pouvez choisir l'option de clonage complet pour votre action GitHub source dans CodePipeline. Utilisez cette option pour exécuter des CodeBuild commandes pour les métadonnées Git dans votre action de création de pipeline.

> **Note**
>
> L'option de clonage complet décrite ici consiste à spécifier si les métadonnées du référentiel CodePipeline doivent être clonées, qui ne peuvent être utilisées que par CodeBuild des commandes. Pour utiliser un jeton GitHub d'accès utilisateur à utiliser avec CodeBuild des projets, suivez les étapes décrites ici pour installer le AWS connecteur pour GitHub l'application, puis laissez le champ d'installation de l'application vide. AWS CodeConnections utilisera le jeton d'accès utilisateur pour la connexion.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé CodePipeline pour les artefacts. (Ceci est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient à cette personne sûre et Comptes AWS qu'elle sera fiable.

Dans ce didacticiel, vous allez créer un pipeline qui se connecte à votre GitHub dépôt, utilise l'option de clonage complet pour les données source et exécute une CodeBuild version qui clone votre dépôt et exécute des commandes Git pour le référentiel.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans les régions Asie-Pacifique (Hong Kong), Afrique (Le Cap), Moyen-Orient (Bahreïn), Europe (Zurich) AWS GovCloud ou US-West (). Pour faire référence aux autres actions disponibles, voirIntégrations de produits et de services avec CodePipeline. Pour les considérations relatives à cette action dans la région Europe (Milan), voir la note dansCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

Rubriques

- Conditions préalables
- Étape 1 : Création d'un fichier README
- Étape 2 : Créez votre pipeline et créez votre projet
- Étape 3 : mettre à jour la politique des rôles de CodeBuild service pour utiliser les connexions
- Étape 4 : Afficher les commandes du référentiel dans la sortie de compilation

Conditions préalables

Avant de commencer, vous devez exécuter les opérations suivantes :

- Créez un GitHub référentiel avec votre GitHub compte.

- Préparez vos GitHub informations d'identification. Lorsque vous utilisez le Console de gestion AWS pour établir une connexion, il vous est demandé de vous connecter avec vos GitHub informations d'identification.

Étape 1 : Création d'un fichier README

Après avoir créé votre GitHub dépôt, procédez comme suit pour ajouter un fichier README.

1. Connectez-vous à votre GitHub dépôt et choisissez votre dépôt.
2. Pour créer un nouveau fichier, choisissez Ajouter un fichier > Créer un nouveau fichier. Nommez le fichier `README.md` et ajoutez le texte suivant.

   ```
   This is a GitHub repository!
   ```

3. Choisissez Valider les modifications.

   Assurez-vous que le fichier `README.md` est au niveau racine de votre référentiel.

Étape 2 : Créez votre pipeline et créez votre projet

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Un stage source avec une connexion à votre GitHub référentiel et une action.
- Une phase de construction avec une action de AWS CodeBuild construction.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Bienvenue, Démarrez ou Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisir une option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : Choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyGitHubPipeline`**.

5. Dans Type de pipeline, choisissez V1 pour les besoins de ce didacticiel. Vous pouvez également choisir la V2 ; toutefois, notez que les types de pipelines diffèrent en termes de caractéristiques et de prix. Pour plus d'informations, consultez Types de canalisations.
6. Dans Service role (Rôle de service), choisissez New service role (Nouveau rôle de service).

   > **Note**
   >
   > Si vous choisissez plutôt d'utiliser votre rôle de CodePipeline service existant, assurez-vous d'avoir ajouté l'autorisation `codestar-connections:UseConnection` IAM à votre politique de rôle de service. Pour obtenir des instructions relatives au rôle de CodePipeline service, voir Ajouter des autorisations au rôle CodePipeline de service.

7. Sous Paramètres avancés, conservez les valeurs par défaut. Dans le magasin d'artefacts choisissez Default location (Emplacement par défaut) pour utiliser le magasin d'artefacts par défaut, tel que le compartiment d'artefacts Amazon S3 désigné par défaut, pour votre pipeline dans la région que vous avez sélectionnée pour ce dernier.

   > **Note**
   >
   > Il ne s'agit pas du compartiment source de votre code source. Il s'agit du magasin d'artefacts pour votre pipeline. Un magasin d'artefacts distinct, tel qu'un compartiment S3, est nécessaire pour chaque pipeline.

   Choisissez Suivant.
8. Sur la page Étape 3 : Ajouter un stage source, ajoutez un stage source :

   a. Dans Source provider, sélectionnez GitHub (via GitHub App).

   b. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.

      Installez une application pour toutes vos connexions à un fournisseur particulier. Si vous avez déjà installé le AWS Connector for GitHub app, choisissez-le et ignorez cette étape.

      > **Note**
      >
      > Si vous souhaitez créer un jeton d'accès utilisateur, assurez-vous d'avoir déjà installé le AWS connecteur pour l' GitHub application, puis laissez le champ

Installation de l'application vide. AWS CodeConnections utilisera le jeton d'accès utilisateur pour la connexion. Pour plus d'informations, voir Accéder à votre fournisseur de source dans CodeBuild.

c. Dans Nom du référentiel, choisissez le nom de votre référentiel GitHub.

d. Dans Nom de la branche, choisissez la branche du référentiel que vous souhaitez utiliser.

e. Veillez à ce que l'option Démarrer le pipeline lors de la modification du code source soit sélectionnée.

f. Sous Format d'artefact de sortie, choisissez Full clone pour activer l'option Git clone pour le référentiel source. Seules les actions fournies par CodeBuild peuvent utiliser l'option Git clone. Étape 3 : mettre à jour la politique des rôles de CodeBuild service pour utiliser les connexionsDans ce didacticiel, vous utiliserez cette option pour mettre à jour les autorisations associées à votre rôle de service de CodeBuild projet.

Choisissez Suivant.

9. Dans Étape 4 : Ajouter une phase de construction, ajoutez une étape de construction :

a. Dans le champ Fournisseur de génération, choisissez AWS CodeBuild. Acceptez la région du pipeline comme Région par défaut.

b. Sélectionnez Create a project (Créer un projet).

c. Dans Nom du projet, saisissez un nom pour ce projet de génération.

d. Dans le champ Image d'environnement, choisissez Image gérée. Pour Système d'exploitation, choisissez Ubuntu.

e. Pour Runtime (Exécution), sélectionnez Standard. Pour Image, choisissez aws/codebuild/standard:5.0.

f. Pour Rôle de service, choisissez Nouveau rôle de service.

> **Note**
>
> Notez le nom de votre rôle CodeBuild de service. Vous aurez besoin du nom du rôle pour la dernière étape de ce didacticiel.

g. Sous Buildspec, pour Build specifications (Spécifications de génération), choisissez Insert build commands (Insérer des commandes de génération). Choisissez Passer à l'éditeur, puis collez le texte suivant sous Commandes de génération.

> **Note**
>
> Dans la env section des spécifications de compilation, assurez-vous que l'assistant d'identification pour les commandes git est activé comme indiqué dans cet exemple.

```
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. Choisissez Continuer vers CodePipeline. Cela revient à la CodePipeline console et crée un CodeBuild projet qui utilise vos commandes de construction pour la configuration. Le projet de construction utilise un rôle de service pour gérer les Service AWS autorisations. Cette étape peut prendre quelques minutes.

i. Choisissez Suivant.

10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer l'étape de test, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer.

    Choisissez Suivant.

11. Sur la page Étape 6 : Ajouter une phase de déploiement, choisissez Ignorer l'étape de déploiement, puis acceptez le message d'avertissement en choisissant à nouveau Ignorer. Choisissez Suivant.

12. À l'étape 7 : Révision, choisissez Créer un pipeline.

Étape 3 : mettre à jour la politique des rôles de CodeBuild service pour utiliser les connexions

L'exécution initiale du pipeline échouera car le rôle de CodeBuild service doit être mis à jour avec les autorisations d'utilisation des connexions. Ajoutez l'autorisation `codestar-connections:UseConnection` IAM à votre politique de rôle de service. Pour obtenir des instructions sur la mise à jour de la politique dans la console IAM, consultezAjoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com.

Étape 4 : Afficher les commandes du référentiel dans la sortie de compilation

1. Lorsque votre rôle de service est correctement mis à jour, choisissez Réessayer en cas d'échec CodeBuild .
2. Une fois le pipeline exécuté avec succès, lors de la phase de construction réussie, choisissez Afficher les détails.

Sur la page de détails, choisissez l'onglet Logs. Affichez le résultat CodeBuild de la compilation. Les commandes produisent la valeur de la variable saisie.

Les commandes affichent le contenu du `README.md` fichier, répertorient les fichiers du répertoire, clonent le référentiel, affichent le journal et archivent le référentiel sous forme de fichier ZIP.

Tutoriel : Utiliser un clone complet avec une source de CodeCommit pipeline

Vous pouvez choisir l'option de clonage complet pour votre action CodeCommit source dans CodePipeline. Utilisez cette option pour autoriser l'accès CodeBuild aux métadonnées Git dans votre action de création de pipeline.

Dans ce didacticiel, vous allez créer un pipeline qui accède à votre CodeCommit dépôt, utilise l'option de clonage complet pour les données sources et exécute une CodeBuild version qui clone votre référentiel et exécute des commandes Git pour le référentiel.

> **Note**
>
> CodeBuild les actions sont les seules actions en aval qui prennent en charge l'utilisation des métadonnées Git disponibles avec l'option Git clone. De plus, bien que votre pipeline puisse contenir des actions multicomptes, l' CodeCommitaction et l' CodeBuild action doivent se trouver dans le même compte pour que l'option de clonage complet réussisse.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Rubriques

- Conditions préalables
- Étape 1 : Création d'un fichier README
- Étape 2 : Créez votre pipeline et créez un projet
- Étape 3 : mettre à jour la politique CodeBuild de rôle de service pour cloner le référentiel
- Étape 4 : Afficher les commandes du référentiel dans la sortie de compilation

Conditions préalables

Avant de commencer, vous devez créer un CodeCommit référentiel dans le même AWS compte et la même région que votre pipeline.

Étape 1 : Création d'un fichier README

Suivez ces étapes pour ajouter un fichier README à votre référentiel source. Le fichier README fournit un exemple de fichier source à lire pour l'action CodeBuild en aval.

Pour ajouter un fichier README

1. Connectez-vous à votre référentiel et choisissez votre référentiel.
2. Pour créer un nouveau fichier, choisissez Ajouter un fichier > Créer un fichier. Donnez un nom au fichier `README.md` et ajoutez le texte suivant.

   ```
   This is a CodeCommit repository!
   ```

3. Choisissez Valider les modifications.

   Assurez-vous que le fichier `README.md` est au niveau racine de votre référentiel.

Étape 2 : Créez votre pipeline et créez un projet

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Une étape source avec une action CodeCommit source.
- Une étape de construction avec une action de AWS CodeBuild construction.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Bienvenue, Démarrez ou Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyCodeCommitPipeline`**.
5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.
6. Dans Rôle du service, sélectionnez l'une des options suivantes :
   - Choisissez Existing service role (Rôle de service existant).
   - Choisissez votre rôle CodePipeline de service actuel. Ce rôle doit disposer de l'autorisation `codecommit:GetRepository` IAM pour votre politique de rôle de service. Consultez la section Ajouter des autorisations au rôle CodePipeline de service.
7. Sous Paramètres avancés, conservez les valeurs par défaut. Choisissez Suivant.
8. Sur la page Étape 3 : Ajouter une étape source, procédez comme suit :

   a. Dans Fournisseur de source, choisissez CodeCommit.

   b. Dans Nom du référentiel, choisissez le nom de votre référentiel.

   c. Dans Nom de la succursale, choisissez le nom de votre succursale.

   d. Veillez à ce que l'option Démarrer le pipeline lors de la modification du code source soit sélectionnée.

   e. Sous Format d'artefact de sortie, choisissez Full clone pour activer l'option Git clone pour le référentiel source. Seules les actions fournies par CodeBuild peuvent utiliser l'option de clonage Git.

   Choisissez Suivant.
9. À l'étape 4 : Ajouter une étape de construction, procédez comme suit :

a. Dans le champ Fournisseur de génération, choisissez AWS CodeBuild. Acceptez la région du pipeline comme Région par défaut.

b. Sélectionnez Create a project (Créer un projet).

c. Dans Nom du projet, saisissez un nom pour ce projet de génération.

d. Dans le champ Image d'environnement, choisissez Image gérée. Pour Système d'exploitation, choisissez Ubuntu.

e. Pour Runtime (Exécution), sélectionnez Standard. Pour Image, choisissez aws/codebuild / standard:5.0.

f. Pour Rôle de service, choisissez Nouveau rôle de service.

> **Note**
>
> Notez le nom de votre rôle CodeBuild de service. Vous aurez besoin du nom du rôle pour la dernière étape de ce didacticiel.

g. Sous Buildspec, pour Build specifications (Spécifications de génération), choisissez Insert build commands (Insérer des commandes de génération). Choisissez Passer à l'éditeur, puis, sous Commandes de génération, collez le code suivant.

```
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. Choisissez Continuer vers CodePipeline. Cela vous ramène à la CodePipeline console et crée un CodeBuild projet qui utilise vos commandes de build pour la configuration. Le projet de génération utilise un rôle de service pour gérer les Service AWS autorisations. Cette étape peut prendre quelques minutes.

i. Choisissez Suivant.

10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.

11. Sur la page Étape 6 : Ajouter une phase de déploiement, choisissez Ignorer l'étape de déploiement, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer. Choisissez Suivant.

12. À l'étape 7 : Révision, choisissez Créer un pipeline.

Étape 3 : mettre à jour la politique CodeBuild de rôle de service pour cloner le référentiel

L'exécution initiale du pipeline échouera car vous devez mettre à jour le rôle de CodeBuild service avec les autorisations nécessaires pour extraire des données de votre référentiel.

Ajoutez l'autorisation `codecommit:GitPull` IAM à votre politique de rôle de service. Pour obtenir des instructions sur la mise à jour de la politique dans la console IAM, consultezAjouter CodeBuild GitClone des autorisations pour les actions CodeCommit source.

Étape 4 : Afficher les commandes du référentiel dans la sortie de compilation

Pour afficher le résultat de la compilation

1. Lorsque votre rôle de service est correctement mis à jour, choisissez Réessayer lors de l' CodeBuild étape qui a échoué.
2. Une fois le pipeline exécuté avec succès, lors de la phase de création réussie, choisissez Afficher les détails.

   Sur la page de détails, choisissez l'onglet Journaux. Afficher le résultat CodeBuild de la compilation. Les commandes génèrent la valeur de la variable saisie.

   Les commandes affichent le contenu du `README.md` fichier, répertorient les fichiers du répertoire, clonent le référentiel, affichent le journal et s'exécutent`git describe --all`.

Tutoriel : Création d'un pipeline avec AWS CloudFormation StackSets actions de déploiement

Dans ce didacticiel, vous allez utiliser la AWS CodePipeline console pour créer un pipeline avec des actions de déploiement permettant de créer un ensemble de piles et de créer des instances de pile. Lorsque le pipeline s'exécute, le modèle crée un ensemble de piles et crée et met à jour les instances dans lesquelles l'ensemble de piles est déployé.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Il existe deux manières de gérer les autorisations pour un ensemble de piles : les rôles IAM autogérés et les rôles AWS IAM gérés. Ce didacticiel fournit des exemples d'autorisations autogérées.

Pour l'utiliser StackSets le plus efficacement possible CodePipeline, vous devez avoir une compréhension claire des concepts sous-jacents CloudFormation StackSets et de leur fonctionnement. Consultez StackSets les concepts dans le guide de AWS CloudFormation l'utilisateur.

Rubriques

- Conditions préalables
- Étape 1 : Téléchargez l'échantillon AWS CloudFormation modèle et fichier de paramètres
- Étape 2 : Créer votre pipeline
- Étape 3 : Afficher le déploiement initial
- Étape 4 : ajouter une CloudFormationStackInstances action
- Étape 5 : Afficher les ressources du stack set pour votre déploiement
- Étape 6 : effectuez une mise à jour de votre stack set

Conditions préalables

Pour les opérations liées au stack set, vous utilisez deux comptes différents : un compte d'administration et un compte cible. Vous créez des ensembles de piles dans le compte administrateur. Vous créez des piles individuelles qui appartiennent à une pile définie dans le compte cible.

Pour créer un rôle d'administrateur avec votre compte administrateur

- Suivez les instructions de la section Configuration des autorisations de base pour les opérations relatives aux ensembles de piles. Votre rôle doit être nommé **`AWSCloudFormationStackSetAdministrationRole`**.

Pour créer un rôle de service dans le compte cible

- Créez un rôle de service dans le compte cible qui approuve le compte administrateur. Suivez les instructions de la section Configuration des autorisations de base

pour les opérations relatives aux ensembles de piles. Votre rôle doit être nommé **`AWSCloudFormationStackSetExecutionRole`**.

Étape 1 : Téléchargez l'échantillon AWS CloudFormation modèle et fichier de paramètres

Créez un compartiment source pour votre modèle d'ensemble de piles et vos fichiers de paramètres. Téléchargez l'exemple de fichier AWS CloudFormation modèle, configurez un fichier de paramètres, puis compressez les fichiers avant de les charger dans votre compartiment source S3.

> **Note**
>
> Assurez-vous de compresser les fichiers sources avant de les charger dans votre compartiment source S3, même si le seul fichier source est le modèle.

Pour créer un compartiment source S3

1. Connectez-vous à Console de gestion AWS et ouvrez la console Amazon S3 à l'adresse https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Pour Bucket name (Nom de compartiment), saisissez un nom pour le compartiment.

   Dans Région, choisissez la région dans laquelle vous souhaitez créer votre pipeline. Choisissez Créer un compartiment.
4. Une fois le compartiment créé, une bannière de réussite apparaît. Choisissez Go to bucket details (Accéder aux détails du compartiment).
5. Dans l'onglet Propriétés, choisissez Versioning. Choisissez Activer la gestion des versions, puis Enregistrer.

Pour créer le AWS CloudFormation fichier modèle

1. Téléchargez le fichier modèle d'exemple suivant pour générer CloudTrail la configuration des ensembles de piles :https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSCloudtrail.yml.

2. Enregistrez le fichier sous le nom `template.yml`.

Pour créer le fichier parameters.txt

1. Créez un fichier contenant les paramètres de votre déploiement. Les paramètres sont des valeurs que vous souhaitez mettre à jour dans votre pile au moment de l'exécution. L'exemple de fichier suivant met à jour les paramètres du modèle pour votre ensemble de piles afin de permettre la validation de la journalisation et les événements globaux.

```
[
  {
    "ParameterKey": "EnableLogFileValidation",
    "ParameterValue": "true"
  },
  {
    "ParameterKey": "IncludeGlobalEvents",
    "ParameterValue": "true"
  }
]
```

2. Enregistrez le fichier sous le nom `parameters.txt`.

Pour créer le fichier accounts.txt

1. Créez un fichier contenant les comptes sur lesquels vous souhaitez créer des instances, comme indiqué dans l'exemple de fichier suivant.

```
[
    "111111222222","333333444444"
]
```

2. Enregistrez le fichier sous le nom `accounts.txt`.

Pour créer et télécharger des fichiers sources

1. Combinez les fichiers en un seul fichier ZIP. Vos fichiers devraient ressembler à ceci dans votre fichier ZIP.

```
template.yml
parameters.txt
```

```
accounts.txt
```

2. Téléchargez le fichier ZIP dans votre compartiment S3. Ce fichier est l'artefact source créé par l'assistant de création de pipeline pour votre action de déploiement dans CodePipeline.

Étape 2 : Créer votre pipeline

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Stage source avec une action source S3 où l'artefact source est votre fichier modèle et tous les fichiers sources connexes.
- Étape de déploiement avec une action de déploiement d'un ensemble de CloudFormation piles qui crée l'ensemble de piles.
- Étape de déploiement avec une action de déploiement d'instances de CloudFormation pile qui crée les piles et les instances au sein des comptes cibles.

Pour créer un pipeline avec une CloudFormationStackSet action

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Sur la page Bienvenue, Démarrez ou Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyStackSetsPipeline`**.
5. Dans Type de pipeline, choisissez V1 pour les besoins de ce didacticiel. Vous pouvez également choisir V2 ; toutefois, notez que les types de pipelines diffèrent en termes de caractéristiques et de prix. Pour de plus amples informations, veuillez consulter Types de canalisations.
6. Dans Rôle de service, choisissez Nouveau rôle de service pour permettre CodePipeline de créer un rôle de service dans IAM.
7. Dans Artifact Store, conservez les valeurs par défaut.

> **Note**
>
> Il ne s'agit pas du compartiment source de votre code source. Il s'agit du magasin d'artefacts pour votre pipeline. Un magasin d'artefacts distinct, tel qu'un compartiment S3, est nécessaire pour chaque pipeline. Lorsque vous créez ou modifiez un pipeline, vous devez disposer d'un compartiment d'artefacts dans la région du pipeline et d'un compartiment d'artefacts par AWS région dans laquelle vous exécutez une action. Pour plus d'informations, consultez Artefacts d'entrée et de sortie et CodePipeline référence de la structure du pipeline.

Choisissez Suivant.

8. Sur la page Étape 3 : Ajouter une étape source, dans Fournisseur source, choisissez Amazon S3.
9. Dans Bucket, entrez le compartiment source S3 que vous avez créé pour ce didacticiel, par exemple`BucketName`. Dans la clé d'objet S3, entrez le chemin du fichier et le nom de fichier de votre fichier ZIP, par exemple`MyFiles.zip`.
10. Choisissez Suivant.
11. À l'étape 4 : Ajouter une étape de construction, choisissez Ignorer l'étape de génération, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.
12. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.
13. À l'étape 6 : ajouter une étape de déploiement :

    a. Dans Deploy provider, choisissez AWS CloudFormation Stack Set.

    b. Dans Nom de l'ensemble de piles, entrez un nom pour l'ensemble de piles. Il s'agit du nom de l'ensemble de piles créé par le modèle.

> **Note**
>
> Notez le nom de votre stack set. Vous l'utiliserez lorsque vous ajouterez la deuxième action de StackSets déploiement à votre pipeline.

c. Dans Chemin du modèle, entrez le nom de l'artefact et le chemin du fichier dans lequel vous avez chargé votre fichier modèle. Par exemple, entrez ce qui suit en utilisant le nom `SourceArtifact` de l'artefact source par défaut.

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

d. Dans Cibles de déploiement, entrez le nom de l'artefact et le chemin du fichier dans lequel vous avez chargé le fichier de votre compte. Par exemple, entrez ce qui suit en utilisant le nom `SourceArtifact` de l'artefact source par défaut.

```
SourceArtifact::accounts.txt
```

e. Dans Cible de déploiement Régions AWS, entrez une région pour le déploiement de votre instance de stack initiale, par exemple`us-east-1`.

f. Développez les options de déploiement. Dans Paramètres, entrez le nom de l'artefact et le chemin du fichier dans lequel vous avez chargé votre fichier de paramètres. Par exemple, entrez ce qui suit en utilisant le nom `SourceArtifact` de l'artefact source par défaut.

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

Pour entrer les paramètres en tant qu'entrée littérale plutôt que sous forme de chemin de fichier, entrez ce qui suit :

```
ParameterKey=EnableLogFileValidation,ParameterValue=true
ParameterKey=IncludeGlobalEvents,ParameterValue=true
```

g. Dans Capabilities, choisissez CAPABILITY_IAM et CAPABILITY_NAMED_IAM.

h. Dans Modèle d'autorisation, choisissez SELF_MANAGED.

i. Dans Pourcentage de tolérance aux défaillances, entrez`20`.

j. Dans Pourcentage simultané maximal, entrez`25`.

k. Choisissez Suivant.

l. À l'étape 7 : Révision, choisissez Créer un pipeline. Votre pipeline s'affiche.

m. Autorisez votre pipeline s'exécuter.

Étape 3 : Afficher le déploiement initial

Consultez les ressources et l'état de votre déploiement initial. Après avoir vérifié que le déploiement a créé avec succès votre stack set, vous pouvez ajouter la deuxième action à votre phase de déploiement.

Pour consulter les ressources

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sous Pipelines, choisissez votre pipeline et choisissez Afficher. Le schéma illustre les étapes source et de déploiement de votre pipeline.
3. Choisissez l' CloudFormation action sur l'CloudFormationStackSetaction de votre pipeline. Le modèle, les ressources et les événements de votre stack set sont affichés dans la CloudFormation console.
4. Dans le panneau de navigation de gauche, choisissez StackSets. Dans la liste, choisissez le nouvel ensemble de piles.
5. Choisissez l'onglet Stack instances. Vérifiez qu'une instance de pile pour chaque compte que vous avez fourni a été créée dans la région us-east-1. Vérifiez que l'état de chaque instance de pile estCURRENT.

Étape 4 : ajouter une CloudFormationStackInstances action

Créez une action suivante dans votre pipeline qui permettra CloudFormation StackSets de créer les instances de stack restantes.

Pour créer une prochaine action dans votre pipeline

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.

   Sous Pipelines, choisissez votre pipeline et choisissez Afficher. Le schéma illustre les étapes source et de déploiement de votre pipeline.
2. Choisissez de modifier le pipeline. Le pipeline s'affiche en mode Édition.
3. Dans la phase de déploiement, choisissez Modifier.

4. Dans l'action de déploiement de AWS CloudFormation Stack Set, choisissez Ajouter un groupe d'actions.
5. Sur la page Modifier l'action, ajoutez les détails de l'action :
   a. Dans Nom de l'action, entrez le nom de l'action.
   b. Dans Action provider, choisissez AWS CloudFormation Stack Instances.
   c. Sous Artefacts d'entrée, choisissez SourceArtifact.
   d. Dans Nom de l'ensemble de piles, entrez le nom de l'ensemble de piles. Il s'agit du nom de l'ensemble de piles que vous avez fourni lors de la première action.
   e. Dans Cibles de déploiement, entrez le nom de l'artefact et le chemin du fichier dans lequel vous avez chargé le fichier de votre compte. Par exemple, entrez ce qui suit en utilisant le nom `SourceArtifact` de l'artefact source par défaut.

      ```
      SourceArtifact::accounts.txt
      ```

   f. Dans Cible de déploiement Régions AWS, entrez les régions pour le déploiement de vos instances de stack restantes, telles que `us-east-2` et `eu-central-1` comme suit :

      ```
      us-east2, eu-central-1
      ```

   g. Dans Pourcentage de tolérance aux défaillances, entrez`20`.
   h. Dans Pourcentage simultané maximal, entrez`25`.
   i. Choisissez Enregistrer.
   j. Relâchez manuellement une modification. Votre pipeline mis à jour s'affiche avec deux actions au cours de la phase de déploiement.

Étape 5 : Afficher les ressources du stack set pour votre déploiement

Vous pouvez consulter les ressources et l'état du déploiement de votre stack set.

Pour consulter les ressources

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sous Pipelines, choisissez votre pipeline, puis choisissez Afficher. Le schéma illustre les étapes source et de déploiement de votre pipeline.

3. Choisissez l' CloudFormation action sur l'**`AWS CloudFormation Stack Instances`**action de votre pipeline. Le modèle, les ressources et les événements de votre stack set sont affichés dans la CloudFormation console.
4. Dans le panneau de navigation de gauche, choisissez StackSets. Dans la liste, choisissez votre stack set.
5. Choisissez l'onglet Stack instances. Vérifiez que toutes les instances de stack restantes pour chaque compte que vous avez fourni ont été créées ou mises à jour dans les régions prévues. Vérifiez que l'état de chaque instance de pile est`CURRENT`.

Étape 6 : effectuez une mise à jour de votre stack set

Effectuez une mise à jour de votre stack set et déployez-la sur les instances. Dans cet exemple, vous modifiez également les cibles de déploiement que vous souhaitez désigner pour la mise à jour. Les instances qui ne font pas partie de la mise à jour passent à un statut obsolète.

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Dans Pipelines, choisissez votre pipeline, puis choisissez Modifier. Dans la phase de déploiement, choisissez Modifier.
3. Choisissez de modifier l'action AWS CloudFormation Stack Set dans votre pipeline. Dans Description, remplacez la description existante par une nouvelle description de l'ensemble de piles.
4. Choisissez de modifier l'action AWS CloudFormation Stack Instances dans votre pipeline. Dans Cible de déploiement Régions AWS, supprimez la `us-east-2` valeur saisie lors de la création de l'action.
5. Enregistrez les Modifications. Choisissez Release change pour exécuter votre pipeline.
6. Ouvrez votre action dans CloudFormation. Choisissez l'onglet StackSet Info. Dans StackSet la description, vérifiez que la nouvelle description est affichée.
7. Choisissez l'onglet Stack instances. Sous État, vérifiez que l'état des instances de pile dans us-east-2 est le bon. `OUTDATED`

Tutoriel : Création d'une règle de vérification des variables pour un pipeline en tant que condition d'entrée

Dans ce didacticiel, vous allez configurer un pipeline qui fournit en continu des fichiers en GitHub tant que fournisseur d'actions source dans votre stage source. Le pipeline terminé détecte les modifications apportées lorsque vous modifiez les fichiers source dans votre référentiel source. Le pipeline s'exécute puis vérifie les variables de sortie par rapport au nom du référentiel source et au nom de la branche fournis dans la condition d'entrée dans l'étape de construction.

Important

Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé par CodePipeline les artefacts. (Ce compartiment est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient Comptes AWS à une société sûre et fiable.

Important

La plupart des actions que vous ajoutez à votre pipeline au cours de cette procédure impliquent AWS des ressources que vous devez créer avant de créer le pipeline. AWS les ressources pour vos actions source doivent toujours être créées dans la même AWS région que celle dans laquelle vous créez votre pipeline. Par exemple, si vous créez votre pipeline dans la région USA Est (Ohio), votre CodeCommit référentiel doit se trouver dans la région USA Est (Ohio).

Vous pouvez ajouter des actions interrégionales lorsque vous créez votre pipeline. AWS les ressources pour les actions interrégionales doivent se trouver dans la même AWS région que celle où vous prévoyez d'exécuter l'action. Pour de plus amples informations, veuillez consulter Ajouter une action interrégionale dans CodePipeline.

Cet exemple utilise l'exemple de pipeline avec une action source GitHub (Version2) et une action de CodeBuild génération dans laquelle la condition d'entrée pour l'étape de génération vérifiera la présence de variables.

Conditions préalables

Avant de commencer, vous devez exécuter les opérations suivantes :

- Créez un GitHub référentiel avec votre GitHub compte.
- Préparez vos GitHub informations d'identification. Lorsque vous utilisez le Console de gestion AWS pour établir une connexion, il vous est demandé de vous connecter avec vos GitHub informations d'identification.
- Une connexion à votre référentiel à configurer GitHub (via une GitHub application) comme action source pour votre pipeline. Pour créer une connexion à votre GitHub référentiel, consultezGitHub connexions.

Étape 1 : créer un exemple de fichier source et l'ajouter à votre GitHub référentiel

Dans cette section, vous allez créer et ajouter votre exemple de fichier source au référentiel que le pipeline utilise pour votre stage source. Pour cet exemple, vous produisez et ajoutez les éléments suivants :

- Un fichier `README.md`.

Après avoir créé votre GitHub référentiel, procédez comme suit pour ajouter votre fichier README.

1. Connectez-vous à votre GitHub référentiel et choisissez votre référentiel.
2. Pour créer un nouveau fichier, choisissez Ajouter un fichier, puis choisissez Créer un nouveau fichier. Donnez un nom au fichier `README.md` et ajoutez le texte suivant.

   ```
   This is a GitHub repository!
   ```

3. Choisissez Valider les modifications. Pour les besoins de ce didacticiel, ajoutez un message de validation contenant le mot « Update » en majuscule, comme dans l'exemple suivant :

   ```
   Update to source files
   ```

> **Note**
>
> La vérification des règles pour les chaînes fait la distinction entre majuscules et minuscules.

Assurez-vous que le fichier `README.md` est au niveau racine de votre référentiel.

Étape 2 : Créer votre pipeline

Dans cette section, vous créez un pipeline avec les actions suivantes :

- Stage source avec connexion à votre GitHub référentiel et à votre action.
- Stage de CodeBuild construction dans lequel le stage possède une condition On Entry configurée pour la règle de vérification des variables.

Pour créer un pipeline avec l'assistant

1. Connectez-vous à la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Bienvenue, Démarrez ou Pipelines, choisissez Créer un pipeline.
3. Sur la page Étape 1 : Choisissez l'option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. À l'étape 2 : choisissez les paramètres du pipeline, dans Nom du pipeline, entrez**`MyVarCheckPipeline`**.
5. CodePipeline fournit des pipelines de type V1 et V2, dont les caractéristiques et le prix diffèrent. Le type V2 est le seul que vous pouvez choisir dans la console. Pour plus d'informations, consultez types de pipeline. Pour plus d'informations sur la tarification de CodePipeline, consultez la section Tarification.
6. Dans Service role (Rôle de service), choisissez New service role (Nouveau rôle de service).

> **Note**
>
> Si vous choisissez plutôt d'utiliser votre rôle de CodePipeline service existant, assurez-vous d'avoir ajouté l'autorisation `codeconnections:UseConnection` IAM à

votre politique de rôle de service. Pour obtenir des instructions relatives au rôle de CodePipeline service, voir Ajouter des autorisations au rôle CodePipeline de service.

7. Sous Paramètres avancés, conservez les valeurs par défaut.

   Choisissez Suivant.

8. Sur la page Étape 3 : Ajouter une étape source, ajoutez une étape source :

   a. Dans Source provider, choisissez GitHub (via GitHub App).

   b. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.

   c. Dans Nom du référentiel, choisissez le nom de votre référentiel GitHub .

   d. Dans Nom de la branche, choisissez la branche du référentiel que vous souhaitez utiliser.

   e. Assurez-vous que l'option Aucun déclencheur est sélectionnée.

   Choisissez Suivant.

9. À l'étape 4 : Ajouter une étape de construction, ajoutez une étape de construction :

   a. Dans le champ Fournisseur de génération, choisissez AWS CodeBuild. Acceptez la région du pipeline comme Région par défaut.

   b. Sélectionnez Create a project (Créer un projet).

   c. Dans Nom du projet, saisissez un nom pour ce projet de génération.

   d. Dans le champ Image d'environnement, choisissez Image gérée. Pour Système d'exploitation, choisissez Ubuntu.

   e. Pour Runtime (Exécution), sélectionnez Standard. Pour Image, choisissez aws/codebuild / standard:5.0.

   f. Pour Rôle de service, choisissez Nouveau rôle de service.

   > **Note**
   >
   > Notez le nom de votre rôle CodeBuild de service. Vous aurez besoin du nom du rôle pour la dernière étape de ce didacticiel.

g. Sous Buildspec, pour Build specifications (Spécifications de génération), choisissez Insert build commands (Insérer des commandes de génération). Choisissez Passer à l'éditeur et collez ce qui suit sous Commandes de création.

```
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. Choisissez Continuer vers CodePipeline. Cela revient à la CodePipeline console et crée un CodeBuild projet qui utilise vos commandes de build pour la configuration. Le projet de génération utilise un rôle de service pour gérer les Service AWS autorisations. Cette étape peut prendre quelques minutes.

i. Choisissez Suivant.

10. À l'étape 5 : Ajouter une phase de test, choisissez Ignorer la phase de test, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer.

    Choisissez Suivant.

11. Sur la page Étape 6 : Ajouter une phase de déploiement, choisissez Ignorer l'étape de déploiement, puis acceptez le message d'avertissement en choisissant de nouveau Ignorer. Choisissez Suivant.

12. À l'étape 7 : Révision, choisissez Créer un pipeline.

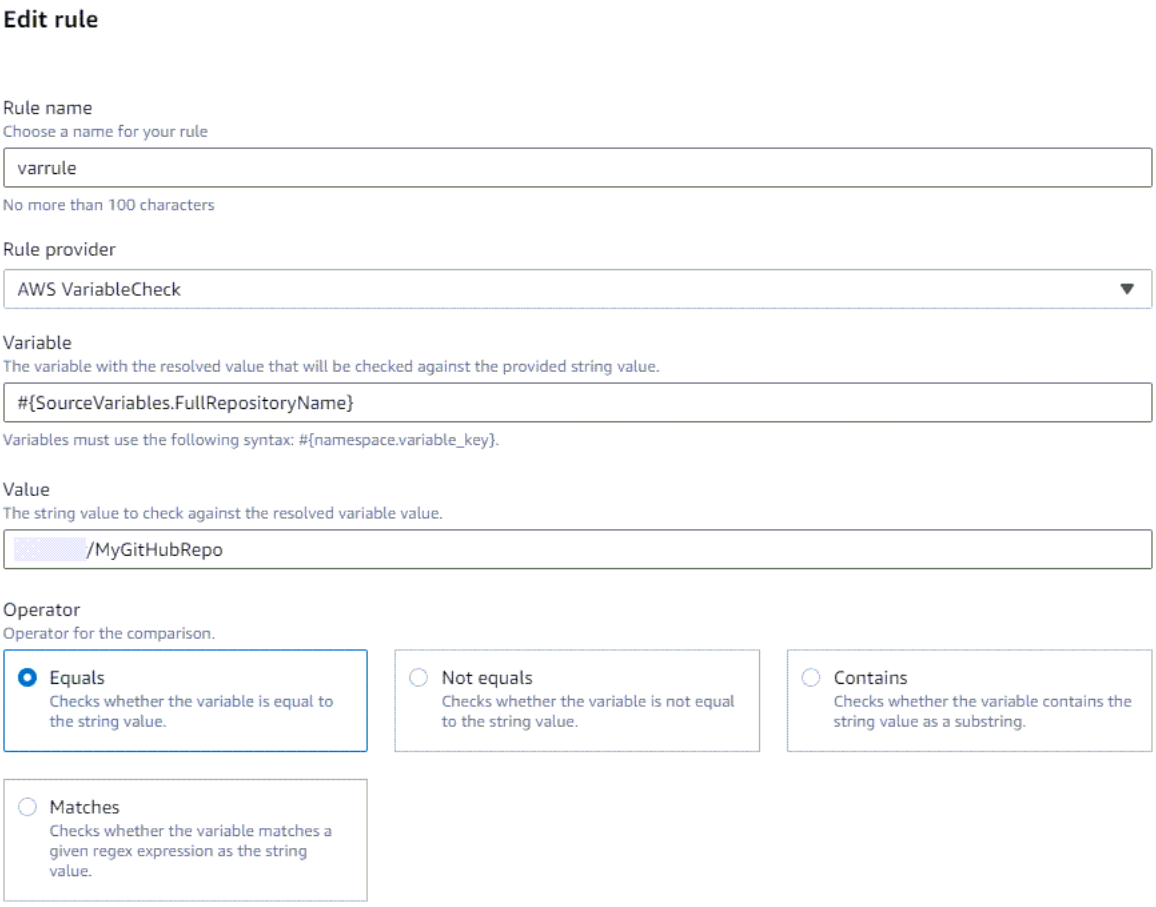
Étape 2 : modifier l'étape de construction pour ajouter la condition et la règle

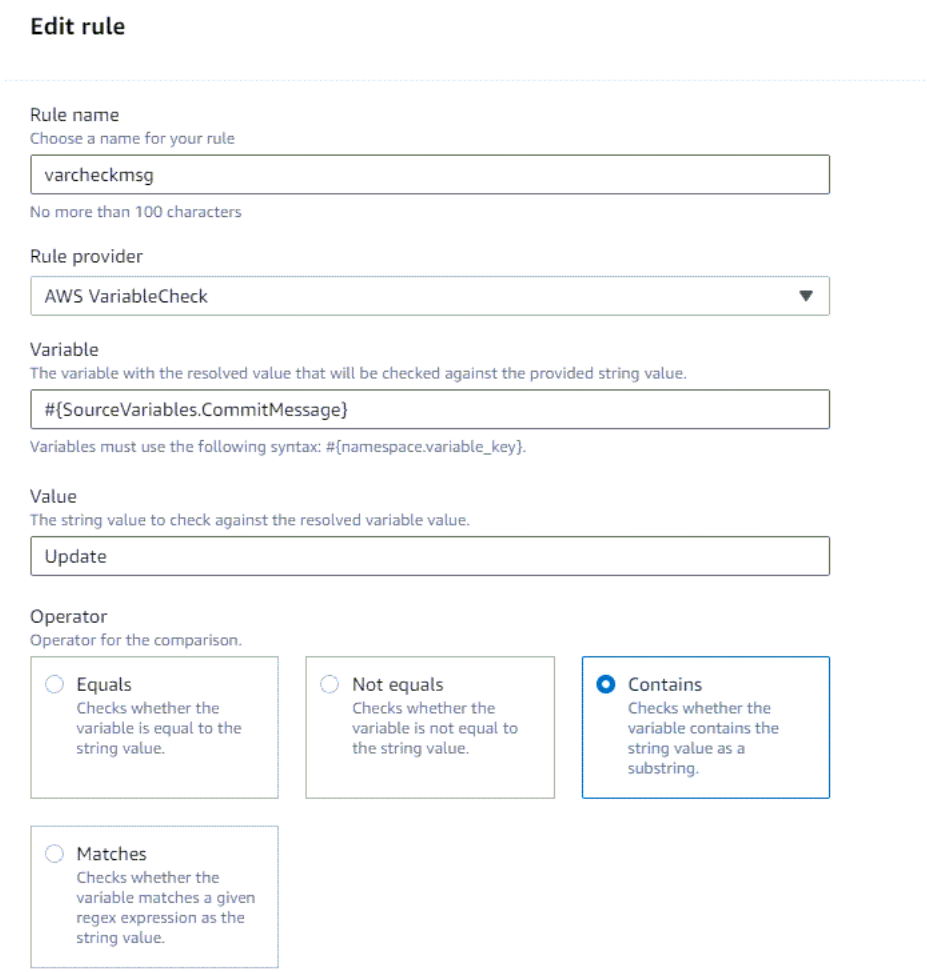
Au cours de cette étape, vous modifiez l'étape pour ajouter une condition On Entry pour la règle de vérification des variables.

1. Choisissez votre pipeline, puis choisissez Modifier. Choisissez d'ajouter une règle de saisie lors de la phase de construction.

   Dans Fournisseur de règles, sélectionnez VariableCheck.

2. Dans Variable, entrez la ou les variables que vous souhaitez vérifier. Dans Valeur, entrez la valeur de chaîne à comparer à la variable résolue. Dans les exemples d'écrans suivants, une règle est créée pour un contrôle « égal » et une autre règle est créée pour un contrôle « contient ».


Edit rule
Rule name
Choose a name for your rule
varrule
No more than 100 characters
Rule provider
AWS VariableCheck
Variable
The variable with the resolved value that will be checked against the provided string value.
#{SourceVariables.FullRepositoryName}
Variables must use the following syntax: #{namespace.variable_key}.
Value
The string value to check against the resolved variable value.
/MyGitHubRepo
Operator
Operator for the comparison.
Equals
Checks whether the variable is equal to the string value.
Not equals
Checks whether the variable is not equal to the string value.
Contains
Checks whether the variable contains the string value as a substring.
Matches
Checks whether the variable matches a given regex expression as the string value.



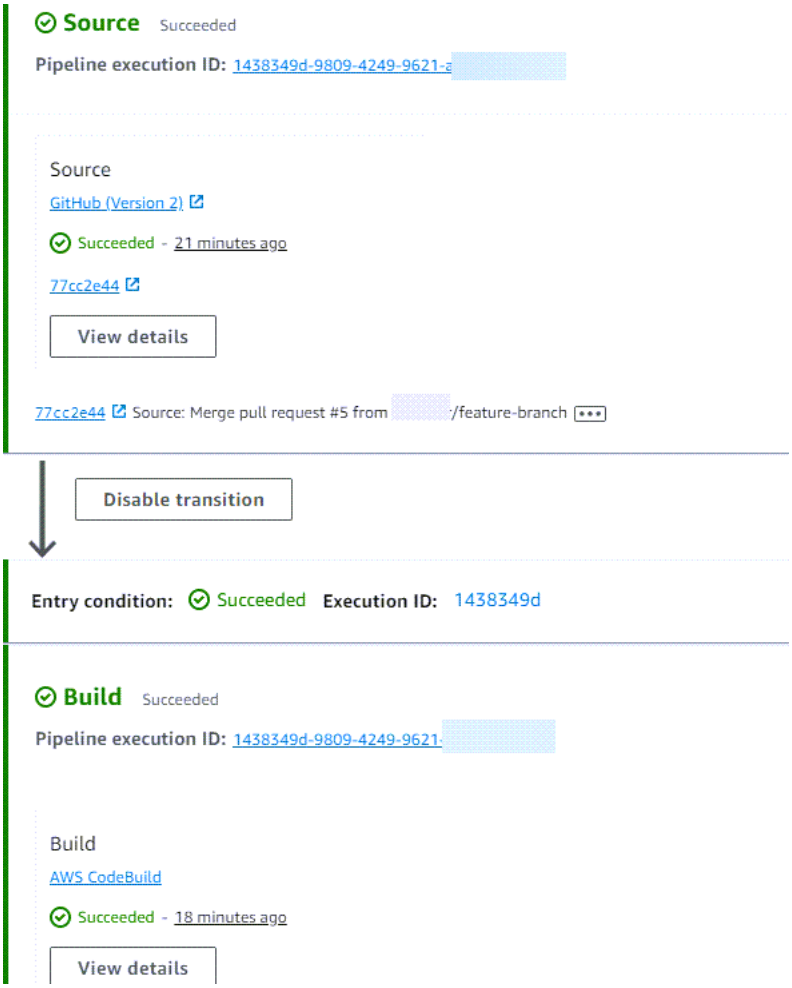
3. Choisissez Enregistrer.

   Sélectionnez Exécuté.

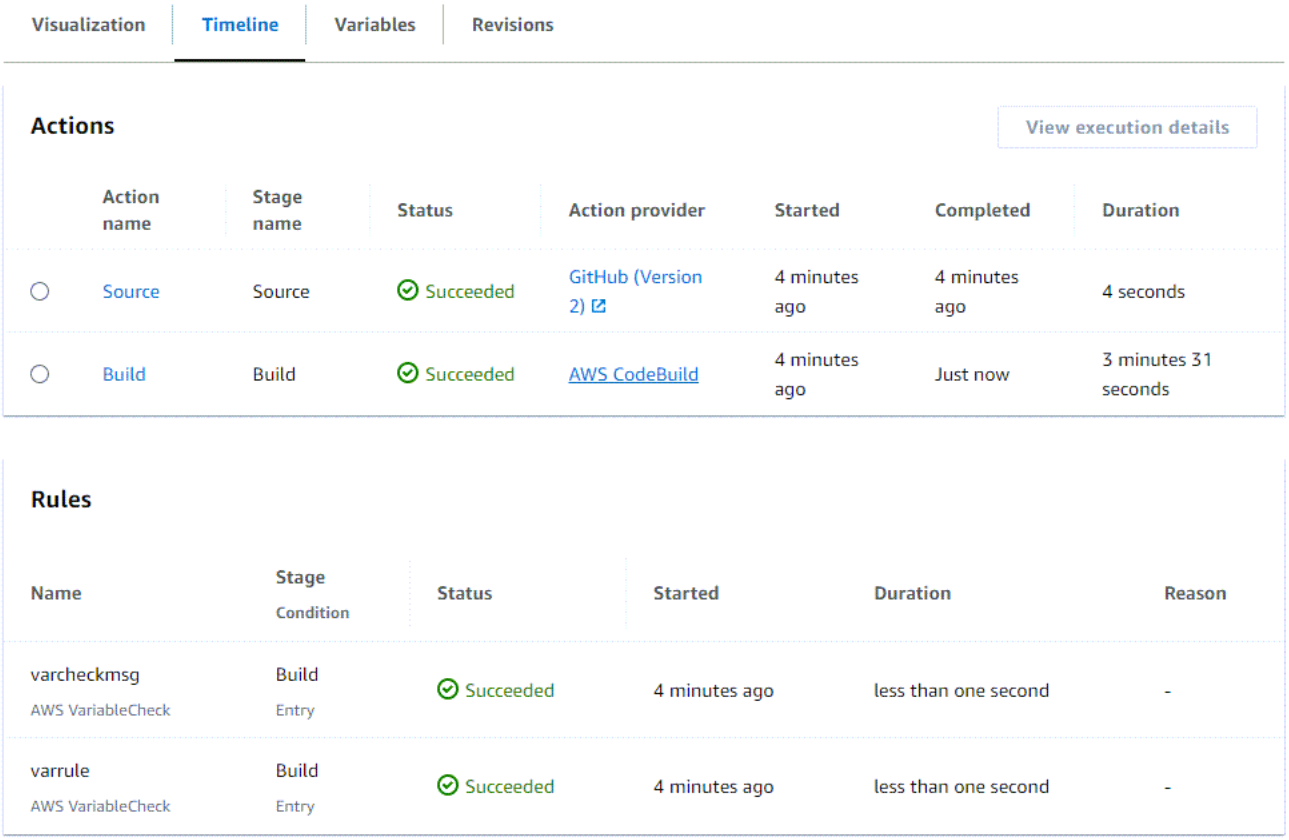
Étape 3 : Exécuter le pipeline et afficher les variables résolues

Au cours de cette étape, vous visualisez les valeurs résolues et les résultats de la règle de vérification des variables.

1. Affichez l'exécution résolue une fois la vérification des règles réussie, comme indiqué dans l'exemple suivant.



2. Consultez les informations sur les variables dans l'onglet Chronologie.


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 cas d'utilisation

Les sections suivantes décrivent les cas d'utilisation de CodePipeline.

Rubriques

- Cas d'utilisation pour CodePipeline

Cas d'utilisation pour CodePipeline

Vous pouvez créer des pipelines qui s'intègrent à d'autres Services AWS. Il peut s'agir Services AWS, par exemple, d'Amazon S3 ou de produits tiers, tels que GitHub. Cette section fournit des exemples d'automatisation CodePipeline de vos publications de code à l'aide de différentes intégrations de produits. Pour une liste complète des intégrations CodePipeline organisées par type d'action, consultezCodePipeline référence de la structure du pipeline.

Rubriques

- À utiliser CodePipeline avec Amazon S3, AWS CodeCommit, et AWS CodeDeploy
- Utilisation CodePipeline avec des fournisseurs d'actions tiers (GitHub et Jenkins)
- CodePipeline À utiliser pour compiler, construire et tester du code avec CodeBuild
- CodePipeline À utiliser avec Amazon ECS pour la livraison continue d'applications basées sur des conteneurs vers le cloud
- Utilisation CodePipeline avec Elastic Beanstalk pour la diffusion continue d'applications Web dans le cloud
- À utiliser CodePipeline avec AWS Lambda pour la livraison continue Lambda-based et les applications sans serveur
- À utiliser CodePipeline avec CloudFormation modèles pour une diffusion continue dans le cloud

À utiliser CodePipeline avec Amazon S3, AWS CodeCommit, et AWS CodeDeploy

Lorsque vous créez un pipeline, il CodePipeline s'intègre à AWS des produits et services qui agissent en tant que fournisseurs d'actions à chaque étape de votre pipeline. Lorsque vous choisissez les étapes dans l'assistant, vous devez choisir une étape source et au moins une étape de génération

ou de déploiement. L'assistant crée les étapes pour vous avec des noms par défaut qui ne peuvent pas être modifiés. Il s'agit des noms d'étape créés lorsque vous configurez un pipeline à trois niveaux complet dans l'assistant :

- Une étape d'action source portant le nom par défaut « Source »
- Une étape d'action de génération portant le nom par défaut « Build »
- Une étape d'action de déploiement portant le nom par défaut « Staging »

Vous pouvez utiliser les didacticiels de ce guide pour créer des pipelines et spécifier des étapes :

- Les étapes ci-dessous vous Didacticiel : Création d'un pipeline simple (compartiment S3) aident à utiliser l'assistant pour créer un pipeline avec deux étapes par défaut : « Source » et « Staging », où votre référentiel Amazon S3 est le fournisseur de source. Ce didacticiel crée un pipeline qui permet AWS CodeDeploy de déployer un exemple d'application depuis un compartiment Amazon S3 vers des instances Amazon EC2 exécutant Amazon Linux.
- Les étapes ci-dessous vous Tutoriel : Création d'un pipeline simple (CodeCommit référentiel) aident à utiliser l'assistant pour créer un pipeline avec une étape « Source » qui utilise votre AWS CodeCommit référentiel comme fournisseur de source. Ce didacticiel crée un pipeline qui permet AWS CodeDeploy de déployer un exemple d'application depuis un AWS CodeCommit référentiel vers une instance Amazon EC2 exécutant Amazon Linux.

Utilisation CodePipeline avec des fournisseurs d'actions tiers (GitHub et Jenkins)

Vous pouvez créer des pipelines qui s'intègrent à des produits tiers tels que GitHub Jenkins. Les étapes décrites dans Didacticiel : Création d'un pipeline à quatre étapes vous montrent comment créer un pipeline qui :

- Récupère le code source d'un GitHub dépôt,
- utilise Jenkins pour générer et tester le code source ;
- Utilisée AWS CodeDeploy pour déployer le code source créé et testé sur des instances Amazon EC2 exécutant Amazon Linux ou Microsoft Windows Server.

CodePipeline À utiliser pour compiler, construire et tester du code avec CodeBuild

CodeBuild est un service de génération géré dans le cloud qui vous permet de créer et de tester votre code sans serveur ni système. Utilisez CodePipeline with CodeBuild pour automatiser l'exécution des révisions dans le pipeline afin de fournir en continu les versions logicielles chaque fois que le code source est modifié. Pour plus d'informations, consultez Utiliser CodePipeline avec CodeBuild pour tester le code et exécuter des builds.

CodePipeline À utiliser avec Amazon ECS pour la livraison continue d'applications basées sur des conteneurs vers le cloud

Amazon ECS est un service de gestion de conteneurs qui vous permet de déployer des applications basées sur des conteneurs sur des instances Amazon ECS dans le cloud. CodePipeline Utilisez-le avec Amazon ECS pour automatiser l'exécution des révisions dans le pipeline afin de déployer en continu des applications basées sur des conteneurs chaque fois que le référentiel d'images source est modifié. Pour plus d'informations, consultez Didacticiel : Déploiement continu avec CodePipeline.

Utilisation CodePipeline avec Elastic Beanstalk pour la diffusion continue d'applications Web dans le cloud

Elastic Beanstalk est un service informatique qui vous permet de déployer des applications et des services Web sur des serveurs Web. Utilisez-le CodePipeline avec Elastic Beanstalk pour le déploiement continu d'applications Web dans votre environnement applicatif. Vous pouvez également l'utiliser AWS CodeStar pour créer un pipeline avec une action de déploiement d'Elastic Beanstalk.

À utiliser CodePipeline avec AWS Lambda pour la livraison continue Lambda-based et les applications sans serveur

Vous pouvez utiliser AWS Lambda with CodePipeline pour appeler une AWS Lambda fonction, comme décrit dans Déploiement d'applications sans serveur. Vous pouvez également utiliser AWS Lambda et AWS CodeStar créer un pipeline pour déployer des applications sans serveur.

À utiliser CodePipeline avec CloudFormation modèles pour une diffusion continue dans le cloud

Vous pouvez l'utiliser CloudFormation CodePipeline pour la livraison continue et l'automatisation. Pour plus d'informations, consultez la section Livraison continue avec CodePipeline. CloudFormation est également utilisé pour créer les modèles pour les pipelines créés dans AWS CodeStar.

Utilisation CodePipeline avec Amazon Virtual Private Cloud

AWS CodePipeline prend désormais en charge les points de terminaison Amazon Virtual Private Cloud (Amazon VPC) alimentés par. AWS PrivateLink Cela signifie que vous pouvez vous connecter directement CodePipeline via un point de terminaison privé dans votre VPC, en conservant tout le trafic à l'intérieur de votre VPC et du réseau. AWS

Amazon VPC est un système Service AWS que vous pouvez utiliser pour lancer AWS des ressources dans un réseau virtuel que vous définissez. Avec un VPC, vous contrôlez vos paramètres réseau, tels que :

- Plage d'adresses IP
- Subnets
- Tables de routage
- Passerelles réseau

Les points de terminaison VPC d'interface sont alimentés par AWS PrivateLink une AWS technologie qui facilite la communication privée entre les Services AWS utilisateurs d'une interface Elastic Network avec des adresses IP privées. Pour connecter votre VPC à CodePipeline, vous définissez un point de terminaison VPC d'interface pour. CodePipeline Ce type de point de terminaison vous permet de connecter votre VPC à. Services AWS Le point de terminaison fournit une connectivité fiable et évolutive CodePipeline sans nécessiter de passerelle Internet, d'instance de traduction d'adresses réseau (NAT) ou de connexion VPN. Pour plus d'informations sur la configuration d'un VPC, consultez le Guide de l'utilisateur VPC.

Disponibilité

CodePipeline prend actuellement en charge les points de terminaison VPC dans les domaines suivants : Régions AWS

- USA Est (Ohio)
- USA Est (Virginie du Nord)
- USA Ouest (Californie du Nord)
- US West (Oregon)
- Canada (Centre)

- Europe (Francfort)
- Europe (Irlande)
- Europe (Londres)
- Europe (Milan) *
- Europe (Paris)
- Europe (Stockholm)
- Asie-Pacifique (Hong Kong) *
- Asie-Pacifique (Mumbai)
- Asie-Pacifique (Tokyo)
- Asie-Pacifique (Séoul)
- Asie-Pacifique (Singapour)
- Asie-Pacifique (Sydney)
- Amérique du Sud (São Paulo)
- AWS GovCloud (US-Ouest)

* Vous devez activer cette région avant de pouvoir l'utiliser.

Créez un point de terminaison VPC pour CodePipeline

Vous pouvez utiliser la console Amazon VPC pour créer le fichier com.amazonaws. ***`region`***Point de terminaison VPC .codepipeline. Dans la console, *`region`* il s'agit de l'identifiant de région d'une région Région AWS prise en charge par CodePipeline, par exemple `us-east-2` pour la région USA Est (Ohio). Pour plus d'informations, consultez Création d'un point de terminaison d'interface dans le Amazon VPC Guide de l'utilisateur.

Le point de terminaison est prérempli avec la région que vous avez spécifiée lorsque vous vous êtes connecté à AWS. Si vous vous connectez à une autre région, le point de terminaison de VPC est mis à jour avec la nouvelle région.

> **Note**
>
> D'autres Services AWS solutions qui fournissent un support VPC et s'intègrent CodePipeline, par exemple CodeCommit, peuvent ne pas prendre en charge l'utilisation des points de

terminaison Amazon VPC pour cette intégration. Par exemple, le trafic entre CodePipeline et CodeCommit ne peut pas être limité à la plage de sous-réseaux VPC.

Dépannage de la configuration de votre VPC

Lors du dépannage des problèmes de VPC, utilisez les informations affichées dans les messages d'erreur liés à la connectivité Internet pour vous aider à identifier, à diagnostiquer et à résoudre les problèmes.

1. Vérifiez que votre passerelle Internet est attachée à votre VPC.
2. Assurez-vous que la table de routage de votre sous-réseau public pointe vers la passerelle Internet.
3. Assurez-vous que votre réseau ACLs autorise la circulation du trafic.
4. Assurez-vous que vos groupes de sécurité autorisent la circulation du trafic.
5. Assurez-vous que la table de routage des sous-réseaux privés pointe vers la passerelle privée virtuelle.
6. Assurez-vous que le rôle de service utilisé par CodePipeline dispose des autorisations appropriées. Par exemple, si vous CodePipeline ne disposez pas des autorisations Amazon EC2 requises pour travailler avec un Amazon VPC, vous pouvez recevoir un message d'erreur indiquant « Erreur EC2 inattendue : ». UnauthorizedOperation

Définissez des CI/CD pipelines avec des étapes et des actions

Pour définir un processus de publication automatisé dans AWS CodePipeline, vous créez un pipeline, qui est une structure de flux de travail qui décrit la manière dont les modifications logicielles passent par un processus de publication. Le pipeline est composé des étapes et actions que vous configurez.

> **Note**
>
> Lorsque vous ajoutez des étapes Build, Deploy, Test ou Invoke, en plus des options par défaut fournies CodePipeline, vous pouvez choisir des actions personnalisées que vous avez déjà créées pour être utilisées avec vos pipelines. Les actions personnalisées peuvent être utilisées pour des tâches telles que l'exécution d'un processus de génération développé en interne ou d'une suite de tests. Des identificateurs de version sont inclus pour vous aider à faire la distinction entre les différentes versions d'une action personnalisée dans les listes de fournisseurs. Pour de plus amples informations, veuillez consulter Créez et ajoutez une action personnalisée dans CodePipeline.

Avant de créer un pipeline, vous devez d'abord suivre les étapes décrites dans Commencer avec CodePipeline.

Pour plus d'informations sur les pipelinesCodePipeline concepts, consultezCodePipeline tutoriels, et, si vous souhaitez utiliser le AWS CLI pour créer un pipeline,Création d'un pipeline, d'étapes et d'actions. Pour afficher la liste des pipelines, consultez Afficher les pipelines et les détails dans CodePipeline.

Rubriques

- Création d'un pipeline, d'étapes et d'actions
- Modifier un pipeline dans CodePipeline
- Afficher les pipelines et les détails dans CodePipeline
- Supprimer un pipeline dans CodePipeline
- Créez un pipeline CodePipeline qui utilise les ressources d'un autre AWS compte
- Migrez les pipelines de sondage pour utiliser la détection des modifications basée sur les événements

- Création du rôle CodePipeline de service
- Balisage des ressources
- Marquer un pipeline dans CodePipeline
- Création d'une règle de notification

Création d'un pipeline, d'étapes et d'actions

Vous pouvez utiliser la AWS CodePipeline console ou le AWS CLI pour créer un pipeline. Les pipelines doivent contenir au moins deux étapes. La première étape d'un pipeline doit être une étape source. Le pipeline doit avoir au moins une autre étape qui est une étape de génération ou de déploiement.

> **Important**
>
> Dans le cadre de la création d'un pipeline, un compartiment d'artefacts S3 fourni par le client sera utilisé CodePipeline pour les artefacts. (Ceci est différent du compartiment utilisé pour une action source S3.) Si le compartiment d'artefacts S3 se trouve dans un compte différent de celui de votre pipeline, assurez-vous que le compartiment d'artefacts S3 appartient à cette personne sûre et Comptes AWS qu'elle sera fiable.

Vous pouvez ajouter à votre pipeline des actions qui se trouvent dans un Région AWS autre pipeline. Une action interrégionale est une action dans laquelle an Service AWS est le fournisseur d'une action et le type d'action ou le type de fournisseur se trouve dans une AWS région différente de votre pipeline. Pour plus d'informations, consultez Ajouter une action interrégionale dans CodePipeline.

Vous pouvez également créer des pipelines qui créent et déploient des applications basées sur des conteneurs en utilisant Amazon ECS comme fournisseur de déploiement. Avant de créer un pipeline qui déploie des applications basées sur des conteneurs avec Amazon ECS, vous devez créer un fichier de définitions d'images comme décrit dans. Référence pour les fichiers de définitions d'image

CodePipeline utilise des méthodes de détection des modifications pour démarrer votre pipeline lorsqu'une modification du code source est poussée. Ces méthodes de détection sont basées sur le type de source :

- CodePipeline utilise Amazon CloudWatch Events pour détecter les modifications apportées à votre référentiel CodeCommit source et à votre branche ou à votre compartiment source S3.

> **Note**
>
> Lorsque vous utilisez la console pour créer ou modifier un pipeline, les ressources de détection des modifications sont créées pour vous. Si vous utilisez la AWS CLI pour créer le pipeline, vous devez créer vous-même les ressources supplémentaires. Pour plus d'informations, consultez CodeCommit actions à la source et EventBridge.

Rubriques

- Création d'un pipeline personnalisé (console)
- Création d'un pipeline (interface de ligne de commande)
- Création d'un pipeline à partir de modèles statiques

Création d'un pipeline personnalisé (console)

Pour créer un pipeline personnalisé dans la console, vous devez fournir l'emplacement du fichier source et des informations sur les fournisseurs que vous utiliserez pour vos actions.

Lorsque vous utilisez la console pour créer un pipeline, vous devez inclure une étape source et l'une des actions suivantes :

- Une étape de génération.
- Une étape de déploiement.

Lorsque vous utilisez l'assistant de pipeline, il CodePipeline crée les noms des étapes (source, build, staging). Ces noms ne peuvent pas être modifiés. Vous pouvez utiliser des noms plus spécifiques (par exemple, BuildToGamma ou DeployToProd) pour les étapes que vous ajouterez ultérieurement.

Étape 1 : Créer et nommer votre pipeline

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Sur la page Welcome (Bienvenue), choisissez Créer un pipeline.

   Si c'est la première fois que vous l'utilisez CodePipeline, choisissez Get Started.

3. Sur la page Étape 1 : Choisir une option de création, sous Options de création, choisissez l'option Créer un pipeline personnalisé. Choisissez Suivant.
4. Sur la page Étape 2 : Choisir les paramètres du pipeline, dans Nom du pipeline, entrez le nom de votre pipeline.

   Dans un seul AWS compte, chaque pipeline que vous créez dans une AWS région doit porter un nom unique. Les noms peuvent être réutilisés pour des pipelines de régions différentes.

   > **Note**
   >
   > Une fois le pipeline créé, vous ne pouvez plus modifier son nom. Pour plus d'informations sur les autres limitations, consultez Quotas dans AWS CodePipeline.

5. Dans Type de pipeline, choisissez l'une des options suivantes. Les types de pipelines diffèrent en termes de caractéristiques et de prix. Pour plus d'informations, consultez Types de canalisations.

   - Les pipelines de type V1 ont une structure JSON qui contient des paramètres standard au niveau du pipeline, de l'étape et de l'action.
   - Les pipelines de type V2 ont la même structure que les pipelines de type V1, avec une prise en charge de paramètres supplémentaires, tels que les déclencheurs sur les balises Git et les variables au niveau du pipeline.

6. Dans Rôle du service, sélectionnez l'une des options suivantes :

   - Choisissez Nouveau rôle de service pour autoriser CodePipeline la création d'un nouveau rôle de service dans IAM.
   - Choisissez Existing service role (Rôle de service existant) pour utiliser un rôle de service déjà créé dans IAM. Dans ARN de rôle, choisissez votre ARN de rôle de service dans la liste.

   > **Note**
   >
   > En fonction de la date de création de votre rôle de service, vous devrez peut-être mettre à jour ses autorisations pour en prendre en charge d'autres Services AWS. Pour plus d'informations, consultez Ajouter des autorisations au rôle CodePipeline de service.

Pour plus d'informations sur le rôle de service et sa déclaration de stratégie, consultez Gérer le rôle CodePipeline de service.

7. (Facultatif) Sous Variables, choisissez Ajouter une variable pour ajouter des variables au niveau du pipeline.

   Pour plus d'informations sur les variables au niveau du pipeline, consultezRéférence des variables. Pour un didacticiel avec une variable au niveau du pipeline transmise au moment de l'exécution du pipeline, voir. Tutoriel : Utiliser des variables au niveau du pipeline

   > **Note**
   >
   > Bien qu'il soit facultatif d'ajouter des variables au niveau du pipeline, pour un pipeline spécifié avec des variables au niveau du pipeline pour lequel aucune valeur n'est fournie, l'exécution du pipeline échouera.

8. (Facultatif) Développez Advanced settings (Paramètres avancés).
9. Dans Magasin d'artefacts, effectuez l'une des actions suivantes :

   a. Choisissez Emplacement par défaut pour utiliser le magasin d'artefacts par défaut, tel que le compartiment d'artefacts S3 désigné par défaut, pour votre pipeline dans le répertoire que Région AWS vous avez sélectionné pour votre pipeline.

   b. Choisissez Custom location (Emplacement personnalisé) si vous disposez déjà d'un magasin d'artefacts, par exemple, un compartiment d'artefacts S3, dans la même région que votre pipeline. Dans Bucket (Compartiment), choisissez le nom du compartiment.

   > **Note**
   >
   > Il ne s'agit pas du compartiment source de votre code source. Il s'agit du magasin d'artefacts pour votre pipeline. Un magasin d'artefacts distinct, tel qu'un compartiment S3, est nécessaire pour chaque pipeline. Lorsque vous créez ou modifiez un pipeline, vous devez disposer d'un compartiment d'artefacts dans la région du pipeline et d'un compartiment d'artefacts par AWS région dans laquelle vous exécutez une action. Pour plus d'informations, consultez Artefacts d'entrée et de sortie et CodePipeline référence de la structure du pipeline.

10. Dans Encryption key (Clé de chiffrement), effectuez l'une des opérations suivantes :

    a. Pour utiliser la CodePipeline valeur par défaut AWS KMS key pour chiffrer les données dans le magasin d'artefacts du pipeline (compartiment S3), choisissez Default AWS Managed Key.

    b. Pour utiliser votre clé gérée par le client afin de chiffrer les données dans le magasin d'artefacts du pipeline (compartiment S3), choisissez Clé gérée par le client. Choisissez l'ID de clé, l'ARN de clé ou l'alias ARN.

11. Choisissez Suivant.

Étape 2 : Créer une étape source

1. Sur la page Étape 3 : Ajouter une étape source, dans Source provider, choisissez le type de référentiel dans lequel votre code source est stocké, puis spécifiez les options requises. Les champs supplémentaires s'affichent en fonction du fournisseur de source sélectionné comme suit.

   - Pour Bitbucket Cloud GitHub (via une GitHub application), GitHub Enterprise Server ou GitLab autogéré : GitLab.com

     1. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.
     2. Choisissez le référentiel que vous souhaitez utiliser comme emplacement source pour votre pipeline.

        Choisissez d'ajouter un déclencheur ou de filtrer les types de déclencheurs pour démarrer votre pipeline. Pour plus d'informations sur l'utilisation des déclencheurs, consultezAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ». Pour plus d'informations sur le filtrage à l'aide de modèles globulaires, consultezUtilisation de modèles globulaires dans la syntaxe.
     3. Dans Format d'artefact de sortie, choisissez le format de vos artefacts.
        - Pour stocker les artefacts de sortie de l' GitHub action à l'aide de la méthode par défaut, choisissez CodePipelinepar défaut. L'action accède aux fichiers depuis le GitHub référentiel et stocke les artefacts dans un fichier ZIP dans le magasin d'artefacts du pipeline.

- Pour stocker un fichier JSON contenant une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git, choisissez Full clone (Clone complet). Cette option ne peut être utilisée que par les actions CodeBuild en aval.

  Si vous choisissez cette option, vous devrez mettre à jour les autorisations associées à votre rôle de service de CodeBuild projet, comme indiqué dansRésolution des problèmes CodePipeline. Pour consulter un didacticiel expliquant comment utiliser l'option de clonage complet, voirTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

- Pour Amazon S3 :

  1. Dans Emplacement Amazon S3, indiquez le nom du compartiment S3 et le chemin d'accès à l'objet dans un compartiment où la gestion des versions est activée. Le format du nom et du chemin du compartiment ressemblent à l'exemple suivant :

     ```
     s3://bucketName/folderName/objectName
     ```

     > **Note**
     >
     > Lorsque Amazon S3 est le fournisseur source de votre pipeline, vous pouvez compresser votre ou vos fichiers source dans un seul fichier .zip et télécharger le fichier .zip dans votre compartiment source. Vous pouvez également charger un seul fichier décompressé ; toutefois, les actions en aval qui attendent un fichier .zip échoueront.

  2. Après avoir choisi le compartiment source S3, vous CodePipeline créez la règle Amazon CloudWatch Events et AWS CloudTrail le journal à créer pour ce pipeline. Acceptez les valeurs par défaut sous Options de détection des modifications. Cela permet d' CodePipeline utiliser Amazon CloudWatch Events et AWS CloudTrail de détecter les modifications apportées à votre nouveau pipeline. Choisissez Suivant.

- Pour AWS CodeCommit :

  - Dans Nom du référentiel, choisissez le nom du CodeCommit référentiel que vous souhaitez utiliser comme emplacement source pour votre pipeline. Dans Nom de branche, depuis la liste déroulante, choisissez la branche que vous souhaitez utiliser.
  - Dans Format d'artefact de sortie, choisissez le format de vos artefacts.
    - Pour stocker les artefacts de sortie de l' CodeCommit action à l'aide de la méthode par défaut, choisissez CodePipelinepar défaut. L'action accède aux fichiers depuis le

CodeCommit référentiel et stocke les artefacts dans un fichier ZIP dans le magasin d'artefacts du pipeline.

- Pour stocker un fichier JSON contenant une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git, choisissez Full clone (Clone complet). Cette option ne peut être utilisée que par les actions CodeBuild en aval.

  Si vous choisissez cette option, vous devrez ajouter l'`codecommit:GitPull`autorisation à votre rôle de CodeBuild service, comme indiqué dansAjouter CodeBuild GitClone des autorisations pour les actions CodeCommit source. Vous devrez également ajouter les `codecommit:GetRepository` autorisations à votre rôle de CodePipeline service, comme indiqué dansAjouter des autorisations au rôle CodePipeline de service. Pour consulter un didacticiel expliquant comment utiliser l'option de clonage complet, voirTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

- Une fois que vous avez choisi le nom du CodeCommit référentiel et la branche, un message s'affiche dans les options de détection des modifications indiquant la règle Amazon CloudWatch Events à créer pour ce pipeline. Acceptez les valeurs par défaut sous Options de détection des modifications. Cela permet d' CodePipeline utiliser Amazon CloudWatch Events pour détecter les modifications apportées à votre nouveau pipeline.

- Pour Amazon ECR :
  - Dans Nom du référentiel, choisissez le nom de votre référentiel Amazon ECR.
  - Dans Balise d'image, spécifiez le nom et la version de l'image, s'ils sont différents de LATEST.
  - Dans Artefacts de sortie, choisissez l'artefact de sortie par défaut, par exemple MyApp, qui contient le nom de l'image et les informations d'URI du référentiel que vous souhaitez utiliser à l'étape suivante.

    Pour un didacticiel sur la création d'un pipeline pour Amazon ECS avec des déploiements CodeDeploy bleu-vert incluant un stage source Amazon ECR, consultez. Tutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR

  Lorsque vous incluez un stage source Amazon ECR dans votre pipeline, l'action source génère un `imageDetail.json` fichier en tant qu'artefact de sortie lorsque vous validez une modification. Pour de plus amples informations concernant le fichier `imageDetail.json`, veuillez consulter Detail.json fichier image pour les actions de blue/green déploiement d'Amazon ECS.

> **Note**
>
> L'objet et le type de fichier doivent être compatibles avec le système de déploiement que vous prévoyez d'utiliser (par exemple, Elastic Beanstalk ou). CodeDeploy Les types de fichier .zip, .tar et .tgz peuvent être pris en charge. Pour plus d'informations sur les types de conteneurs pris en charge pour Elastic Beanstalk, consultez Personnalisation et configuration des environnements Elastic Beanstalk et des plateformes prises en charge. Pour plus d'informations sur le déploiement de révisions avec CodeDeploy, consultez les sections Chargement de la révision de votre application et Préparation d'une révision.

2. Pour configurer l'étape pour une nouvelle tentative automatique, choisissez Activer la nouvelle tentative automatique en cas d'échec de l'étape. Pour plus d'informations sur la nouvelle tentative automatique, consultezConfigurer une étape pour une nouvelle tentative automatique en cas d'échec.
3. Choisissez Suivant.

Étape 4 : Création d'une phase de construction

Cette étape est facultative si vous prévoyez de créer une étape de déploiement.

1. Sur la page Étape 4 : Ajouter une phase de construction, effectuez l'une des opérations suivantes, puis choisissez Next :

   - Choisissez Ignorer l'étape de construction si vous prévoyez de créer une phase de test ou de déploiement.
   - Pour choisir l'action Commandes pour votre phase de construction, sélectionnez Commandes.

   > **Note**
   >
   > L'exécution de l'action Commandes entraînera des frais distincts. AWS CodeBuild Si vous prévoyez d'insérer des commandes de compilation dans le cadre CodeBuild d'une action, choisissez Autres fournisseurs de build, puis sélectionnez CodeBuild.

   Dans Commandes, entrez les commandes shell pour votre action. Pour plus d'informations sur l'action Commandes, consultezRéférence des actions des commandes.

- Pour choisir d'autres fournisseurs de build CodeBuild, par exemple, choisissez Autres fournisseurs. Dans Fournisseur de génération, choisissez un fournisseur d'action personnalisée de services de génération, puis indiquez les détails de configuration pour ce fournisseur. Pour obtenir un exemple sur la manière d'ajouter Jenkins comme fournisseur de génération, consultez Didacticiel : Création d'un pipeline à quatre étapes.
- Dans Fournisseur de génération, choisissez AWS CodeBuild.

  Dans Région, choisissez la AWS région dans laquelle se trouve la ressource. Le champ Région indique l'endroit où les AWS ressources sont créées pour ce type d'action et ce type de fournisseur. Ce champ s'affiche uniquement pour les actions dont le fournisseur d'actions est un Service AWS. Le champ Région correspond par défaut à la même AWS région que votre pipeline.

  Dans Nom du projet, choisissez votre projet de génération. Si vous avez déjà créé un projet de construction dans CodeBuild, choisissez-le. Vous pouvez également créer un projet de construction dans CodeBuild puis revenir à cette tâche. Suivez les instructions de la section Création d'un pipeline utilisant CodeBuild dans le Guide de l'utilisateur CodeBuild.

  Sous Spécifications de construction, le fichier CodeBuild buildspec est facultatif et vous pouvez saisir des commandes à la place. Dans Insérer des commandes de construction, entrez les commandes shell pour votre action. Pour plus d'informations sur les considérations relatives à l'utilisation des commandes de génération, consultezRéférence des actions des commandes. Choisissez Utiliser un fichier buildspec si vous souhaitez exécuter des commandes dans d'autres phases ou si vous avez une longue liste de commandes.

  Dans Variables d'environnement, pour ajouter des variables d' CodeBuildenvironnement à votre action de génération, choisissez Ajouter une variable d'environnement. Chaque variable est composée de trois entrées :

  - Dans Nom, entrez le nom ou clé de la variable d'environnement.
  - Dans Valeur, entrez la valeur de la variable d'environnement. Si vous choisissez Parameter pour le type de variable, assurez-vous que cette valeur est le nom d'un paramètre que vous avez déjà stocké dans le magasin de paramètres de AWS Systems Manager.

    > **Note**
    >
    > Nous déconseillons vivement l'utilisation de variables d'environnement pour stocker des valeurs sensibles, en particulier des AWS informations d'identification. Lorsque

vous utilisez la CodeBuild console ou la AWS CLI, les variables d'environnement sont affichées en texte brut. Pour les valeurs sensibles, nous vous recommandons d'utiliser plutôt le type Parameter (Paramètre).

- (Facultatif) Dans Type, entrez le type de variable d'environnement. Les valeurs valides sont Plaintext (Texte brut) ou Parameter (Paramètre). La valeur par défaut est Plaintext (Texte brut).

(Facultatif) Dans Type de construction, choisissez l'une des options suivantes :

- Pour exécuter chaque build en une seule exécution d'action de build, choisissez Single build.
- Pour exécuter plusieurs builds lors de la même exécution d'une action de build, choisissez Batch build.

(Facultatif) Si vous avez choisi d'exécuter des constructions par lots, vous pouvez choisir Combiner tous les artefacts du lot dans un seul emplacement pour placer tous les artefacts de construction dans un seul artefact de sortie.

2. Pour configurer l'étape pour une nouvelle tentative automatique, choisissez Activer la nouvelle tentative automatique en cas d'échec de l'étape. Pour plus d'informations sur la nouvelle tentative automatique, consultezConfigurer une étape pour une nouvelle tentative automatique en cas d'échec.
3. Choisissez Suivant.

Étape 5 : Création d'une phase de test

Cette étape est facultative si vous prévoyez de créer une phase de construction ou de déploiement.

1. Sur la page Étape 5 : Ajouter une phase de test, effectuez l'une des opérations suivantes, puis choisissez Next :
   - Choisissez Ignorer la phase de test si vous prévoyez de créer une phase de construction ou de déploiement.
   - Dans Fournisseur de tests, choisissez le fournisseur d'actions de test et renseignez les champs appropriés.
2. Choisissez Suivant.

Étape 6 : Création d'une phase de déploiement

Cette étape est facultative si vous avez déjà créé une étape de génération.

1. Sur la page Étape 6 : Ajouter une phase de déploiement, effectuez l'une des opérations suivantes, puis choisissez Next :

   - Choisissez Ignorer l'étape de déploiement si vous avez créé une phase de génération ou de test au cours des étapes précédentes.

     > **Note**
     >
     > Cette option n'apparaît pas si vous avez déjà ignoré l'étape de génération ou de test.

   - Dans Deploy provider (Fournisseur de déploiement), choisissez une action personnalisée que vous avez créée pour un fournisseur de déploiement.

     Dans Région, pour les actions interrégionales uniquement, choisissez la AWS région dans laquelle la ressource est créée. Le champ Région indique l'endroit où les AWS ressources sont créées pour ce type d'action et ce type de fournisseur. Ce champ s'affiche uniquement pour les actions dont le fournisseur d'actions est un Service AWS. Le champ Région correspond par défaut à la même AWS région que votre pipeline.

   - Dans Fournisseur de déploiement, les champs sont disponibles pour les fournisseurs par défaut comme suit :

     - CodeDeploy

       Dans Nom de l'application, entrez ou choisissez le nom d'une CodeDeploy application existante. Dans Groupe de déploiement, saisissez le nom d'un groupe de déploiement pour l'application. Choisissez Suivant. Vous pouvez également créer une application, un groupe de déploiement ou les deux dans la console CodeDeploy.

     - AWS Elastic Beanstalk

       Dans Nom de l'application, entrez ou choisissez le nom d'une application Elastic Beanstalk existante. Dans Nom de l'environnement, saisissez un environnement pour l'application. Choisissez Suivant. Vous pouvez également créer une application, un environnement ou les deux dans la console Elastic Beanstalk.

     - AWS OpsWorks Stacks

Dans Stack, saisissez ou choisissez le nom de la pile que vous souhaitez utiliser. Dans Couche, choisissez la couche à laquelle appartiennent vos instances cibles. Dans App, choisissez l'application que vous souhaitez mettre à jour et déployer. Si vous avez besoin de créer une application, choisissez Crée une nouvelle dans AWS OpsWorks.

Pour plus d'informations sur l'ajout d'une application à une pile et à une couche AWS OpsWorks, consultez la section Ajouter des applications dans le guide de AWS OpsWorks l'utilisateur.

Pour un exemple complet d'utilisation d'un pipeline simple CodePipeline comme source de code à exécuter sur des AWS OpsWorks couches, consultez la section Utilisation avec CodePipeline . AWS OpsWorks Stacks

- AWS CloudFormation

  Effectuez l'une des actions suivantes :

  - En mode Action, choisissez Créer ou mettre à jour une pile, entrez le nom de la pile et le nom du fichier modèle, puis choisissez le nom du rôle AWS CloudFormation à assumer. Entrez éventuellement le nom d'un fichier de configuration et choisissez une option de fonctionnalité IAM.
  - En mode Action, choisissez Créer ou remplacer un ensemble de modifications, entrez un nom de pile et un nom d'ensemble de modifications, puis choisissez le nom d'un rôle AWS CloudFormation à assumer. Entrez éventuellement le nom d'un fichier de configuration et choisissez une option de fonctionnalité IAM.

  Pour plus d'informations sur l'intégration de AWS CloudFormation fonctionnalités dans un pipeline CodePipeline, voir Continuous Delivery with CodePipeline dans le guide de AWS CloudFormation l'utilisateur.

- Amazon ECS

  Dans Nom du cluster, entrez ou choisissez le nom d'un cluster Amazon ECS existant. Dans Nom du service, saisissez ou choisissez le nom du service exécuté sur le cluster. Vous pouvez également créer un cluster et un service. Dans Nom du fichier d'image, tapez le nom du fichier de définitions d'image qui décrit le conteneur et l'image du service.

> **Note**
>
> L'action de déploiement Amazon ECS nécessite un `imagedefinitions.json` fichier comme entrée pour l'action de déploiement. Le nom de fichier par défaut est imagedefinitions.json. Si vous choisissez d'utiliser un autre nom de fichier, vous devez l'indiquer lorsque vous créez la phase de déploiement du pipeline. Pour plus d'informations, consultez fichier imagedefinitions.json pour les actions de déploiement standard d'Amazon ECS.

Choisissez Suivant.

> **Note**
>
> Assurez-vous que votre cluster Amazon ECS est configuré avec deux instances ou plus. Les clusters Amazon ECS doivent contenir au moins deux instances afin que l'une soit maintenue en tant qu'instance principale et l'autre utilisée pour les nouveaux déploiements.

Pour un didacticiel concernant le déploiement d'applications basées sur conteneur avec votre pipeline, consultez Didacticiel : Déploiement continu avec CodePipeline.

- Amazon ECS (Blue/Green)

  Entrez l' CodeDeploy application et le groupe de déploiement, la définition de la tâche Amazon ECS et les informations sur AppSpec le fichier, puis choisissez Next.

  > **Note**
  >
  > L'action Amazon ECS (Blue/Green) nécessite un Detail.json fichier image comme artefact d'entrée pour l'action de déploiement. Étant donné que l'action source Amazon ECR crée ce fichier, les pipelines dotés d'une action source Amazon ECR n'ont pas besoin de fournir de fichier. `imageDetail.json` Pour plus d'informations, consultez Detail.json fichier image pour les actions de blue/green déploiement d'Amazon ECS.

Pour un didacticiel sur la création d'un pipeline pour les déploiements bleu-vert vers un cluster Amazon ECS avec CodeDeploy, consultez. Tutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR

- AWS Service Catalog

  Choisissez Enter deployment configuration (Entrer une configuration de déploiement) si vous souhaitez utiliser des champs dans la console pour spécifier votre configuration, ou choisissez Configuration file (Fichier de configuration) si vous disposez d'un fichier de configuration distinct. Entrez les informations sur le produit et la configuration, puis choisissez Suivant.

  Pour un didacticiel sur le déploiement des modifications apportées aux produits dans Service Catalog avec votre pipeline, consultezTutoriel : Création d'un pipeline qui se déploie vers Service Catalog.

- Kit Alexa Skills

  Dans Alexa Skill ID (ID Alexa Skill), entrez l'ID de votre compétence Alexa. Dans Client ID (ID client) et Client secret (Clé secret client), entrez les informations d'identification générées à l'aide d'un profil de sécurité Login with Amazon (LWA). Dans Refresh token (Jeton d'actualisation), entrez le jeton d'actualisation que vous avez généré à l'aide de la commande CLI ASK pour récupérer un jeton d'actualisation. Choisissez Suivant.

  Pour accéder à un didacticiel sur le déploiement des compétences Alexa avec votre pipeline et la génération des informations d'identification LWA, consultez Didacticiel : Création d'un pipeline qui déploie un kit Amazon Alexa Skill.

- Amazon S3

  Dans Bucket (Compartiment), entrez le nom du compartiment S3 que vous souhaitez utiliser. Choisissez Extract file before deploy (Extraire le fichier avant le déploiement) si l'artefact d'entrée de votre étape de déploiement est un fichier ZIP. Si Extract file before deploy (Extraire le fichier avant le déploiement) est sélectionné, vous pouvez, si vous le souhaitez, entrer une valeur pour Deployment path (Chemin du déploiement), où votre fichier ZIP sera décompressé. S'il n'est pas sélectionné, vous devez entrer une valeur dans la clé d'objet S3.

> **Note**
>
> La plupart des artefacts source et des artefacts de sortie d'étape de génération sont compressés. Tous les fournisseurs de sources de pipeline, à l'exception d'Amazon S3, compressent vos fichiers source avant de les fournir comme artefact d'entrée pour l'action suivante.

(Facultatif) Dans Caned ACL, entrez l'ACL prédéfinie à appliquer à l'objet déployé sur Amazon S3.

> **Note**
>
> L'application d'une liste ACL prête à l'emploi remplace toutes les listes ACL existantes appliquées à l'objet.

(Facultatif) Dans Cache control (Contrôle de cache), spécifiez les paramètres de contrôle de cache pour les demandes pour télécharger des objets à partir du compartiment. Pour obtenir la liste des valeurs valides, consultez le champ d'en-tête `Cache-Control` pour les opérations HTTP. Pour entrer plusieurs valeurs dans Cache control (Contrôle de cache), utilisez une virgule entre chaque valeur. Vous pouvez ajouter un espace après chaque virgule (facultatif), comme illustré dans cet exemple.



L'exemple précédent s'affiche dans l'interface de ligne de commande comme suit :

```
"CacheControl": "public, max-age=0, no-transform"
```

Choisissez Suivant.

Pour un didacticiel sur la création d'un pipeline avec un fournisseur d'actions de déploiement Amazon S3, consultezTutoriel : Création d'un pipeline qui utilise Amazon S3 comme fournisseur de déploiement.

2. Pour configurer l'étape pour une nouvelle tentative automatique, choisissez Activer la nouvelle tentative automatique en cas d'échec de l'étape. Pour plus d'informations sur la nouvelle tentative automatique, consultezConfigurer une étape pour une nouvelle tentative automatique en cas d'échec.
3. Pour configurer la phase de restauration automatique, choisissez Configurer la restauration automatique en cas d'échec de l'étape. Pour plus d'informations sur l'annulation automatique, consultezConfigurer une étape pour la restauration automatique.
4. Choisissez Étape suivante.

Étape 7 : Passez en revue le pipeline

- Sur la page Étape 7 : Révision, passez en revue la configuration de votre pipeline, puis choisissez Create pipeline pour créer le pipeline ou Previous pour revenir en arrière et modifier vos choix. Pour quitter l'assistant sans créer de pipeline, choisissez Cancel.

Maintenant que vous avez créé votre pipeline, vous pouvez le voir dans la console. Le pipeline commence à s'exécuter après sa création. Pour plus d'informations, consultez Afficher les pipelines et les détails dans CodePipeline. Pour plus d'informations sur les modifications de votre pipeline, consultez Modifier un pipeline dans CodePipeline.

Création d'un pipeline (interface de ligne de commande)

Pour utiliser le AWS CLI pour créer un pipeline, vous devez créer un fichier JSON pour définir la structure du pipeline, puis exécuter la create-pipeline commande avec le `--cli-input-json` paramètre.

> **Important**
>
> Vous ne pouvez pas utiliser le AWS CLI pour créer un pipeline incluant les actions des partenaires. Vous devez plutôt utiliser la CodePipeline console.

Pour plus d'informations sur la structure du pipeline, consultez CodePipeline référence de la structure du pipeline et create-pipeline dans la référence de l' CodePipeline API.

Pour créer un fichier JSON, utilisez l'exemple de fichier JSON de pipeline, modifiez-le, puis appelez ce fichier lorsque vous exécutez la commande create-pipeline.

Prérequis :

Vous avez besoin de l'ARN du rôle de service que vous avez créé pour CodePipeline dansCommencer avec CodePipeline. Vous utilisez le rôle CodePipeline de service ARN dans le fichier JSON du pipeline lorsque vous exécutez la create-pipeline commande. Pour plus d'informations sur la création d'un rôle de service, consultez la section Création du rôle CodePipeline de service. Contrairement à la console, l'exécution de la create-pipeline commande dans le AWS CLI ne permet pas de créer le rôle CodePipeline de service pour vous. Le rôle de service doit déjà exister.

Vous avez besoin du nom d'un compartiment S3 où les artefacts du pipeline sont stockés. Ce compartiment doit être situé dans la même région que votre pipeline. Vous utilisez le nom du compartiment dans le fichier JSON du pipeline lorsque vous exécutez la commande create-pipeline. Contrairement à la console, l'exécution de la create-pipeline commande dans le AWS CLI ne crée pas de compartiment S3 pour le stockage des artefacts. Le compartiment doit déjà exister.

> **Note**
>
> Vous pouvez aussi utiliser la commande get-pipeline pour obtenir une copie de la structure JSON de ce pipeline, puis modifier cette structure dans un éditeur de texte brut.

Pour créer le fichier JSON

1. Sur un terminal (Linux, macOS ou Unix) ou à l'invite de commande (Windows), créez un nouveau fichier texte dans un répertoire local.
2. (Facultatif) Vous pouvez ajouter une ou plusieurs variables au niveau du pipeline. Vous pouvez référencer cette valeur dans la configuration des CodePipeline actions. Vous pouvez ajouter les noms et les valeurs des variables lorsque vous créez le pipeline, et vous pouvez également choisir d'attribuer des valeurs lorsque vous démarrez le pipeline dans la console.

   > **Note**
   >
   > Bien qu'il soit facultatif d'ajouter des variables au niveau du pipeline, pour un pipeline spécifié avec des variables au niveau du pipeline pour lequel aucune valeur n'est fournie, l'exécution du pipeline échouera.

Une variable au niveau du pipeline est résolue au moment de l'exécution du pipeline. Toutes les variables sont immuables, ce qui signifie qu'elles ne peuvent pas être mises à jour après l'attribution d'une valeur. Les variables au niveau du pipeline avec des valeurs résolues s'afficheront dans l'historique pour chaque exécution.

Vous fournissez des variables au niveau du pipeline à l'aide de l'attribut variables de la structure du pipeline. Dans l'exemple suivant, la valeur de la variable `Variable1` est`Value1`.

```
      "variables": [
           {
               "name": "Timeout",
               "defaultValue": "1000",
               "description": "description"
           }
       ]
```

Ajoutez cette structure au JSON de pipeline ou à l'exemple de JSON à l'étape suivante. Pour plus d'informations sur les variables, y compris les informations sur les espaces de noms, consultezRéférence des variables.

3. Ouvrez le fichier dans un éditeur de texte brut et modifiez les valeurs pour refléter la structure que vous souhaitez créer. Vous devez au moins modifier le nom du pipeline. Vous devez également prendre en compte si vous souhaitez modifier :

   - Le compartiment S3 où les artefacts de ce pipeline sont stockés.
   - L'emplacement source de votre code.
   - Le fournisseur du déploiement.
   - Comment vous souhaitez que votre code soit déployé.
   - Les balises de votre pipeline.

   La structure du modèle de pipeline en deux étapes suivant met en évidence les valeurs que vous devez penser à modifier pour votre pipeline. Votre pipeline contient probablement plus de deux étapes :

```
{
    "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"
    }]
}
```

Cet exemple ajoute le balisage du pipeline en incluant la clé de balise `Project` et la valeur `ProjectA` sur le pipeline. Pour plus d'informations sur le balisage des ressources CodePipeline, consultezBalisage des ressources.

Assurez-vous que le paramètre `PollForSourceChanges` est défini comme suit dans votre fichier JSON :

```
                "PollForSourceChanges": "false",
```

CodePipeline utilise Amazon CloudWatch Events pour détecter les modifications apportées à votre référentiel CodeCommit source et à votre branche ou à votre compartiment source S3. L'étape suivante comporte les instructions de création de ces ressources pour votre pipeline. Vous pouvez définir l'indicateur sur `false` pour désactiver les vérifications périodiques, lesquelles ne sont pas nécessaires lorsque les méthodes de détection des modifications recommandées sont utilisées.

4. Pour créer une action de génération, de test ou de déploiement dans une région différente de celle de votre pipeline, vous devez ajouter les éléments suivants à la structure de votre pipeline. Pour obtenir des instructions, consultez Ajouter une action interrégionale dans CodePipeline.

   - Ajoutez le paramètre `Region` à la structure de pipeline de votre action.
   - Utilisez le `artifactStores` paramètre pour spécifier un compartiment d'artefacts pour chaque AWS région dans laquelle vous avez une action.

5. Une fois que vous êtes satisfait de sa structure, enregistrez votre fichier en lui attribuant un nom tel que **`pipeline.json`**.

Pour créer un pipeline

1. Exécutez la commande create-pipeline et utilisez le paramètre `--cli-input-json` pour spécifier le fichier JSON que vous venez de créer.

   Pour créer un pipeline nommé à l'*MySecondPipeline*aide d'un fichier JSON nommé pipeline.json qui inclut le nom « *MySecondPipeline* » comme valeur `name` dans le JSON, votre commande devrait ressembler à ce qui suit :

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

   > **Important**
   >
   > N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

   Cette commande renvoie la structure de l'ensemble du pipeline que vous avez créé.
2. Pour afficher le pipeline, ouvrez la CodePipeline console et choisissez-le dans la liste des pipelines, ou utilisez la get-pipeline-state commande. Pour plus d'informations, consultez Afficher les pipelines et les détails dans CodePipeline.
3. Si vous utilisez l'interface de ligne de commande pour créer un pipeline, vous devez créer manuellement les ressources de détection des modifications recommandées pour votre pipeline :

   - Pour un pipeline doté d'un CodeCommit référentiel, vous devez créer manuellement la règle CloudWatch Events, comme décrit dansCréation d'une EventBridge règle pour une CodeCommit source (CLI).
   - Pour un pipeline avec une source Amazon S3, vous devez créer manuellement la règle et le suivi des CloudWatch AWS CloudTrail événements, comme décrit dansConnexion à la source Amazon S3 : actions qui utilisent EventBridge et AWS CloudTrail.

Création d'un pipeline à partir de modèles statiques

Vous pouvez créer un pipeline dans la console qui utilise un modèle pour configurer un pipeline avec le code source et les propriétés que vous spécifiez. Vous devez fournir l'emplacement du fichier source et des informations sur les fournisseurs de source que vous utiliserez pour vos actions. Vous

pouvez spécifier une action source pour Amazon ECR ou pour tout référentiel tiers pris en charge par AWS CodeConnections, tel que GitHub.

Le modèle créera une pile CloudFormation pour votre pipeline qui inclut les ressources suivantes :

- Un pipeline est créé avec le type de pipeline V2. Dans Type de pipeline, choisissez l'une des options suivantes. Les types de pipelines diffèrent en termes de caractéristiques et de prix. Pour plus d'informations, consultez Types de canalisations.
- Un rôle de service est créé pour votre pipeline et référencé dans le modèle.
- Un magasin d'artefacts est créé à l'aide du magasin d'artefacts par défaut, tel que le compartiment d'artefacts S3 désigné par défaut, pour votre pipeline dans le répertoire que Région AWS vous avez sélectionné pour votre pipeline.

Pour consulter la collection de modèles de démarrage open source utilisés pour l'assistant de création de modèles statiques, consultez le référentiel à l'adresse https://github.com/aws/codepipeline-starter-templates.

Lorsque vous utilisez des modèles statiques pour créer un pipeline, la structure du pipeline est configurée dans chaque modèle en fonction des besoins du cas d'utilisation. Par exemple, le modèle de déploiement vers CloudFormation est utilisé comme exemple dans cette procédure. Le modèle génère un pipeline nommé DeployToCloudFormationServiceselon la structure suivante :

- Une phase de construction qui contient une action source avec une configuration que vous spécifiez dans l'assistant.
- Une phase de déploiement avec une action de déploiement et une pile de ressources associée CloudFormation.

Lorsque vous utilisez un modèle statique pour créer un pipeline, il CodePipeline crée les noms des étapes (source, build, staging). Ces noms ne peuvent pas être modifiés. Vous pouvez utiliser des noms plus spécifiques (par exemple, BuildToGamma ou DeployToProd) pour les étapes que vous ajouterez ultérieurement.

Étape 1 : accéder à la console

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Sur la page Welcome (Bienvenue), choisissez Créer un pipeline.

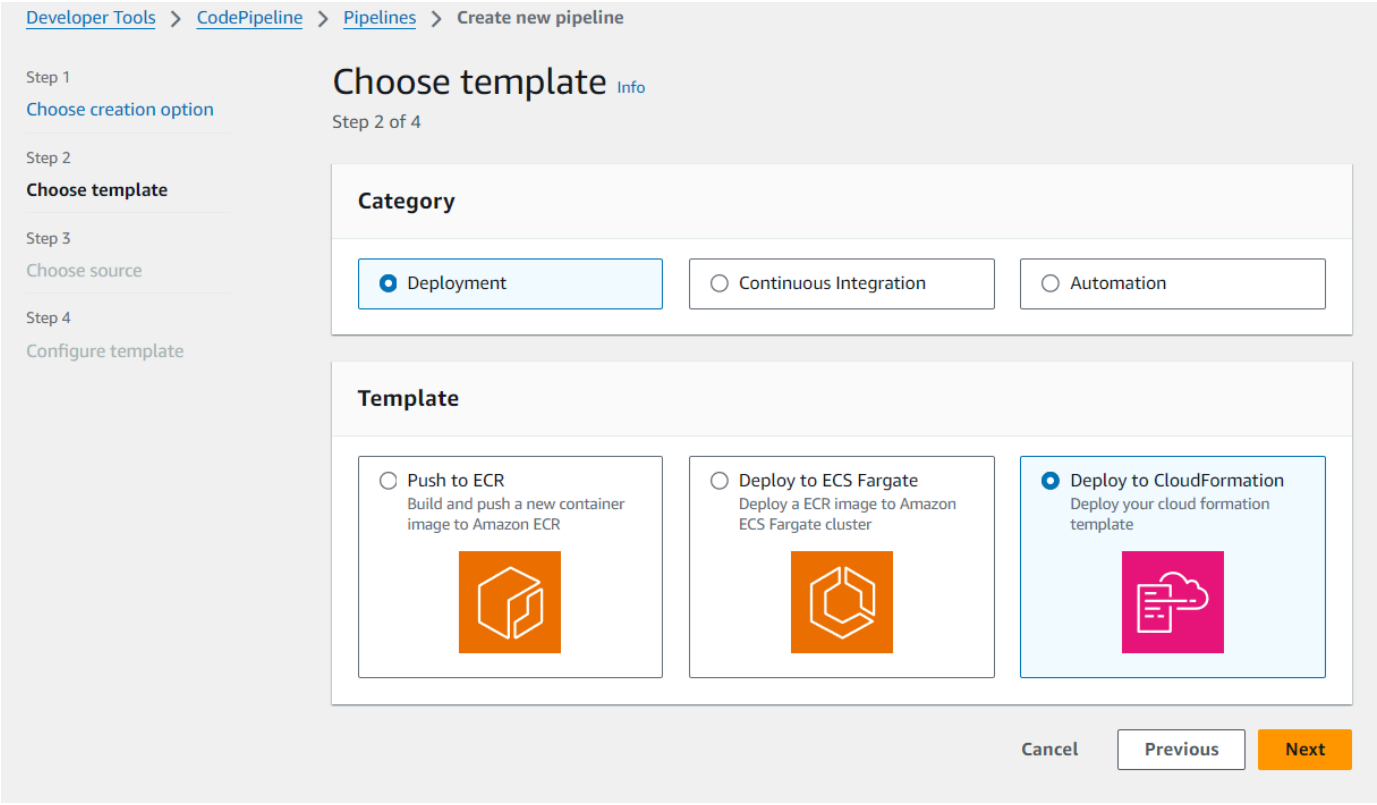
Si c'est la première fois que vous l'utilisez CodePipeline, choisissez Get Started.

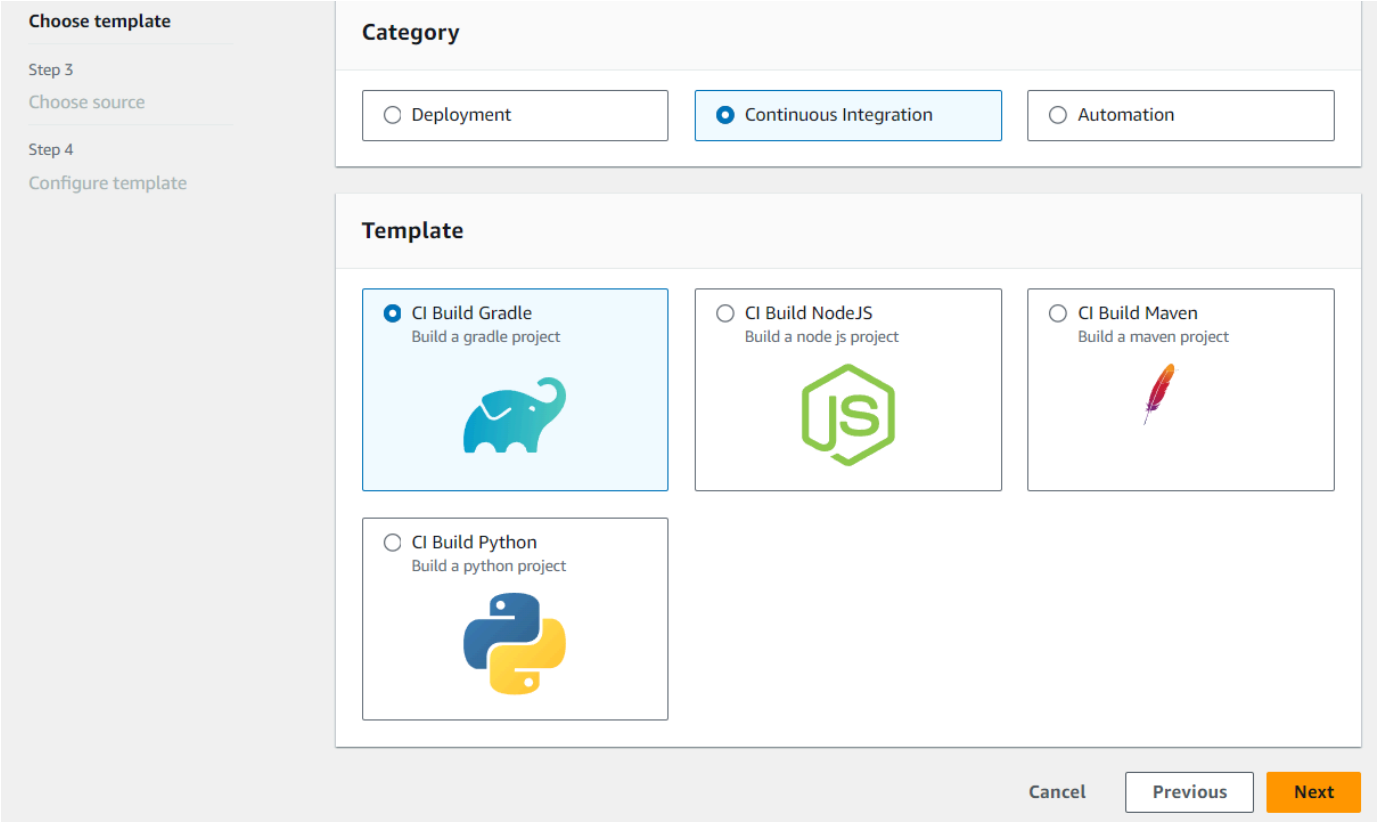
Étape 2 : Choisissez le modèle

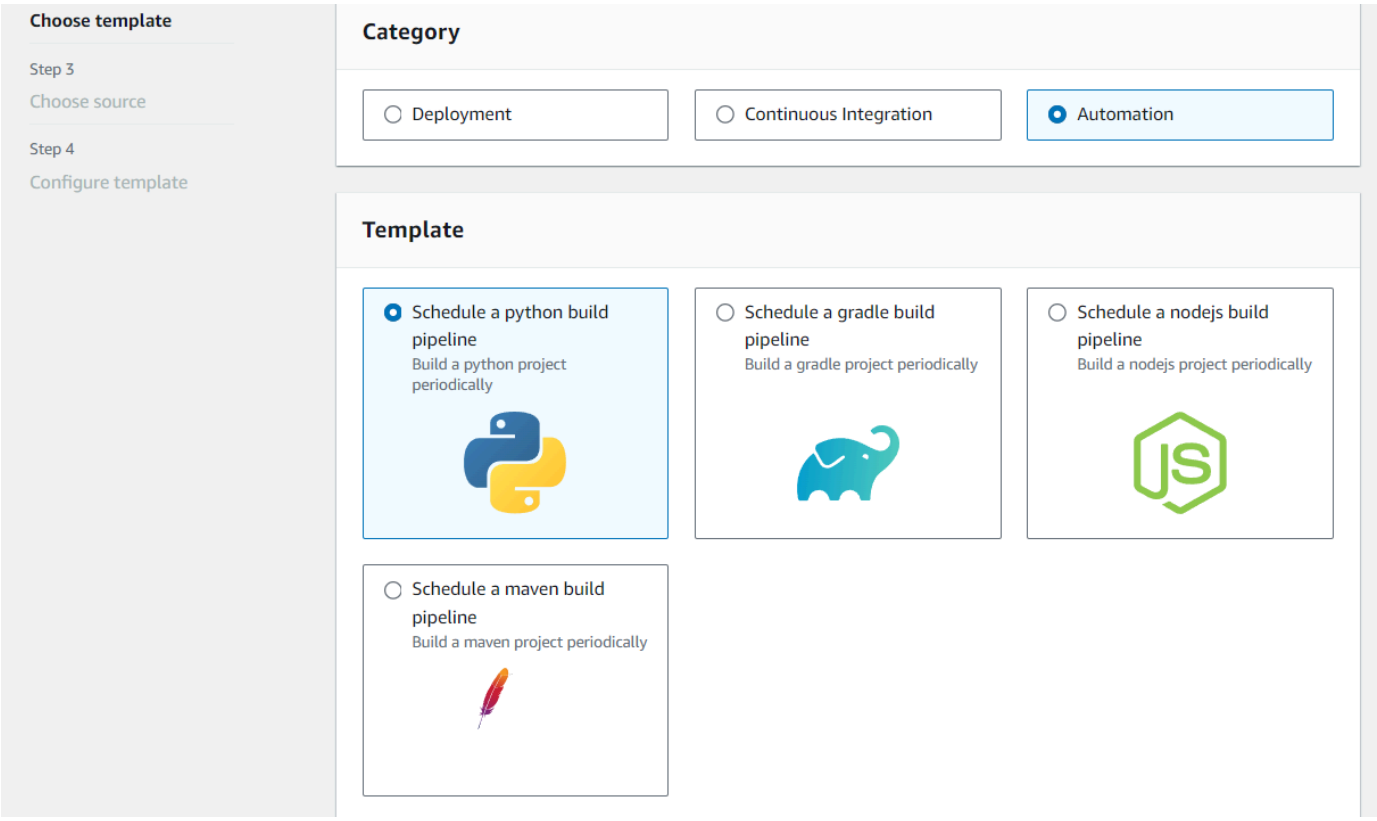
Choisissez un modèle pour créer un pipeline avec une phase de déploiement, une automatisation ou un pipeline CI.

1. Sur la page Étape 1 : Choisir une option de création, effectuez l'une des opérations suivantes, puis choisissez Suivant :

   - Choisissez Déploiement si vous envisagez de créer une étape de déploiement. Consultez les options des modèles déployés sur ECR ou CloudFormation. Pour cet exemple, choisissez Deployment, puis choisissez de déployer vers CloudFormation.
   - Choisissez Intégration continue si vous envisagez de créer un pipeline CI. Consultez les options relatives aux pipelines CI, comme la création selon Gradle.
   - Choisissez Automatisation si vous envisagez de créer un pipeline automatisé. Consultez les options d'automatisation telles que la planification d'une compilation Python.

2. 


Choose template
Step 3
Choose source
Step 4
Configure template
Category
Deployment
Continuous Integration
Automation
Template
CI Build Gradle
Build a gradle project
CI Build NodeJS
Build a node js project
CI Build Maven
Build a maven project
CI Build Python
Build a python project
Cancel
Previous
Next


Choose template
Step 3
Choose source
Step 4
Configure template
Category
Deployment
Continuous Integration
Automation
Template
Schedule a python build pipeline
Build a python project periodically
Schedule a gradle build pipeline
Build a gradle project periodically
Schedule a nodejs build pipeline
Build a nodejs project periodically
Schedule a maven build pipeline
Build a maven project periodically

Étape 3 : Choisissez la source

- Sur la page Étape 3 : Choisir une source, dans Fournisseur de source, choisissez le fournisseur du référentiel dans lequel votre code source est stocké, spécifiez les options requises, puis choisissez Étape suivante.

  - Pour Bitbucket Cloud GitHub (via une GitHub application), GitHub Enterprise Server ou GitLab autogéré : GitLab.com

    1. Sous Connexion, choisissez une connexion existante ou créez-en une nouvelle. Pour créer ou gérer une connexion pour votre action GitHub source, consultezGitHub connexions.
    2. Choisissez le référentiel que vous souhaitez utiliser comme emplacement source pour votre pipeline.

       Choisissez d'ajouter un déclencheur ou de filtrer les types de déclencheurs pour démarrer votre pipeline. Pour plus d'informations sur l'utilisation des déclencheurs, consultezAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ». Pour plus d'informations sur le filtrage à l'aide de modèles globulaires, consultezUtilisation de modèles globulaires dans la syntaxe.
    3. Dans Format d'artefact de sortie, choisissez le format de vos artefacts.
       - Pour stocker les artefacts de sortie de l' GitHub action à l'aide de la méthode par défaut, choisissez CodePipelinepar défaut. L'action accède aux fichiers depuis le GitHub référentiel et stocke les artefacts dans un fichier ZIP dans le magasin d'artefacts du pipeline.
       - Pour stocker un fichier JSON contenant une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git, choisissez Full clone (Clone complet). Cette option ne peut être utilisée que par les actions CodeBuild en aval.

         Si vous choisissez cette option, vous devrez mettre à jour les autorisations associées à votre rôle de service de CodeBuild projet, comme indiqué dansRésolution des problèmes CodePipeline. Pour consulter un didacticiel expliquant comment utiliser l'option de clonage complet, voirTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

  - Pour Amazon ECR :
    - Dans Nom du référentiel, choisissez le nom de votre référentiel Amazon ECR.
    - Dans Balise d'image, spécifiez le nom et la version de l'image, s'ils sont différents de LATEST.

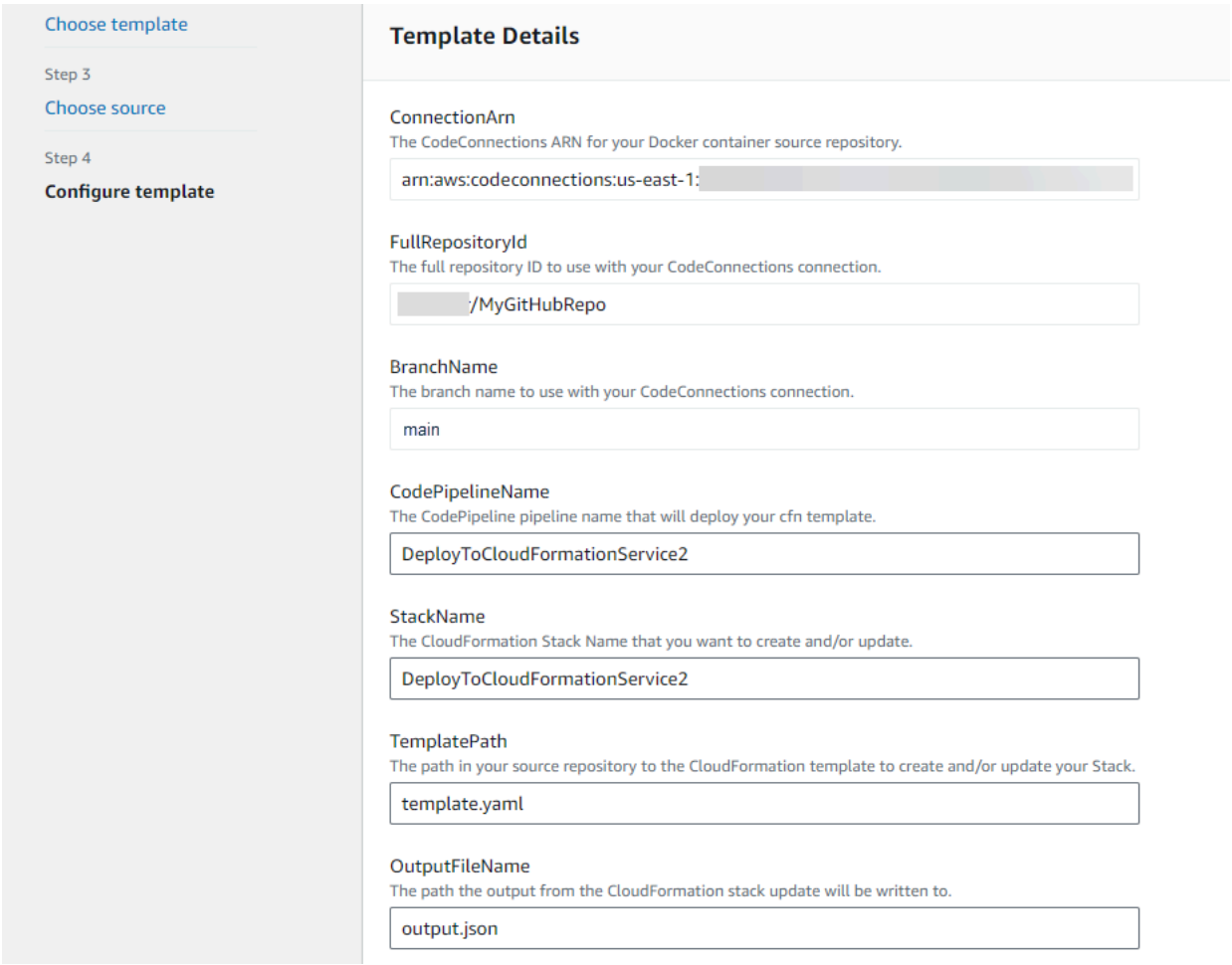
- Dans Artefacts de sortie, choisissez l'artefact de sortie par défaut, par exemple MyApp, qui contient le nom de l'image et les informations d'URI du référentiel que vous souhaitez utiliser à l'étape suivante.

Lorsque vous incluez un stage source Amazon ECR dans votre pipeline, l'action source génère un `imageDetail.json` fichier en tant qu'artefact de sortie lorsque vous validez une modification. Pour de plus amples informations concernant le fichier `imageDetail.json`, veuillez consulter Detail.json fichier image pour les actions de blue/green déploiement d'Amazon ECS.

> **Note**
>
> L'objet et le type de fichier doivent être compatibles avec le système de déploiement que vous prévoyez d'utiliser (par exemple, Elastic Beanstalk ou). CodeDeploy Les types de fichier .zip, .tar et .tgz peuvent être pris en charge. Pour plus d'informations sur les types de conteneurs pris en charge pour Elastic Beanstalk, consultez Personnalisation et configuration des environnements Elastic Beanstalk et des plateformes prises en charge. Pour plus d'informations sur le déploiement de révisions avec CodeDeploy, consultez les sections Chargement de la révision de votre application et Préparation d'une révision.

Étape 4 : Configuration du modèle

Pour cet exemple, le déploiement vers CloudFormation a été sélectionné. Au cours de cette étape, ajoutez la configuration de votre modèle.



1. À l'étape 4 : Configurer le modèle, dans Nom de la pile, entrez le nom de votre pipeline.
2. Modifiez la politique IAM de l'espace réservé pour les autorisations qui s'appliquent à votre modèle.
3. Choisissez Créer un pipeline à partir d'un modèle
4. Un message s'affiche indiquant que les ressources de votre pipeline sont en cours de création.

Étape 5 : Afficher le pipeline

- Maintenant que vous avez créé votre pipeline, vous pouvez le visualiser dans la CodePipeline console et voir le stack in CloudFormation. Le pipeline commence à s'exécuter après sa création. Pour plus d'informations, consultez Afficher les pipelines et les détails dans CodePipeline. Pour plus d'informations sur les modifications de votre pipeline, consultez Modifier un pipeline dans CodePipeline.

Modifier un pipeline dans CodePipeline

Un pipeline décrit le processus de publication que vous AWS CodePipeline souhaitez suivre, y compris les étapes et les actions à effectuer. Vous pouvez modifier un pipeline pour ajouter ou supprimer ces éléments. Toutefois, lorsque vous modifiez un pipeline, les valeurs telles que le nom ou les métadonnées du pipeline ne peuvent pas être modifiées.

Vous pouvez modifier le type de pipeline, les variables et les déclencheurs à l'aide de la page d'édition du pipeline. Vous pouvez également ajouter ou modifier des étapes et des actions dans votre pipeline.

Contrairement à la création d'un pipeline, la modification d'un pipeline ne déclenche pas une nouvelle exécution de la plus récente révision dans le pipeline. Si vous souhaitez exécuter la plus récente révision dans un pipeline que vous venez de modifier, vous devez la relancer manuellement. Sinon, le pipeline modifié s'exécute la prochaine fois que vous apportez une modification à un emplacement de la source configuré dans l'étape source. Pour plus d'informations, veuillez consulter Lancement manuel d'un pipeline.

Vous pouvez ajouter à votre pipeline des actions situées dans une AWS région différente de votre pipeline. Lorsqu'un Service AWS est le fournisseur d'une action et que ce type d'action/ce type de fournisseur se trouve dans une AWS région différente de celle de votre pipeline, il s'agit d'une action interrégionale. Pour de plus amples informations sur les actions inter-régions, veuillez consulter Ajouter une action interrégionale dans CodePipeline.

CodePipeline utilise des méthodes de détection des modifications pour démarrer votre pipeline lorsqu'une modification du code source est poussée. Ces méthodes de détection sont basées sur le type de source :

- CodePipeline utilise Amazon CloudWatch Events pour détecter les modifications apportées à votre référentiel CodeCommit source ou à votre compartiment source Amazon S3.

> **Note**
>
> Les ressources de détection des modifications sont créées automatiquement lorsque vous utilisez la console. Lorsque vous utilisez la console pour créer ou modifier un pipeline, les ressources supplémentaires sont créées pour vous. Si vous utilisez le AWS CLI pour créer le pipeline, vous devez créer vous-même les ressources supplémentaires. Pour plus d'informations sur la création ou la mise à jour d'un CodeCommit pipeline, consultezCréation

d'une EventBridge règle pour une CodeCommit source (CLI). Pour plus d'informations sur l'utilisation de la CLI pour créer ou mettre à jour un pipeline Amazon S3, consultezCréation d'une EventBridge règle pour une source Amazon S3 (CLI).

Rubriques

- Modification d'un pipeline (console)
- Modification d'un pipeline (AWS CLI)

Modification d'un pipeline (console)

Vous pouvez utiliser la CodePipeline console pour ajouter, modifier ou supprimer des étapes dans un pipeline et pour ajouter, modifier ou supprimer des actions dans une étape.

Lorsque vous mettez à jour un pipeline, vous CodePipeline effectuez correctement toutes les actions en cours, puis les étapes et les exécutions du pipeline au cours desquelles les actions en cours ont été effectuées échouent. Lorsqu'un pipeline est mis à jour, vous devez le réexécuter. Pour plus d'informations sur l'exécution d'un pipeline, consultezLancement manuel d'un pipeline.

Pour modifier un pipeline

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés.
2. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier. Une vue détaillée du pipeline s'affiche alors, laquelle indique notamment l'état de chaque action, dans chaque étape du pipeline.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Pour modifier le type de pipeline, choisissez Modifier sur la carte Modifier : propriétés du pipeline. Choisissez l'une des options suivantes, puis cliquez sur OK.

   - Les pipelines de type V1 ont une structure JSON qui contient des paramètres standard au niveau du pipeline, de l'étape et de l'action.
   - Les pipelines de type V2 ont la même structure que les pipelines de type V1, avec une prise en charge de paramètres supplémentaires, tels que des déclencheurs et des variables au niveau du pipeline.

Les types de pipelines diffèrent en termes de caractéristiques et de prix. Pour de plus amples informations, veuillez consulter Types de canalisations.

5. Pour modifier les variables du pipeline, choisissez Modifier les variables sur la carte Modifier : Variables. Ajoutez ou modifiez des variables pour le niveau du pipeline, puis choisissez OK.

   Pour plus d'informations sur les variables au niveau du pipeline, consultezRéférence des variables. Pour un didacticiel avec une variable au niveau du pipeline transmise au moment de l'exécution du pipeline, voir. Tutoriel : Utiliser des variables au niveau du pipeline

   > **Note**
   >
   > Bien qu'il soit facultatif d'ajouter des variables au niveau du pipeline, pour un pipeline spécifié avec des variables au niveau du pipeline pour lequel aucune valeur n'est fournie, l'exécution du pipeline échouera.

6. Pour modifier les déclencheurs du pipeline, choisissez Modifier les déclencheurs sur la carte Modifier : déclencheurs. Ajoutez ou modifiez des déclencheurs, puis choisissez OK.

   Pour plus d'informations sur l'ajout de déclencheurs, consultez les étapes de création d'une connexion à Bitbucket Cloud GitHub (via GitHub App), GitHub Enterprise Server, GitLab .com ou GitLab autogérée, telles que. GitHub connexions

7. Pour modifier les étapes et les actions sur la page d'édition, effectuez l'une des opérations suivantes :

   - Pour modifier une étape, choisissez Modifier une étape. Vous pouvez ajouter des actions en série et en parallèle avec des actions existants :

     Vous pouvez également modifier des actions dans cette vue, en sélectionnant l'icône de modification de ces actions. Pour supprimer une action, choisissez l'icône de suppression de cette action.

   - Pour modifier une action, choisissez l'icône de modification de cette action, puis choisissez Action de modification, modifiez les valeurs. Les éléments marqués d'un astérisque (*) sont obligatoires.

     - Pour le nom du CodeCommit référentiel et la branche, un message s'affiche indiquant la règle Amazon CloudWatch Events à créer pour ce pipeline. Si vous supprimez la

CodeCommit source, un message s'affiche indiquant la règle Amazon CloudWatch Events à supprimer.

- Pour un compartiment source Amazon S3, un message s'affiche indiquant la règle Amazon CloudWatch Events AWS CloudTrail et le journal à créer pour ce pipeline. Si vous supprimez la source Amazon S3, un message s'affiche indiquant la règle Amazon CloudWatch Events et AWS CloudTrail le journal à supprimer. Si le AWS CloudTrail sentier est utilisé par d'autres pipelines, il n'est pas supprimé et l'événement de données est supprimé.

- Pour ajouter une étape, choisissez + Ajouter une étape à l'emplacement du pipeline où vous souhaitez ajouter une étape. Donnez un nom à l'étape, puis ajoutez-y au moins une action. Les éléments marqués d'un astérisque (*) sont obligatoires.
- Pour supprimer une étape, choisissez l'icône de suppression de cette étape. L'étape ainsi que la totalité de ses actions sont supprimées.
- Pour configurer une étape afin qu'elle soit annulée automatiquement en cas d'échec, choisissez Modifier l'étape, puis cochez la case Configurer la restauration automatique en cas d'échec de l'étape.

Par exemple, si vous souhaitez ajouter une action en série à une étape dans un pipeline :

1. Dans l'étape où vous souhaitez ajouter votre action, choisissez Modifier une étape, puis choisissez + Ajouter un groupe d'actions.
2. Dans Modifier une action, dans Nom de l'action, saisissez le nom de votre action. La liste Fournisseur d'actions affiche les options de fournisseur par catégorie. Recherchez la catégorie (par exemple, Déployer). Dans la catégorie, choisissez le fournisseur (par exemple, AWS CodeDeploy). Dans Région, choisissez la AWS région dans laquelle la ressource est créée ou dans laquelle vous prévoyez de la créer. Le champ Région indique où les AWS ressources sont créées pour ce type d'action et ce type de fournisseur. Ce champ s'affiche uniquement pour les actions dont le fournisseur d'actions est un Service AWS. Le champ Région correspond par défaut à la même AWS région que votre pipeline.

   Pour obtenir des exemples d'action d'ajout et d'utilisation des champs par défaut pour chaque fournisseur, consultez la section Création d'un pipeline personnalisé (console).

Pour l'ajouter CodeBuild en tant qu'action de génération ou action de test à une étape, voir Utiliser CodePipeline avec CodeBuild pour tester le code et exécuter des builds dans le guide de CodeBuild l'utilisateur.

> **Note**
>
> Certains fournisseurs d'actions, par exemple GitHub, vous demandent de vous connecter au site Web du fournisseur avant de terminer la configuration de l'action. Lorsque vous vous connectez au site web d'un fournisseur, veillez à utiliser les informations d'identification appropriées pour ce site. N'utilisez pas vos AWS informations d'identification.

3. Lorsque vous avez terminé de configurer votre action, choisissez Enregistrer.

> **Note**
>
> Vous ne pouvez pas renommer une étape dans la vue de la console. Vous pouvez ajouter une étape en lui donnant un nouveau nom; puis supprimer l'ancienne. Assurez-vous que vous avez ajouté toutes les actions que vous souhaitez inclure dans cette étape avant de supprimer l'ancienne.

8. Lorsque vous avez terminé de modifier votre pipeline, cliquez sur Enregistrer les modifications du pipeline pour revenir à la page récapitulative.

> **Important**
>
> Une fois que vous avez enregistré vos modifications, vous ne pouvez pas les annuler. Vous devez modifier à nouveau le pipeline. Si une révision est en cours d'exécution dans votre pipeline lorsque vous enregistrez vos modifications, ce dernier ne finalise pas l'exécution. Si vous souhaitez exécuter une validation ou une modification spécifique dans le pipeline modifié, vous devez l'exécuter manuellement dans l'ensemble du pipeline. Dans le cas contraire, la prochaine validation ou modification s'exécute automatiquement dans le pipeline.

9. Pour tester votre action, choisissez Release change pour traiter cette validation via le pipeline et valider une modification de la source spécifiée dans l'étape source du pipeline. Ou suivez

les étapes décrites Lancement manuel d'un pipeline pour utiliser le AWS CLI pour publier manuellement une modification.

Modification d'un pipeline (AWS CLI)

Vous pouvez utiliser la commande update-pipeline pour modifier un pipeline.

Lorsque vous mettez à jour un pipeline, vous CodePipeline effectuez correctement toutes les actions en cours, puis les étapes et les exécutions du pipeline au cours desquelles les actions en cours ont été effectuées échouent. Lorsqu'un pipeline est mis à jour, vous devez le réexécuter. Pour plus d'informations sur l'exécution d'un pipeline, consultezLancement manuel d'un pipeline.

> **Important**
>
> Bien que vous puissiez utiliser le AWS CLI pour modifier des pipelines qui incluent des actions partenaires, vous ne devez pas modifier manuellement le code JSON d'une action partenaire. Si vous le faites, l'action partenaire échouera probablement une fois le pipeline mis à jour.

Pour modifier un pipeline

1. Ouvrez une session de terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et exécutez la get-pipeline commande pour copier la structure du pipeline dans un fichier JSON. Par exemple, pour un pipeline nommé **`MyFirstPipeline`**, saisissez la commande suivante :

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Cette commande ne renvoie rien, mais le fichier que vous avez créé doit apparaître dans le répertoire où vous avez exécuté la commande.

2. Ouvrez le fichier JSON dans un éditeur de texte brut et modifiez la structure du fichier pour refléter les modifications que vous souhaitez apporter au pipeline. Par exemple, vous pouvez ajouter ou supprimer des étapes, ou ajouter une autre action à une étape.

   L'exemple suivant montre comment ajouter une autre étape de déploiement dans le fichier pipeline.json. Cette étape s'exécute après la première étape de déploiement nomméeStaging.

> **Note**
>
> Il s'agit juste d'une partie du fichier, et non pas de la structure entière. Pour de plus amples informations, veuillez consulter CodePipeline référence de la structure du pipeline.

```
,
        {
            "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
                }
            ]
        }
    ]
}
```

Pour plus d'informations sur l'ajout d'une action d'approbation à un pipeline à l'aide du CLI, consultez Ajoutez une action d'approbation manuelle à un pipeline dans CodePipeline.

Assurez-vous que le paramètre `PollForSourceChanges` est défini comme suit dans votre fichier JSON :

```
                "PollForSourceChanges": "false",
```

CodePipeline utilise Amazon CloudWatch Events pour détecter les modifications apportées à votre référentiel CodeCommit source et à votre branche ou à votre compartiment source Amazon S3. L'étape suivante comporte les instructions de création manuelle de ces ressources. Vous pouvez définir l'indicateur sur `false`pour désactiver les vérifications périodiques, lesquelles ne sont pas nécessaires lorsque vous utilisez les méthodes de détection des modifications recommandées.

3. Pour ajouter une action de génération, de test ou de déploiement dans une région différente de celle de votre pipeline, vous devez ajouter les éléments suivants à la structure de pipeline. Pour obtenir des instructions complètes, veuillez consulter Ajouter une action interrégionale dans CodePipeline.

   - Ajoutez le paramètre `Region` à la structure de pipeline de votre action.

- Utilisez le paramètre `artifactStores` afin de spécifier un compartiment d'artefacts pour chaque région dans laquelle vous avez une action.

4. Si vous utilisez la structure de pipeline extraite à l'aide de la commande get-pipeline, vous devez modifier la structure dans le fichier JSON. Vous devez supprimer les lignes `metadata` du fichier afin que la commande update-pipeline puisse l'utiliser. Supprimez la section de la structure de pipeline dans le fichier JSON (les lignes `"metadata": { }` et les champs `"created"`, `"pipelineARN"` et `"updated"`).

   Par exemple, supprimez les lignes suivantes de la structure :

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

   Enregistrez le fichier.

5. Si vous utilisez l'interface de ligne de commande pour modifier un pipeline, vous devez gérer manuellement les ressources de détection des modifications recommandées pour votre pipeline :

   - Pour un CodeCommit référentiel, vous devez créer la règle CloudWatch Events, comme décrit dansCréation d'une EventBridge règle pour une CodeCommit source (CLI).
   - Pour une source Amazon S3, vous devez créer la règle et le suivi AWS CloudTrail des CloudWatch événements, comme décrit dansConnexion à la source Amazon S3 : actions qui utilisent EventBridge et AWS CloudTrail.

6. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline :

   > **Important**
   >
   > N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

   Cette commande affiche toute la structure du pipeline mise à jour.

> **Note**
>
> La commande update-pipeline interrompt le pipeline. Si une révision est exécutée dans le pipeline lorsque vous exécutez la commande update-pipeline celle-ci est interrompue. Vous devez lancer manuellement le pipeline pour exécuter cette révision dans le pipeline mis à jour.

7. Ouvrez la CodePipeline console et choisissez le pipeline que vous venez de modifier.

   Le pipeline affiche vos modifications. La prochaine fois que vous apporterez une modification à l'emplacement source, le pipeline exécute cette modification via sa structure révisée.

8. Pour exécuter manuellement la dernière mise à jour dans la structure révisée du pipeline, exécutez la commande start-pipeline-execution. Pour de plus amples informations, veuillez consulter Lancement manuel d'un pipeline.

Pour plus d'informations sur la structure d'un pipeline et les valeurs attendues, consultez CodePipeline référence de la structure du pipeline une référence AWS CodePipeline d'API.

Afficher les pipelines et les détails dans CodePipeline

Vous pouvez utiliser la AWS CodePipeline console ou le AWS CLI pour afficher les détails des pipelines associés à votre AWS compte.

Rubriques

- Afficher les pipelines (console)
- Afficher les détails des actions dans un pipeline (console)
- Afficher l'ARN du pipeline et l'ARN du rôle de service (console)
- Affichage des détails et de l'historique d'un pipeline (interface de ligne de commande)
- Afficher les résultats des règles pour les conditions d'étape dans l'historique d'exécution

Afficher les pipelines (console)

Vous pouvez afficher l'état, les transitions et les mises à jour d'artefacts d'un pipeline.

> **Note**
>
> La vue détaillée d'un pipeline cesse automatiquement de s'actualiser dans votre navigateur au bout d'une heure. Pour afficher des informations à jour, actualisez la page.

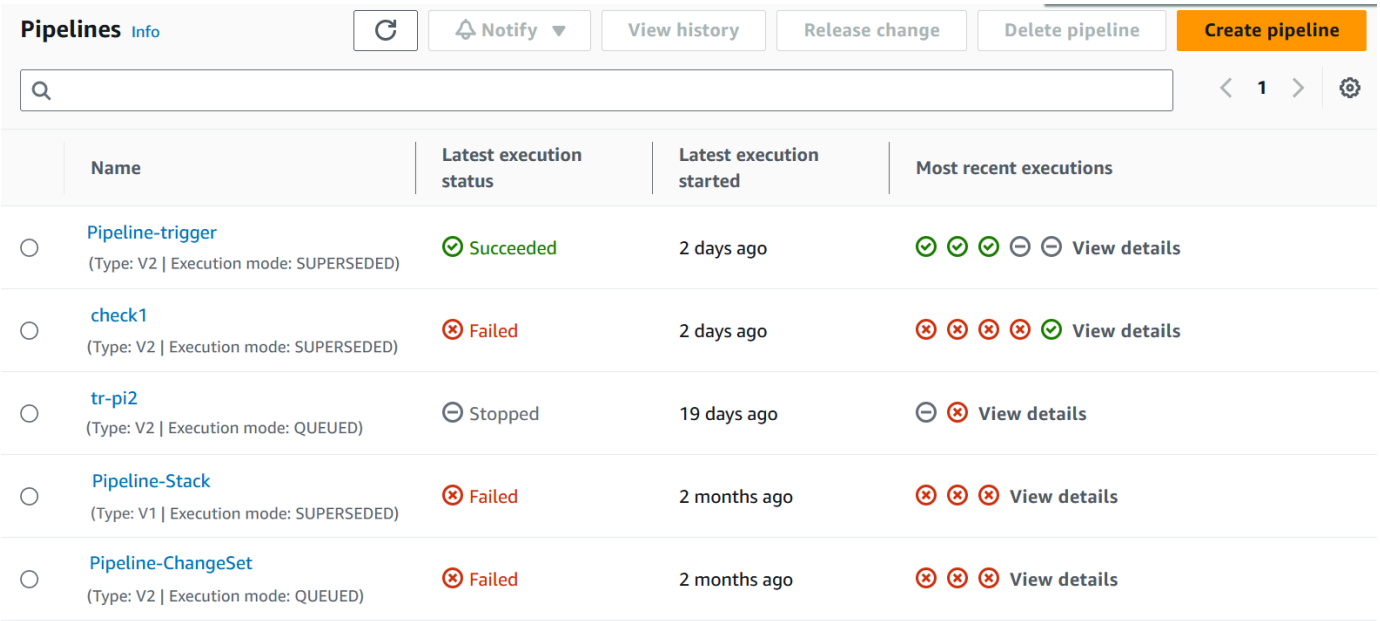
Pour afficher les pipelines

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.

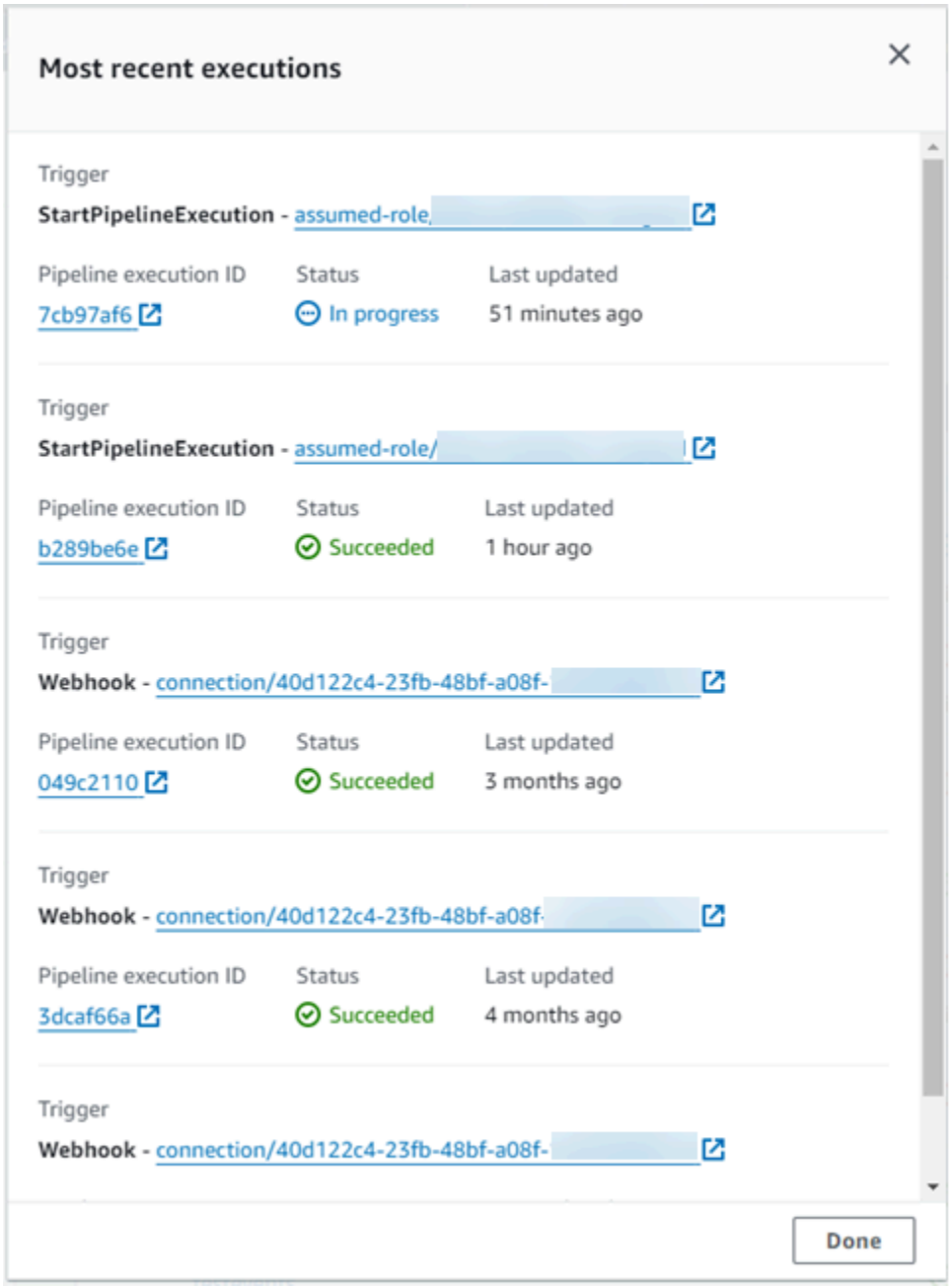
   La page Pipelines s'affiche. La liste de tous vos pipelines pour cette région s'affiche.

   Le nom, le type, le statut, la version, la date de création et la date de dernière modification de tous les pipelines associés à votre AWS compte sont affichés, ainsi que l'heure d'exécution la plus récente.

2. Le statut des cinq dernières exécutions est indiqué.



   Choisissez Afficher les détails à côté d'une ligne spécifique pour afficher une boîte de dialogue de détails répertoriant les exécutions les plus récentes.

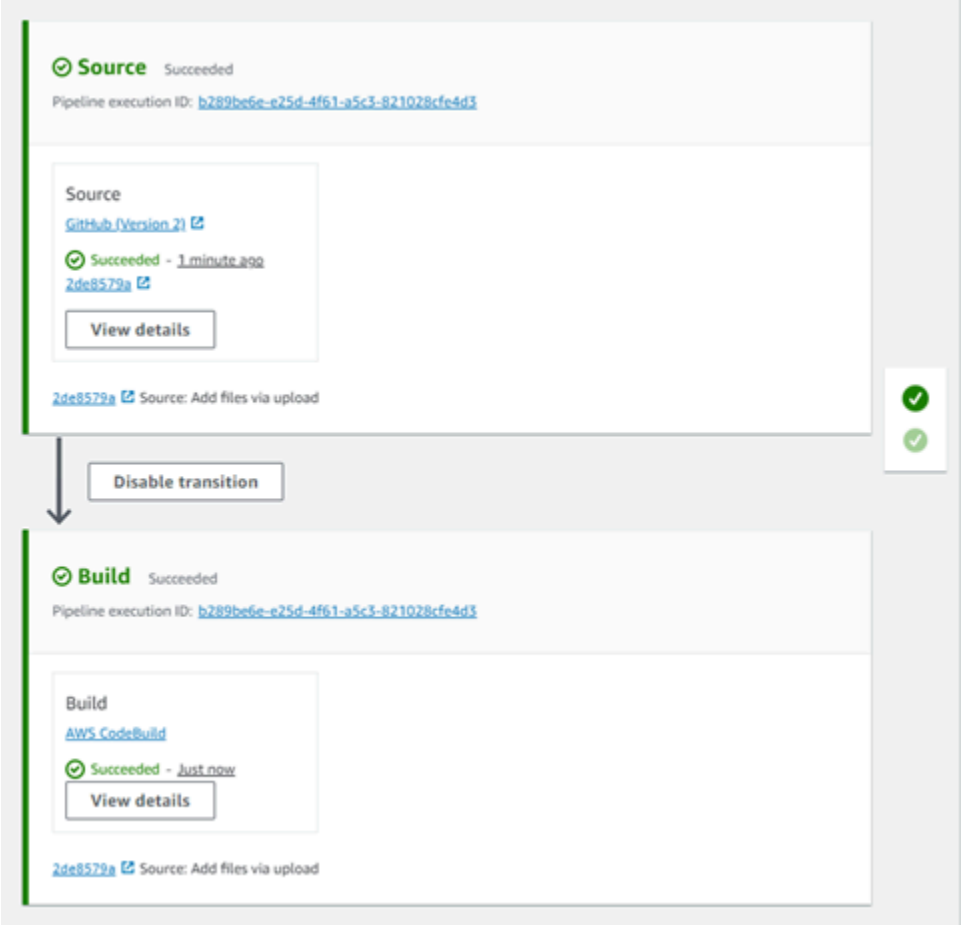


Pour afficher des détails des dernières exécutions de pipeline, sélectionnez Afficher l'historique. Pour les exécutions passées, vous pouvez consulter les détails des révisions associés aux artefacts source, tels que l'exécution IDs, le statut, les heures de début et de fin, la durée, la validation IDs et les messages.

> **Note**
>
> Pour un pipeline en mode d'exécution PARALLÈLE, la vue principale du pipeline ne montre pas la structure du pipeline ni les exécutions en cours. Pour un pipeline en mode

d'exécution PARALLÈLE, vous pouvez accéder à la structure du pipeline en choisissant l'ID de l'exécution que vous souhaitez consulter sur la page d'historique des exécutions. Choisissez History dans le menu de navigation de gauche, choisissez l'ID d'exécution pour l'exécution parallèle, puis visualisez le pipeline dans l'onglet Visualization.

3. Pour voir les détails d'un seul pipeline, choisissez le pipeline dans Name. Une vue détaillée du pipeline, qui inclut l'état de chaque action dans chaque étape et l'état des transitions, s'affiche.



La vue graphique affiche les informations suivantes pour chaque étape :

- Nom de l'étape.
- Chaque action configurée pour l'étape.
- L'état de transition entre les étapes (activé ou désactivé), indiqué par l'état de la flèche entre les étapes. Une transition activée est indiquée par une flèche avec un bouton Désactiver la transition en regard de celle-ci. Une transition désactivée est indiquée par une flèche barrée avec un bouton Activer la transition en regard de celle-ci.
- Barre de couleurs qui indique l'état de l'étape :

- Gris : Pas encore d'exécution
- Blue : En cours
- Vert : Réussite
- Rouge : Échec

La vue graphique affiche également les informations suivantes sur les actions de chaque étape :

- Le nom de l'action.
- Le fournisseur de l'action, tel que CodeDeploy.
- La dernière exécution de l'action.
- Si l'action a réussi ou a échoué.
- Les liens vers d'autres détails sur la dernière exécution de l'action, le cas échéant.
- Détails sur les révisions de source qui sont exécutées lors de la dernière exécution du pipeline au cours de la phase ou, pour les CodeDeploy déploiements, sur les dernières révisions de source déployées sur les instances cibles.
- Un bouton Afficher les détails qui ouvre une boîte de dialogue contenant des détails sur l'exécution de l'action, les journaux et la configuration de l'action.

> **Note**
>
> L'onglet Logs est disponible pour CodeBuild les CloudFormation actions exécutées dans le compte du pipeline.

4. Pour afficher les détails relatifs au fournisseur de l'action, choisissez le fournisseur. Par exemple, dans l'exemple de pipeline précédent, si vous choisissez CodeDeploy l'une des étapes de préparation ou de production, la page de CodeDeploy console du groupe de déploiement configuré pour cette étape s'affiche.
5. La progression d'une action est affichée à côté d'une action en cours (indiquée par un message En cours). Si l'action est en cours, vous voyez l'avancement en cours et les étapes ou les actions à mesure qu'elles surviennent.
6. Pour approuver ou refuser des actions configurées en vue d'une approbation manuelle, choisissez Review.
7. Pour retenter des actions qui ont échoué dans une étape, choisissez Retry.

8. L'état de la dernière exécution de l'action, y compris les résultats de cette action (réussite ou échec), est affiché.

Afficher les détails des actions dans un pipeline (console)

Vous pouvez consulter les détails d'un pipeline, y compris les détails des actions de chaque étape.

> **Note**
>
> La vue détaillée d'un pipeline cesse automatiquement de s'actualiser dans votre navigateur au bout d'une heure. Pour afficher des informations à jour, actualisez la page.

Pour afficher les détails des actions dans un pipeline

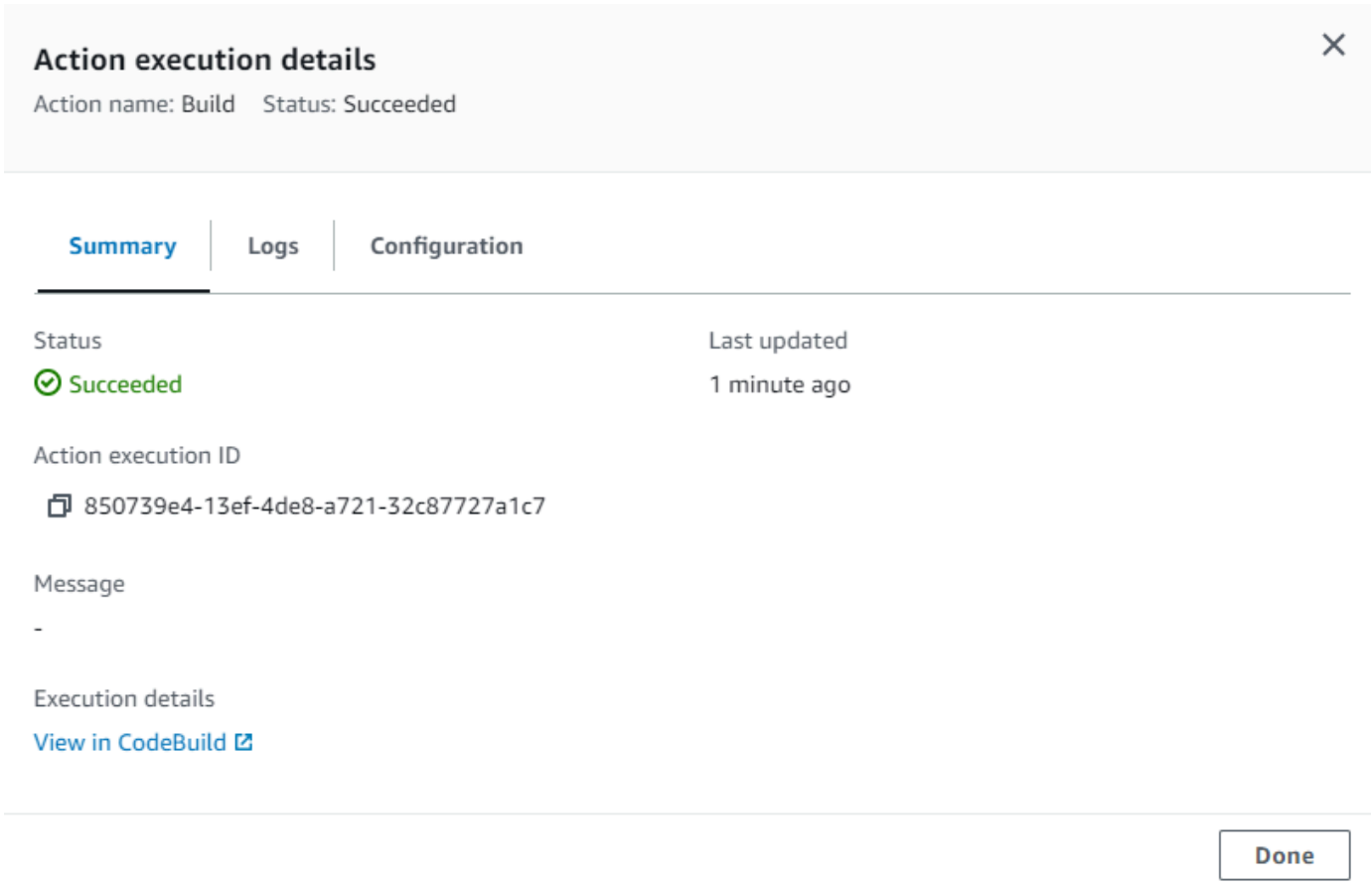
1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.

   La page Pipelines s'affiche.

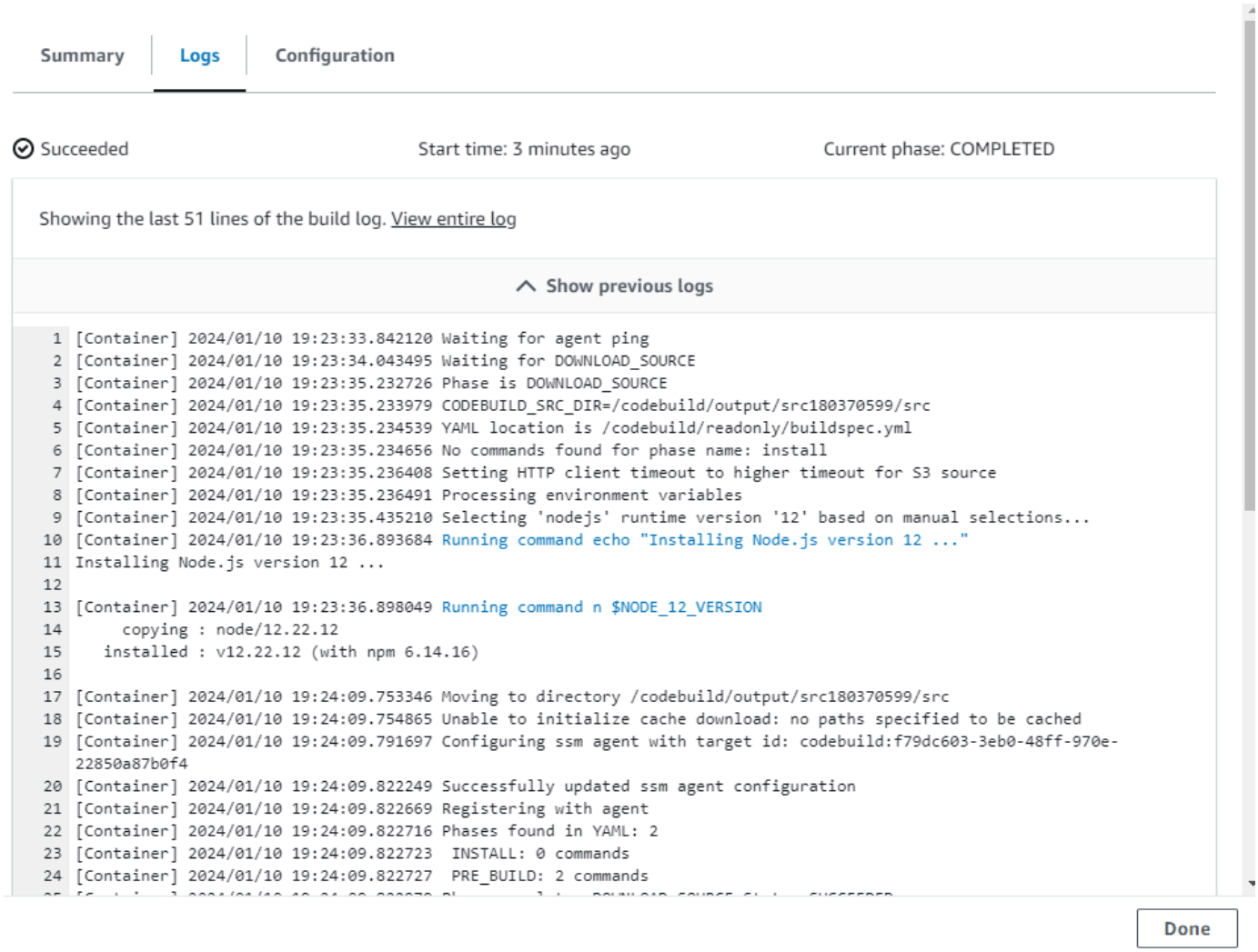
2. Pour n'importe quelle action, choisissez Afficher les détails pour ouvrir une boîte de dialogue contenant des détails sur l'exécution de l'action, les journaux et la configuration de l'action.

   > **Note**
   >
   > L'onglet Logs est disponible pour CodeBuild les CloudFormation actions.

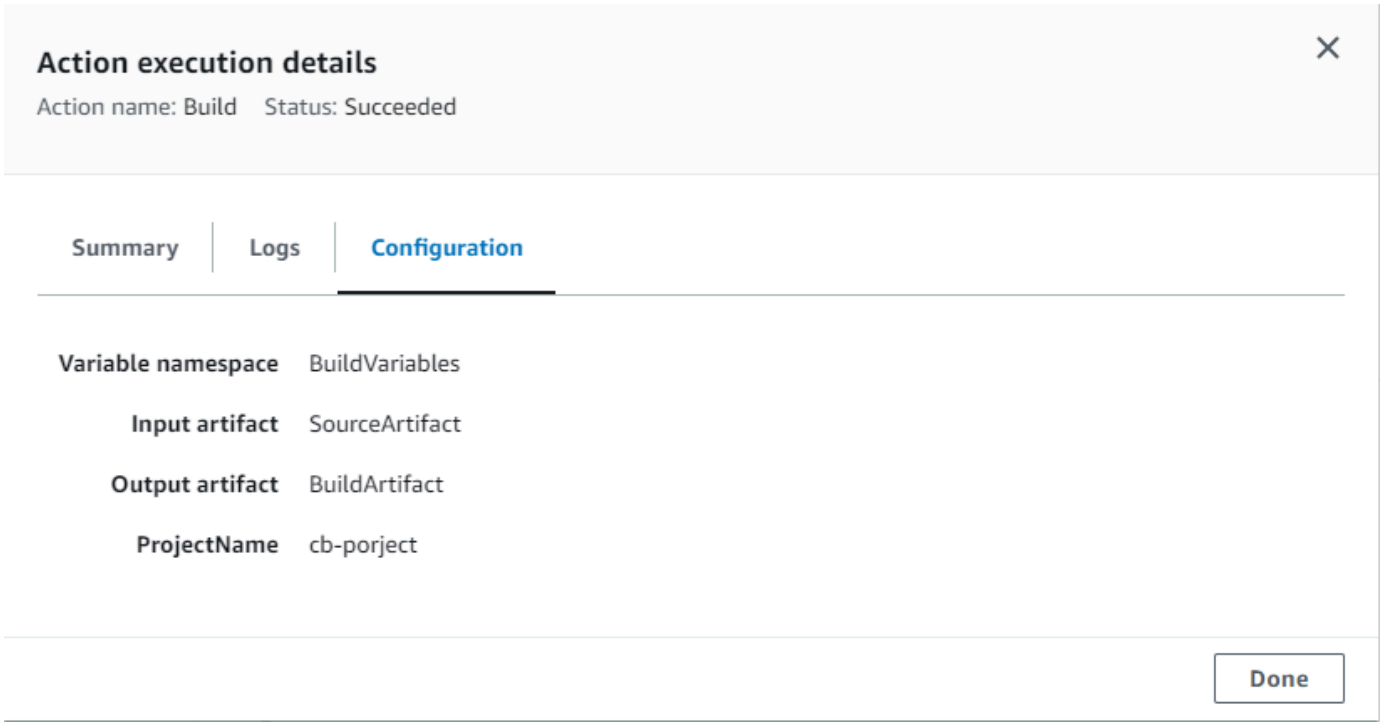
3. Pour voir le résumé d'une action dans une étape d'un pipeline, choisissez Afficher les détails de l'action, puis cliquez sur l'onglet Résumé.



4. Pour consulter les journaux d'actions d'une action contenant des journaux, choisissez Afficher les détails de l'action, puis sélectionnez l'onglet Journaux.



5. Pour consulter les détails de configuration d'une action, cliquez sur l'onglet Configuration.



Afficher l'ARN du pipeline et l'ARN du rôle de service (console)

Vous pouvez utiliser la console pour afficher les paramètres du pipeline, tels que l'ARN du pipeline, l'ARN du rôle de service et le magasin d'artefacts du pipeline.

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte seront affichés.
2. Choisissez le nom de votre pipeline, puis sélectionnez Paramètres dans le volet de navigation de gauche. La page affiche les informations suivantes :

   - Le nom du pipeline
   - Le pipeline Amazon Resource Name (ARN)

     L'ARN de pipeline est élaboré dans ce format :

     arn:aws:codepipeline : : : *region account pipeline-name*

     Exemple d'ARN de pipeline :

     `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

- L'ARN du rôle de CodePipeline service pour votre pipeline
- La version du pipeline
- Le nom et l'emplacement du magasin d'artefacts pour le pipeline

Affichage des détails et de l'historique d'un pipeline (interface de ligne de commande)

Vous pouvez exécuter les commandes suivantes pour afficher les détails sur vos pipelines et exécutions de pipeline :

- list-pipelinescommande pour afficher un résumé de tous les pipelines associés à votre AWS compte.
- Commande get-pipeline pour vérifier les détails d'un seul pipeline.
- list-pipeline-executions pour afficher des résumés des exécutions les plus récentes d'un pipeline.
- get-pipeline-executionpour afficher des informations sur une exécution d'un pipeline, ainsi que des informations sur les artéfacts, l'ID d'exécution du pipeline, et le nom, la version et l'état du pipeline
- get-pipeline-state commande pour afficher le pipeline, l'étape et le statut de l'action.
- list-action-executions pour afficher les détails de l'exécution de l'action d'un pipeline.

1. Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et utilisez le AWS CLI pour exécuter la list-pipelines commande :

   ```
   aws codepipeline list-pipelines
   ```

   Cette commande affiche une liste de tous les pipelines associés à votre compte AWS .

2. Pour afficher les détails relatifs à un pipeline, exécutez la commande get-pipeline, en spécifiant le nom unique du pipeline. Par exemple, pour afficher les détails relatifs à un pipeline nommé *MyFirstPipeline*, saisissez ce qui suit :

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

   Cette commande affiche la structure du pipeline.

Afficher les résultats des règles pour les conditions d'étape dans l'historique d'exécution

Vous pouvez consulter les résultats des règles pour une exécution dans laquelle une condition d'étape exécutait une règle et entraînait un résultat pour l'étape, tel qu'un retour en arrière ou un échec.

Les valeurs d'état valides pour les conditions et les règles sont les suivantes : `InProgress | Failed | Errored | Succeeded | Cancelled | Abandoned | Overridden`

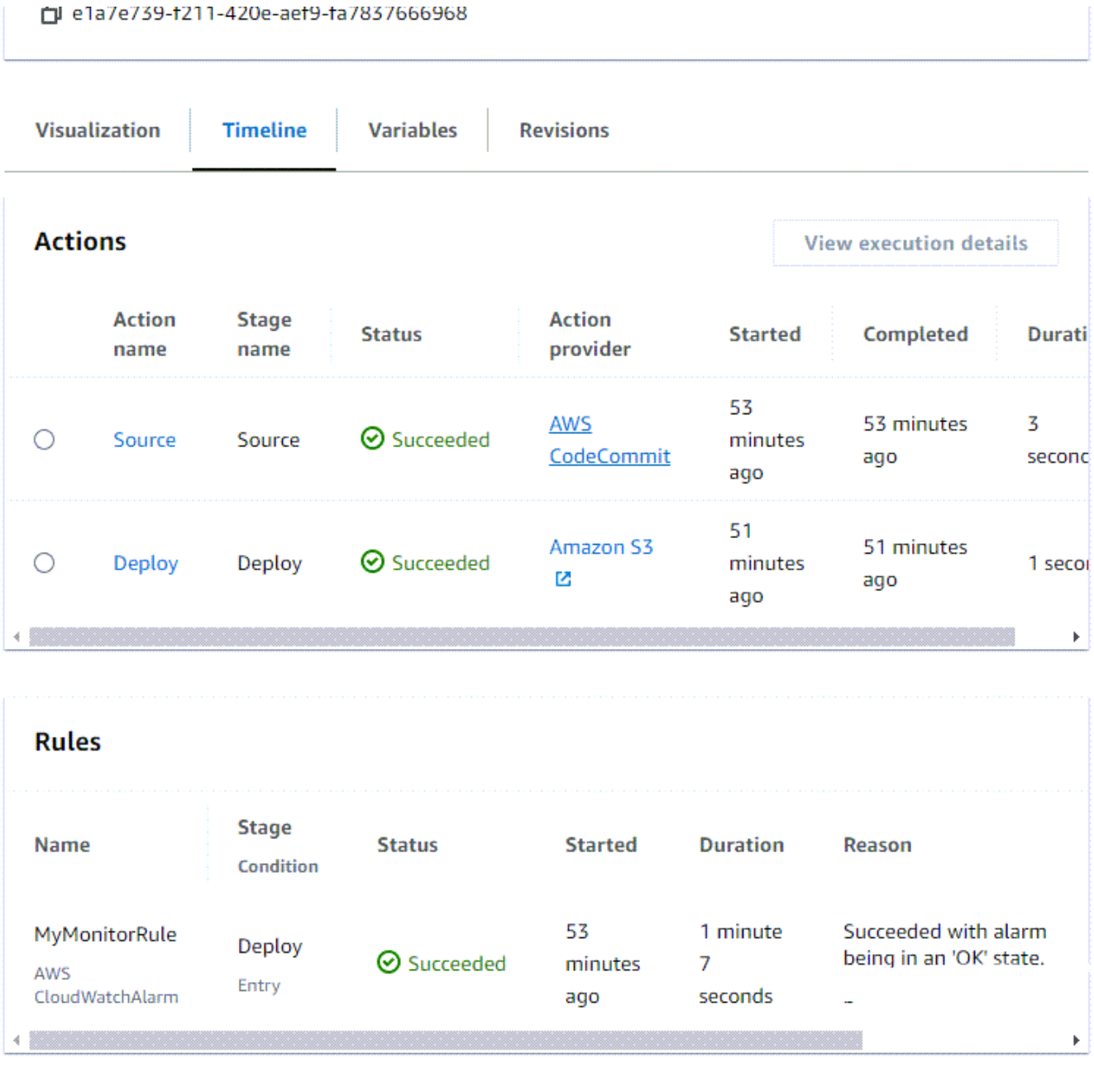
Afficher les résultats des règles pour les conditions d'étape dans l'historique d'exécution (console)

Vous pouvez utiliser la console pour afficher les résultats des règles d'une exécution dans laquelle une condition d'étape exécutait une règle et engageait un résultat pour l'étape.

Afficher les résultats des règles relatives aux conditions d'étape (console)

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.

   Les noms et le statut de tous les pipelines associés à votre pipeline Compte AWS sont affichés.
2. Dans Nom, choisissez le nom du pipeline que vous souhaitez consulter.
3. Choisissez Historique, puis sélectionnez l'exécution. Sur la page d'historique, choisissez l'onglet Chronologie. Sous Règles, consultez les résultats des règles pour l'exécution.



Afficher les résultats des règles pour les conditions d'étape avec **`list-rule-executions`** (CLI)

Vous pouvez utiliser la CLI pour afficher les résultats des règles d'une exécution dans laquelle une condition d'étape exécutait une règle et engageait un résultat pour l'étape.

- Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et utilisez le AWS CLI pour exécuter la list-rule-executions commande pour un pipeline nommé *`MyPipeline`* :

```
aws codepipeline list-rule-executions --pipeline-name MyFirstPipeline
```

Cette commande renvoie une liste de toutes les exécutions de règles terminées associées au pipeline.

L'exemple suivant montre les données renvoyées pour un pipeline avec une condition d'étape dans laquelle la règle est nommée*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."
                }
            }
        }
```

Supprimer un pipeline dans CodePipeline

Vous pouvez à tout moment modifier un pipeline pour mettre à jour ses fonctionnalités, ou encore le supprimer. Vous pouvez utiliser la AWS CodePipeline console ou la delete-pipeline commande du AWS CLI pour supprimer un pipeline.

Rubriques

- Suppression d'un pipeline (console)
- Suppression d'un pipeline (interface de ligne de commande)

Suppression d'un pipeline (console)

Pour supprimer un pipeline

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.

   Les noms et le statut de tous les pipelines associés à votre AWS compte sont affichés.
2. Dans Name, choisissez le nom du pipeline que vous souhaitez supprimer.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Sur la page Edit, choisissez Delete.
5. Tapez **`delete`** dans le champ pour confirmer, puis choisissez Supprimer.

> **Important**
>
> Cette action ne peut pas être annulée.

Suppression d'un pipeline (interface de ligne de commande)

Pour supprimer manuellement un pipeline, utilisez la commande delete-pipeline. AWS CLI

> **Important**
>
> La suppression d'un pipeline est irréversible. Il n'existe aucune boîte de dialogue de confirmation. Une fois la commande exécutée, le pipeline est supprimé, mais toutes les ressources utilisées dans ce dernier sont conservées. Cela permet de créer facilement un nouveau pipeline en utilisant ces ressources afin d'automatiser la publication de votre logiciel.

Pour supprimer un pipeline

1. Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et utilisez le AWS CLI pour exécuter la delete-pipeline commande, en spécifiant le nom du pipeline que vous souhaitez supprimer. Par exemple, pour supprimer un pipeline nommé *MyFirstPipeline* :

   ```
   aws codepipeline delete-pipeline --name MyFirstPipeline
   ```

   Cette commande ne donne aucun résultat.

2. Supprimez toutes les ressources dont vous n'avez plus besoin.

   > **Note**
   >
   > La suppression d'un pipeline ne supprime pas les ressources utilisées dans le pipeline, telles que l' CodeDeploy application Elastic Beanstalk que vous avez utilisée pour déployer votre code ou, si vous avez créé votre pipeline CodePipeline depuis la console, le compartiment Amazon CodePipeline S3 créé pour stocker les artefacts de vos pipelines. Assurez-vous de supprimer les ressources qui ne sont plus nécessaires afin de ne pas être facturé pour celles-ci par la suite. Par exemple, lorsque vous utilisez la console pour créer un pipeline pour la première fois, vous CodePipeline créez un compartiment Amazon S3 pour stocker tous les artefacts de tous vos pipelines. Si vous avez supprimé tous vos pipelines, suivez les étapes énoncées dansSuppression d'un compartiment.

Créez un pipeline CodePipeline qui utilise les ressources d'un autre AWS compte

Vous pouvez créer un pipeline qui utilise les ressources créées ou gérées par un autre compte AWS . Par exemple, vous pouvez utiliser un compte pour votre pipeline et un autre pour vos ressources CodeDeploy .

> **Note**
>
> Lorsque vous créez un pipeline avec des actions à partir de plusieurs comptes, vous devez configurer vos actions afin qu'elles puissent toujours accéder aux artefacts en fonction des limitations des pipelines entre comptes. Les limitations suivantes s'appliquent aux actions entre comptes :
>
> - En général, une action peut consommer un artefact seulement si :
>   - L'action est dans le même compte que le compte de pipeline OU
>   - L'artefact a été créé dans le compte de pipeline pour une action dans un autre compte OU
>   - L'artefact a été généré par une action précédente dans le même compte que l'action
>
>   En d'autres termes, vous ne pouvez pas transmettre un artefact d'un compte à un autre si aucun compte n'est le compte de pipeline.
> - Cross-account les actions ne sont pas prises en charge pour les types d'actions suivants :
>   - Les actions de génération Jenkins

Pour cet exemple, vous devez créer une clé AWS Key Management Service (AWS KMS) à utiliser, ajouter la clé au pipeline et configurer des politiques et des rôles de compte pour permettre l'accès entre comptes. Pour une clé AWS KMS, vous pouvez utiliser l'ID de clé, l'ARN de clé ou l'alias ARN.

> **Note**
>
> Les alias sont reconnus uniquement dans le compte qui a créé la clé KMS. Pour les actions entre comptes, vous ne pouvez utiliser l'ID de clé ou l'ARN de clé que pour identifier la clé. Cross-account les actions impliquent l'utilisation du rôle de l'autre compte (AccountB). Par conséquent, la spécification de l'ID de clé utilisera la clé de l'autre compte (AccountB).

Dans cette procédure pas à pas et ses exemples, vous *AccountA* trouverez le compte utilisé à l'origine pour créer le pipeline. Il a accès au compartiment Amazon S3 utilisé pour stocker les artefacts du pipeline et au rôle de service utilisé par AWS CodePipeline. *AccountB*est le compte initialement utilisé pour créer l' CodeDeploy application, le groupe de déploiement et le rôle de service utilisés par CodeDeploy.

*AccountA*Pour modifier un pipeline afin d'utiliser l' CodeDeploy application créée par*AccountB*, *AccountA* il faut :

- Demandez l'ARN ou l'ID de compte de *AccountB* (dans cette procédure pas à pas, l'*AccountB*ID est*012ID_ACCOUNT_B*).
- Créez ou utilisez une clé gérée par le AWS KMS client dans la région pour le pipeline, et accordez l'autorisation d'utiliser cette clé au rôle de service (*CodePipeline_Service_Role*) et*AccountB*.
- Créez une politique de compartiment Amazon S3 qui accorde l'*AccountB*accès au compartiment Amazon S3 (par exemple,*codepipeline-us-east-2-1234567890*).
- Créez une politique qui permet *AccountA* d'assumer un rôle configuré par *AccountB* et d'associer cette politique au rôle de service (*CodePipeline_Service_Role*).
- Modifiez le pipeline pour utiliser la AWS KMS clé gérée par le client au lieu de la clé par défaut.

Car *AccountB* pour permettre l'accès à ses ressources à un pipeline créé en*AccountA*, *AccountB* il faut :

- Demandez l'ARN ou l'ID de compte de *AccountA* (dans cette procédure pas à pas, l'*AccountA*ID est*012ID_ACCOUNT_A*).
- Créez une politique appliquée au rôle d'instance Amazon EC2 pour lequel il est configuré et CodeDeploy qui autorise l'accès au compartiment Amazon S3 (*codepipeline-us-east-2-1234567890*).
- Créez une politique appliquée au rôle d'instance Amazon EC2 pour lequel il est configuré et CodeDeploy qui autorise l'accès à la clé gérée par le AWS KMS client utilisée pour chiffrer les artefacts du pipeline. *AccountA*
- Configurez et associez un rôle IAM (*CrossAccount_Role*) avec une politique de relation de confiance qui permet au rôle de CodePipeline service *AccountA* d'assumer le rôle.
- Créez une politique qui autorise l'accès aux ressources de déploiement requises par le pipeline et associez-la à*CrossAccount_Role*.

- Créez une politique qui autorise l'accès au compartiment Amazon S3 (*codepipeline-us-east-2-1234567890*) et associez-le à*CrossAccount_Role*.

Rubriques

- Prérequis : créer un AWS KMS clé de chiffrement
- Étape 1 : Configurer des stratégies de compte et des rôles
- Étape 2 : Modifier le pipeline

Prérequis : créer un AWS KMS clé de chiffrement

Customer-managed les clés sont spécifiques à une région, comme toutes les AWS KMS clés. Vous devez créer votre AWS KMS clé gérée par le client dans la même région que celle où le pipeline a été créé (par exemple,`us-east-2`).

Pour créer une clé gérée par le client dans AWS KMS

1. Connectez-vous au Console de gestion AWS avec *AccountA* et ouvrez la AWS KMS console.
2. Sur la gauche, choisissez Customer managed keys (Clés gérées par le client).
3. Choisissez Create key. Dans Configure key (Configurer la clé), laissez l'option Symmetric (Symétrique) sélectionnée par défaut et choisissez Next (Suivant).
4. Dans Alias, entrez un alias à utiliser pour cette clé (par exemple,*PipelineName-Key*). Si vous le souhaitez, vous pouvez fournir une description et des balises pour cette clé, puis choisir Next (Suivant).
5. Dans Définir les autorisations administratives clés, choisissez le ou les rôles que vous souhaitez utiliser en tant qu'administrateurs pour cette clé, puis choisissez Suivant.
6. Dans Définir les autorisations d'utilisation des clés, sous Ce compte, sélectionnez le nom du rôle de service pour le pipeline (par exemple, CodePipeline _Service_Role). Sous Autres AWS comptes, choisissez Ajouter un autre AWS compte. Entrez l'ID de compte *AccountB* pour compléter l'ARN, puis choisissez Next.
7. Dans Review and edit key policy (Revoir et modifier la stratégie de clé), passez en revue la stratégie, puis choisissez Finish (Terminer).
8. Dans la liste des clés, choisissez l'alias de votre clé et copiez son ARN (par exemple, ***arn:aws:kms:us-east-2:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE***). Vous en aurez besoin lorsque vous modifierez votre pipeline et configurerez des stratégies.

Étape 1 : Configurer des stratégies de compte et des rôles

Après avoir créé la AWS KMS clé, vous devez créer et joindre des politiques qui permettront l'accès entre comptes. Cela nécessite des actions de la part de *AccountA* et*AccountB*.

Rubriques

- Configurez les politiques et les rôles dans le compte qui créera le pipeline (Compte A)
- Configurez les politiques et les rôles dans le compte propriétaire du AWS ressource (Compte B)

Configurez les politiques et les rôles dans le compte qui créera le pipeline (***Compte A***)

Pour créer un pipeline qui utilise les CodeDeploy ressources associées à un autre AWS compte, vous *AccountA* devez configurer des politiques à la fois pour le compartiment Amazon S3 utilisé pour stocker les artefacts et pour le rôle de service pour CodePipeline.

Pour créer une politique pour le compartiment Amazon S3 qui accorde l'accès à AccountB (console)

1. Connectez-vous au Console de gestion AWS avec *AccountA* et ouvrez la console Amazon S3 à l'adresse https://console.aws.amazon.com/s3/.
2. Dans la liste des compartiments Amazon S3, choisissez le compartiment Amazon S3 dans lequel les artefacts de vos pipelines sont stockés. Ce compartiment est nommé`codepipeline-`*`region`*`-`*`1234567EXAMPLE`*, où se *`region`* trouve la AWS région dans laquelle vous avez créé le pipeline. *`1234567EXAMPLE`* Il s'agit d'un nombre aléatoire à dix chiffres qui garantit que le nom du compartiment est unique (par exemple,*`codepipeline-us-east-2-1234567890`*).
3. Sur la page détaillée du compartiment Amazon S3, choisissez Propriétés.
4. Dans le volet des propriétés, développez Autorisations, puis choisissez Ajouter une stratégie de compartiment.

> **Note**
>
> Si une politique est déjà associée à votre compartiment Amazon S3, choisissez Modifier la politique du compartiment. Vous pouvez ensuite ajouter les déclarations de l'exemple suivant à la stratégie existante. Pour ajouter une nouvelle politique, cliquez sur le lien et suivez les instructions du générateur AWS de politiques. Pour plus d'informations, consultez la section Présentation des politiques IAM.

5. Dans la fenêtre Bucket Policy Editor, créez une politique qui autorisera l'*AccountB*accès aux artefacts du pipeline et permettra *AccountB* d'ajouter des artefacts de sortie si une action, telle qu'une source personnalisée ou une action de génération, les crée.
6. Choisissez Save, puis fermez l'éditeur de stratégie.
7. Choisissez Enregistrer pour enregistrer les autorisations pour le compartiment Amazon S3.

Pour créer une politique pour le rôle de service pour CodePipeline (console)

1. Connectez-vous au Console de gestion AWS avec *AccountA* et ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le panneau de navigation, choisissez Rôles.
3. Dans la liste des rôles, sous Nom de rôle, choisissez le nom du rôle de service pour CodePipeline.
4. Sous l'onglet Autorisations, sélectionnez Ajouter une politique en ligne.
5. Choisissez l'onglet JSON et entrez la politique suivante pour *AccountB* permettre d'assumer le rôle. Dans l'exemple suivant, *012ID_ACCOUNT_B* est l'ARN pour *AccountB* :

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": [
            "arn:aws:iam::111122223333:role/*"
        ]
    }
}
```

6. Choisissez Examiner une politique.
7. Dans le champ Name (Nom), saisissez un nom pour cette stratégie. Choisissez Create Policy (Créer une politique).

Configurez les politiques et les rôles dans le compte propriétaire du AWS ressource (*Compte B*)

Lorsque vous créez une application, un déploiement et un groupe de déploiement dans CodeDeploy, vous créez également un rôle d'instance Amazon EC2. (Ce rôle est créé si vous utilisez l'assistant d'exécution du déploiement, mais vous pouvez également le créer manuellement.) Pour qu'un pipeline créé dans *AccountA* utilise CodeDeploy les ressources créées dans*AccountB*, vous devez :

- Configurez une politique pour le rôle d'instance qui lui permet d'accéder au compartiment Amazon S3 dans lequel sont stockés les artefacts du pipeline.
- Créez un deuxième rôle dans la *AccountB* configuration pour l'accès entre comptes.

  Ce second rôle doit non seulement avoir accès au compartiment Amazon S3 dans*AccountA*, mais il doit également contenir une politique qui autorise l'accès aux CodeDeploy ressources et une politique de relation de confiance qui permet au rôle de CodePipeline service dans *AccountA* d'assumer le rôle.

> **Note**
>
> Ces politiques sont spécifiques à la configuration CodeDeploy des ressources à utiliser dans un pipeline créé à l'aide d'un AWS compte différent. Les autres AWS ressources nécessiteront des politiques spécifiques à leurs besoins en ressources.

Pour créer une politique pour le rôle d'instance Amazon EC2 configuré pour CodeDeploy (console)

1. Connectez-vous au Console de gestion AWS avec *AccountB* et ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le panneau de navigation, choisissez Rôles.
3. Dans la liste des rôles, sous Nom du rôle, choisissez le nom du rôle de service utilisé comme rôle d'instance Amazon EC2 pour l' CodeDeploy application. Le nom de ce rôle peut varier, et plusieurs rôles d'instance peuvent être utilisés par un groupe de déploiement. Pour plus d'informations, consultez Créer un profil d'instance IAM pour vos instances Amazon EC2.
4. Sous l'onglet Autorisations, sélectionnez Ajouter une politique en ligne.

5. Choisissez l'onglet JSON et entrez la politique suivante pour autoriser l'accès au compartiment Amazon S3 utilisé *AccountA* pour stocker les artefacts pour les pipelines (dans cet exemple,*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. Choisissez Examiner une politique.
7. Dans le champ Name (Nom), saisissez un nom pour cette stratégie. Choisissez Create Policy (Créer une politique).
8. Créez une deuxième politique indiquant AWS KMS où se ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** trouve l'ARN de la clé gérée par le client créée *AccountA* et configurée *AccountB* pour permettre son utilisation :

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**
>
> Vous devez utiliser l'ID de compte indiqué *AccountA* dans cette politique dans le cadre de l'ARN de ressource pour la AWS KMS clé, comme indiqué ici, sinon la stratégie ne fonctionnera pas.

9. Choisissez Examiner une politique.
10. Dans le champ Name (Nom), saisissez un nom pour cette stratégie. Choisissez Create Policy (Créer une politique).

Créez maintenant un rôle IAM à utiliser pour l'accès entre comptes et configurez-le de manière à ce que le rôle de CodePipeline service dans *AccountA* puisse assumer ce rôle. Ce rôle doit contenir des politiques qui autorisent l'accès aux CodeDeploy ressources et au compartiment Amazon S3 utilisé pour stocker les artefacts*AccountA*.

Pour configurer le rôle multicompte dans IAM

1. Connectez-vous au Console de gestion AWS avec *AccountB* et ouvrez la console IAM à https://console.aws.amazon.com/iam l'adresse.
2. Dans le panneau de navigation, choisissez Rôles. Sélectionnez Create role (Créer un rôle).
3. Sous Sélectionner un type d'entité de confiance, choisissez Autre compte AWS . Sous Spécifier les comptes qui peuvent utiliser ce rôle , dans ID de AWS compte, entrez l'ID du compte qui créera le pipeline dans CodePipeline (*AccountA*), puis choisissez Suivant : Autorisations.

   > **Important**
   >
   > Cette étape crée la politique de relation de confiance entre *AccountB* et*AccountA*. Toutefois, cela accorde un accès de niveau root au compte et CodePipeline recommande de le limiter au rôle de CodePipeline service dans*AccountA*. Suivez l'étape 16 pour restreindre les autorisations.

4. Dans Joindre des politiques d'autorisations AmazonS3ReadOnlyAccess, choisissez, puis choisissez Suivant : Tags.

   > **Note**
   >
   > Ce n'est pas la stratégie que vous allez utiliser. Vous devez choisir une stratégie pour mettre fin à l'assistant.

5. Choisissez Suivant : Vérification. Tapez un nom pour ce rôle dans Nom du rôle (par exemple,*CrossAccount_Role*). Vous pouvez donner à ce rôle le nom que vous voulez, à condition qu'il respecte les conventions de dénomination d'IAM. Pensez à donner au rôle un nom qui indique clairement son but. Choisissez Create Role (Créer un rôle).
6. Dans la liste des rôles, choisissez le rôle que vous venez de créer (par exemple*CrossAccount_Role*) pour ouvrir la page de résumé de ce rôle.
7. Sous l'onglet Autorisations, sélectionnez Ajouter une politique en ligne.
8. Choisissez l'onglet JSON et entrez la politique suivante pour autoriser l'accès aux CodeDeploy ressources :

   JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:GetApplicationRevision",
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource": "*"
    }
  ]
}
```

9. Choisissez Examiner une politique.
10. Dans le champ Name (Nom), saisissez un nom pour cette stratégie. Choisissez Create Policy (Créer une politique).
11. Sous l'onglet Autorisations, sélectionnez Ajouter une politique en ligne.
12. Choisissez l'onglet JSON et entrez la politique suivante pour permettre à ce rôle de récupérer les artefacts d'entrée et de placer les artefacts de sortie dans le compartiment Amazon S3 dans *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. Choisissez Examiner une politique.
14. Dans le champ Name (Nom), saisissez un nom pour cette stratégie. Choisissez Create Policy (Créer une politique).
15. Dans l'onglet Autorisations, recherchez AmazonS3ReadOnlyAccess dans la liste des politiques sous Nom de la stratégie, puis choisissez l'icône de suppression (X) à côté de la stratégie. A l'invite, choisissez Detach.
16. Sélectionnez l'onglet Relation de confiance, puis choisissez Modifier la politique de confiance. Choisissez l'option Ajouter un principal dans la colonne de gauche. Pour Type principal, choisissez IAM Roles, puis indiquez l'ARN du rôle de CodePipeline service dans*AccountA*. `arn:aws:iam::Account_A:root`Supprimez-le de la liste AWS des principaux, puis choisissez Mettre à jour la politique.

Étape 2 : Modifier le pipeline

Vous ne pouvez pas utiliser la CodePipeline console pour créer ou modifier un pipeline qui utilise des ressources associées à un autre AWS compte. Cependant, vous pouvez utiliser la console pour créer la structure générale du pipeline, puis l'utiliser AWS CLI pour modifier le pipeline et ajouter ces ressources. Vous pouvez également utiliser la structure d'un pipeline existant et y ajouter manuellement les ressources.

Pour ajouter les ressources associées à une autre AWS compte (AWS CLI)

1. Sur un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows), exécutez la get-pipeline commande sur le pipeline auquel vous souhaitez ajouter des ressources. Copiez le résultat de la commande dans un fichier JSON. Par exemple, pour un pipeline nommé MyFirstPipeline, vous devez saisir quelque chose de semblable à ce qui suit :

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Le résultat est envoyé au fichier *pipeline.json*.
2. Ouvrez le fichier JSON dans un éditeur de texte brut. Une fois `"type": "S3"` dans le magasin d'artefacts, ajoutez la clé de chiffrement KMS, l'ID et les informations de type, *codepipeline-us-east-2-1234567890* y compris le nom du compartiment Amazon S3 utilisé pour stocker les artefacts pour le pipeline et ***arn:aws:kms:us-***

east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE l'ARN de la clé gérée par le client que vous venez de créer :

```
{
  "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. Ajoutez une action de déploiement dans une étape pour utiliser les CodeDeploy ressources associées*AccountB*, y compris les `roleArn` valeurs du rôle multicompte que vous avez créé (*CrossAccount_Role*).

   L'exemple suivant montre le JSON qui ajoute une action de déploiement nommée*ExternalDeploy*. Il utilise les CodeDeploy ressources créées *AccountB* dans une étape nommée*Staging*. Dans l'exemple suivant, l'ARN pour *AccountB* est *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**
>
> Il ne s'agit pas du format JSON pour l'ensemble du pipeline, mais seulement de la structure de l'action dans une étape.

4. Vous devez supprimer les lignes `metadata` du fichier afin que la commande update-pipeline puisse l'utiliser. Supprimez la section de la structure de pipeline dans le fichier JSON (les lignes `"metadata": { }` et les champs `"created"`, `"pipelineARN"` et `"updated"`).

   Par exemple, supprimez les lignes suivantes de la structure :

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

   Enregistrez le fichier.

5. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline d'une manière similaire à l'exemple suivant :

> **Important**
>
> N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

Cette commande affiche toute la structure du pipeline mise à jour.

Pour tester le pipeline qui utilise des ressources associées à un autre AWS compte

1. Sur un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows), exécutez la start-pipeline-execution commande en spécifiant le nom du pipeline, comme suit :

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

   Pour de plus amples informations, veuillez consulter Lancement manuel d'un pipeline.

2. Connectez-vous au Console de gestion AWS avec *AccountA* et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés.

3. Dans Name, choisissez le nom du pipeline que vous venez de modifier. Une vue détaillée du pipeline s'affiche alors, laquelle indique notamment l'état de chaque action, dans chaque étape du pipeline.

4. Observez la progression à travers le pipeline. Attendez un message de réussite concernant l'action qui utilise la ressource associée à un autre AWS compte.

   > **Note**
   >
   > Vous recevrez un message d'erreur si vous essayez d'afficher les détails de l'action alors que vous êtes connecté*AccountA*. Déconnectez-vous, puis connectez-vous avec *AccountB* pour afficher les détails du déploiement dans CodeDeploy.

Migrez les pipelines de sondage pour utiliser la détection des modifications basée sur les événements

AWS CodePipeline prend en charge une livraison continue complète de bout en bout, y compris le démarrage de votre pipeline chaque fois qu'il y a un changement de code. Il existe deux méthodes prises en charge pour démarrer votre pipeline en cas de modification de code : la détection des modifications basée sur les événements et le sondage. Nous vous recommandons d'utiliser la détection des modifications basée sur les événements pour les pipelines.

Utilisez les procédures décrites ici pour migrer (mettre à jour) vos pipelines de sondage vers la méthode de détection des modifications basée sur les événements pour votre pipeline.

La méthode de détection des changements basée sur les événements recommandée pour les pipelines est déterminée par la source du pipeline, telle que. CodeCommit Dans ce cas, par exemple, le pipeline de sondage devra migrer vers la détection des modifications basée sur les événements avec. EventBridge

Comment migrer les pipelines de sondages

Pour migrer les pipelines de sondage, déterminez vos pipelines de sondage, puis déterminez la méthode recommandée de détection des modifications basée sur les événements :

- Suivez les étapes décrites Afficher les pipelines de sondages dans votre compte pour déterminer vos pipelines de sondages.
- Dans le tableau, recherchez le type de source de votre pipeline, puis choisissez la procédure avec l'implémentation que vous souhaitez utiliser pour migrer votre pipeline de sondage. Chaque section contient plusieurs méthodes de migration, telles que l'utilisation de l'interface de ligne de commande ou CloudFormation.

Migration de pipelines vers la méthode recommandée de détection des modifications		
Source du pipeline	Méthode de détection basée sur les événements recommandée	Procédures de migration
AWS CodeCommit	EventBridge (recommandé).	Consultez Migrer les pipelines de sondage avec une CodeCommit source.
Amazon S3	EventBridge et compartiment activé pour les notifications d'événements (recommandé).	Consultez Migrer les pipelines de sondage avec une source S3 activée pour les événements.
Amazon S3	EventBridge et une AWS CloudTrail piste.	Consultez Migrer les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3.

Migration de pipelines vers la méthode recommandée de détection des modifications		
Source du pipeline	Méthode de détection basée sur les événements recommandée	Procédures de migration
GitHub (via GitHub l'application)	Connexions (recommandées)	Consultez Migrer les pipelines de sondage pour une action source GitHub (via l'application OAuth) vers les connexions.
GitHub (via l'application OAuth)	Webhooks	Consultez Migrer les pipelines de sondage pour une action source GitHub (via l'application OAuth) vers des webhooks.

Important

Pour les mises à jour de configuration des actions de pipeline applicables, telles que les pipelines comportant une action GitHub (via l'application OAuth), vous devez définir explicitement le `PollForSourceChanges` paramètre sur false dans la configuration de votre action Source pour empêcher l'interrogation d'un pipeline. Par conséquent, il est possible de configurer par erreur un pipeline avec à la fois une détection des changements basée sur des événements et un sondage, par exemple en configurant une EventBridge règle et en omettant le paramètre. `PollForSourceChanges` Cela génère des exécutions de pipeline en double ; le pipeline est compté dans le nombre total de pipelines d'interrogation, qui, par défaut, est beaucoup plus faible que celui des pipelines basés sur les événements. Pour de plus amples informations, veuillez consulter Quotas dans AWS CodePipeline.

Afficher les pipelines de sondages dans votre compte

Dans un premier temps, utilisez l'un des scripts suivants pour déterminer quels pipelines de votre compte sont configurés pour les sondages. Il s'agit des pipelines permettant de migrer vers la détection des modifications basée sur les événements.

Afficher les pipelines de sondage dans votre compte (script)

Suivez ces étapes pour utiliser un script afin de déterminer les pipelines de votre compte qui utilisent le sondage.

1. Ouvrez une fenêtre de terminal, puis effectuez l'une des opérations suivantes :

   - Exécutez la commande suivante pour créer un nouveau script nommé PollingPipelinesExtractor.sh.

     ```
     vi PollingPipelinesExtractor.sh
     ```

   - Pour utiliser un script Python, exécutez la commande suivante pour créer un nouveau script Python nommé PollingPipelinesExtractor.py.

     ```
     vi PollingPipelinesExtractor.py
     ```

2. Copiez et collez le code suivant dans le PollingPipelinesExtractor script. Effectuez l'une des actions suivantes :

   - Copiez et collez le code suivant dans le 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."
```

- Copiez et collez le code suivant dans le 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. Pour chaque région où vous avez des pipelines, vous devez exécuter le script correspondant. Pour exécuter le script, effectuez l'une des opérations suivantes :

- Exécutez la commande suivante pour exécuter le script nommé PollingPipelinesExtractor.sh. Dans cet exemple, la région est us-west-2.

```
./PollingPipelinesExtractor.sh us-west-2
```

- Pour le script python, exécutez la commande suivante pour exécuter le script python nommé PollingPipelinesExtractor.py. Dans cet exemple, la région est us-west-2.

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

Dans l'exemple de sortie du script suivant, la région us-west-2 a renvoyé une liste de pipelines de sondage et indique l'heure de dernière exécution pour chaque 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...%
```

Analysez la sortie du script et, pour chaque pipeline de la liste, mettez à jour la source d'interrogation en utilisant la méthode de détection des modifications basée sur les événements recommandée.

> **Note**
>
> Vos pipelines de sondage sont déterminés par la configuration d'action du pipeline pour le `PollForSourceChanges` paramètre. Si le `PollForSourceChanges` paramètre est omis dans la configuration de la source du pipeline, la CodePipeline valeur par défaut est d'interroger votre référentiel pour détecter les modifications apportées à la source. Ce comportement est le même que s'`PollForSourceChanges`il était inclus et défini sur true. Pour plus d'informations, consultez les paramètres de configuration de l'action

source de votre pipeline, tels que les paramètres de configuration de l'action source Amazon S3 dansRéférence des actions source d'Amazon S3.

Notez que ce script génère également un fichier .csv contenant la liste des pipelines de sondage de votre compte et enregistre le fichier .csv dans le dossier de travail actuel.

Migrer les pipelines de sondage avec une CodeCommit source

Vous pouvez migrer votre pipeline de sondage EventBridge afin de détecter les modifications apportées à votre référentiel CodeCommit source ou à votre compartiment source Amazon S3.

CodeCommit-- Pour un pipeline avec une CodeCommit source, modifiez le pipeline afin que la détection des changements soit automatisée EventBridge. Choisissez l'une des méthodes suivantes pour implémenter la migration :

- Console : Migrer les pipelines de sondage (CodeCommit ou la source Amazon S3) (console)
- CLI : Migrer les pipelines de sondage (CodeCommit source) (CLI)
- CloudFormation: Migrer les pipelines de sondage (CodeCommit source) (CloudFormation modèle)

Migrer les pipelines de sondage (CodeCommit ou la source Amazon S3) (console)

Vous pouvez utiliser la CodePipeline console pour mettre à jour votre pipeline afin de détecter les modifications apportées à votre référentiel CodeCommit source ou à votre compartiment source Amazon S3. EventBridge

> **Note**
>
> Lorsque vous utilisez la console pour modifier un pipeline contenant un référentiel CodeCommit source ou un compartiment source Amazon S3, la règle et le rôle IAM sont créés pour vous. Si vous utilisez le AWS CLI pour modifier le pipeline, vous devez créer vous-même la EventBridge règle et le rôle IAM. Pour de plus amples informations, veuillez consulter CodeCommit actions à la source et EventBridge.

Utilisez ces étapes pour modifier un pipeline qui utilise les vérifications périodiques. Si vous voulez créer un pipeline, consultez Création d'un pipeline, d'étapes et d'actions.

Pour modifier l'étape source du pipeline

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés.
2. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier. Une vue détaillée du pipeline s'affiche alors, laquelle indique notamment l'état de chaque action, dans chaque étape du pipeline.
3. Sur la page des détails du pipeline, choisissez Edit.
4. À l'étape Modifier, choisissez l'icône de modification sur l'action source.
5. Développez les options de détection des modifications et choisissez Utiliser les CloudWatch événements pour démarrer automatiquement mon pipeline lorsqu'un changement se produit (recommandé).

   Un message s'affiche indiquant la EventBridge règle à créer pour ce pipeline. Choisissez Mettre à jour.

   Si vous mettez à jour un pipeline qui possède une source Amazon S3, le message suivant s'affiche. Choisissez Mettre à jour.
6. Lorsque vous avez terminé de modifier votre pipeline, cliquez sur Save pipeline changes pour revenir à la page récapitulative.

   Un message affiche le nom de la EventBridge règle à créer pour votre pipeline. Choisissez Save and continue (Enregistrer et continuer).
7. Pour tester votre action, publiez une modification en utilisant le AWS CLI pour valider une modification de la source spécifiée dans l'étape source du pipeline.

Migrer les pipelines de sondage (CodeCommit source) (CLI)

Suivez ces étapes pour modifier un pipeline qui utilise un sondage (contrôles périodiques) afin d'utiliser une EventBridge règle pour démarrer le pipeline. Si vous voulez créer un pipeline, consultez Création d'un pipeline, d'étapes et d'actions.

Pour créer un pipeline piloté par des événements avec CodeCommit, vous modifiez le `PollForSourceChanges` paramètre de votre pipeline, puis vous créez les ressources suivantes :

- EventBridge événement

- Rôle IAM pour autoriser cet événement à lancer votre pipeline

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

1. Exécutez la commande get-pipeline pour copier la structure de pipeline dans un fichier JSON. Par exemple, pour un pipeline nommé `MyFirstPipeline`, exécutez la commande suivante :

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Cette commande ne renvoie rien, mais le fichier que vous avez créé doit apparaître dans le répertoire où vous avez exécuté la commande.

2. Ouvrez le fichier JSON dans un éditeur de texte brut et modifiez l'étape source en remplaçant la valeur du paramètre `PollForSourceChanges` par `false`, comme illustré dans cet exemple.

   Pourquoi est-ce que j'effectue cette modification ? Le remplacement de la valeur de ce paramètre par `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

   ```
   "configuration": {
       "PollForSourceChanges": "false",
       "BranchName": "main",
       "RepositoryName": "MyTestRepo"
   },
   ```

3. Si vous utilisez la structure de pipeline extraite à l'aide de la commande get-pipeline, supprimez les lignes `metadata` du fichier JSON. Sinon, la commande update-pipeline ne peut pas l'utiliser. Supprimez les lignes `"metadata": { }` et les champs `"updated"`, `"created"` et `"pipelineARN"`.

Par exemple, supprimez les lignes suivantes de la structure :

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

Enregistrez le fichier.

4. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline :

> **Important**
>
> N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

Cette commande affiche toute la structure du pipeline mise à jour.

> **Note**
>
> La commande update-pipeline interrompt le pipeline. Si une révision est exécutée dans le pipeline lorsque vous exécutez la commande update-pipeline celle-ci est interrompue. Vous devez lancer manuellement le pipeline pour exécuter cette révision dans le pipeline mis à jour. Utilisez la commande **`start-pipeline-execution`** pour démarrer manuellement votre pipeline.

Pour créer une EventBridge règle avec CodeCommit comme source d'événement et CodePipeline comme cible

1. Ajoutez des autorisations EventBridge à utiliser CodePipeline pour invoquer la règle. Pour plus d'informations, consultez la section Utilisation de politiques basées sur les ressources pour Amazon. EventBridge

a. Utilisez l'exemple suivant pour créer la politique de confiance qui permet EventBridge d'assumer le rôle de service. Nommez la stratégie d'approbation `trustpolicyforEB.json`.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Utilisez la commande suivante pour créer le rôle `Role-for-MyRule` et attachez la stratégie d'approbation.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Créez le JSON de stratégie d'autorisations comme indiqué dans cet exemple pour le pipeline nommé `MyFirstPipeline`. Nommez la stratégie d'autorisations `permissionspolicyforEB.json`.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
```

```
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilisez la commande suivante pour attacher au rôle `Role-for-MyRule` la stratégie d'autorisations `CodePipeline-Permissions-Policy-for-EB`.

Pourquoi est-ce que j'effectue cette modification ? L'ajout de cette politique au rôle crée des autorisations pour EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Appelez la commande put-rule et incluez les paramètres `--name`, `--event-pattern` et `--role-arn`.

Pourquoi est-ce que j'effectue cette modification ? Cette commande permet à CloudFormation de créer l'événement.

L'exemple de commande suivant crée une règle nommée `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. Pour ajouter CodePipeline en tant que cible, appelez la put-targets commande et incluez les paramètres suivants :

- Le paramètre `--rule` s'utilise avec le la règle `rule_name` que vous avez créée à l'aide de la commande put-rule.
- Le paramètre `--targets` s'utilise avec l'ID de liste `Id` de la cible figurant dans la liste des cibles et l'ARN du pipeline cible.

L'exemple de commande suivant spécifie que pour la règle appelée `MyCodeCommitRepoRule`, l'`Id` cible est composé du numéro un, ce qui indique qu'il s'agit de la règle 1 dans une liste de

cibles pour la règle. L'exemple de commande spécifie également un exemple d'ARN pour le pipeline. Le pipeline démarre lorsque des modifications sont effectuées dans le référentiel.

```
aws events put-targets --rule MyCodeCommitRepoRule --targets
 Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

4. (Facultatif) Pour configurer un transformateur d'entrée avec des remplacements de source pour un ID d'image spécifique, utilisez le JSON suivant dans votre commande CLI. L'exemple suivant configure une dérogation dans laquelle :

   - `Source`Dans cet exemple`actionName`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - `COMMIT_ID`Dans cet exemple`revisionType`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - Dans cet exemple`revisionValue`, **<** *`revisionValue`* **>** est dérivé de la variable d'événement source.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "pipeline-ARN",
            "InputTransformer": {
                "sourceRevisions": {
                    "actionName": "Source",
                    "revisionType": "COMMIT_ID",
                    "revisionValue": "<revisionValue>"
                },
                "variables": [
                    {
                        "name": "Branch_Name",
                        "value": "value"
                    }
                ]
            }
        }
    ]
}
```

Migrer les pipelines de sondage (CodeCommit source) (CloudFormation modèle)

Pour créer un pipeline piloté par les événements avec AWS CodeCommit, vous modifiez le `PollForSourceChanges` paramètre de votre pipeline, puis vous ajoutez les ressources suivantes à votre modèle :

- Une EventBridge règle
- Un rôle IAM pour votre règle EventBridge

Si vous l'utilisez CloudFormation pour créer et gérer vos pipelines, votre modèle inclut un contenu tel que le suivant.

> **Note**
>
> La propriété `Configuration` de l'étape source appelée `PollForSourceChanges`. Si cette propriété n'est pas incluse dans votre modèle, `PollForSourceChanges` est défini sur `true` par défaut.

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
        }]
    },
```

Pour mettre à jour votre pipeline CloudFormation modèle et création de EventBridge règle

1. Dans le modèle, ci-dessous`Resources`, utilisez la `AWS::IAM::Role` CloudFormation ressource pour configurer le rôle IAM qui permet à votre événement de démarrer votre pipeline. Cette entrée crée un rôle qui utilise deux stratégies :

   - La première stratégie autorise le rôle à être endossé.

- La deuxième stratégie fournit des autorisations pour démarrer le pipeline.

Pourquoi est-ce que j'effectue cette modification ? L'ajout de la `AWS::IAM::Role` ressource permet CloudFormation de créer des autorisations pour EventBridge. Cette ressource est ajoutée à votre CloudFormation pile.

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. Dans le modèle, ci-dessous`Resources`, utilisez la `AWS::Events::Rule` CloudFormation ressource pour ajouter une EventBridge règle. Ce modèle d'événement crée un événement qui

surveille les modifications push apportées à votre référentiel. Lorsqu'un changement d'état du référentiel est EventBridge détecté, la règle est invoquée `StartPipelineExecution` sur votre pipeline cible.

Pourquoi est-ce que je fais ce changement ? L'ajout de la `AWS::Events::Rule` ressource CloudFormation permet de créer l'événement. Cette ressource est ajoutée à votre CloudFormation pile.

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. (Facultatif) Pour configurer un transformateur d'entrée avec des remplacements de source pour un ID d'image spécifique, utilisez l'extrait YAML suivant. L'exemple suivant configure une dérogation dans laquelle :

   - `Source`Dans cet exemple`actionName`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - `COMMIT_ID`Dans cet exemple`revisionType`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - Dans cet exemple`revisionValue`, < *`revisionValue`* > est dérivé de la variable d'événement source.
   - Les variables de sortie pour `BranchName` et `Value` sont spécifiées.

```
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
    sourceRevisions:
      actionName: Source
      revisionType: COMMIT_ID
      revisionValue: <revisionValue>
    variables:
    - name: BranchName
      value: value
```

4. Enregistrez le modèle mis à jour sur votre ordinateur local, puis ouvrez la CloudFormation console.
5. Choisissez votre pile, puis Créer un jeu de modifications pour la pile actuelle.
6. Chargez le modèle mis à jour, puis affichez les modifications répertoriées dans CloudFormation. Il s'agit des modifications apportées à la pile. Vos nouvelles ressources doivent figurer dans la liste.
7. Sélectionnez Execute (Exécuter).

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Dans de nombreux cas, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut lorsque vous créez un pipeline. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

- Dans le modèle, remplacez la valeur du paramètre `PollForSourceChanges` par `false`. Si vous n'avez pas inclus `PollForSourceChanges` dans votre définition de pipeline, ajoutez ce paramètre et définissez-le sur `false`.

Pourquoi est-ce que j'effectue cette modification ? Le remplacement de la valeur de ce paramètre par `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

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

Lorsque vous créez ces ressources avec CloudFormation, votre pipeline est déclenché lorsque des fichiers de votre référentiel sont créés ou mis à jour. Voici un extrait du modèle 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
                        }
                    ]
                },

...
```

Migrer les pipelines de sondage avec une source S3 activée pour les événements

Pour un pipeline avec une source Amazon S3, modifiez le pipeline afin que la détection des modifications soit automatisée via EventBridge et avec un compartiment source activé pour les notifications d'événements. Il s'agit de la méthode recommandée si vous utilisez l'interface de ligne de commande ou si vous CloudFormation souhaitez migrer votre pipeline.

> **Note**
>
> Cela inclut l'utilisation d'un compartiment activé pour les notifications d'événements, dans lequel vous n'avez pas besoin de créer un historique CloudTrail distinct. Si vous utilisez la console, une règle d'événement et CloudTrail un suivi sont configurés pour vous. Pour ces étapes, voirMigrer les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3.

- CLI : Migrer les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3 (CLI)
- CloudFormation: Migrez les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3 (CloudFormation modèle)

Migrer les pipelines de sondage avec une source S3 activée pour les événements (CLI)

Suivez ces étapes pour modifier un pipeline qui utilise un sondage (contrôles périodiques) afin d'y utiliser un événement à la EventBridge place. Si vous voulez créer un pipeline, consultez Création d'un pipeline, d'étapes et d'actions.

Pour créer un pipeline piloté par les événements avec Amazon S3, vous modifiez le `PollForSourceChanges` paramètre de votre pipeline, puis vous créez les ressources suivantes :

- EventBridge règle de l'événement
- Rôle IAM pour permettre à l' EventBridge événement de démarrer votre pipeline

Pour créer une EventBridge règle avec Amazon S3 comme source et CodePipeline comme cible de l'événement et appliquer la politique d'autorisations

1. Accordez les autorisations nécessaires EventBridge CodePipeline pour invoquer la règle. Pour plus d'informations, consultez la section Utilisation de politiques basées sur les ressources pour Amazon. EventBridge

   a. Utilisez l'exemple suivant pour créer la stratégie d'approbation qui permet à EventBridge d'assumer le rôle de service. Nommez-la `trustpolicyforEB.json`.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Utilisez la commande suivante pour créer le rôle `Role-for-MyRule` et attachez la stratégie d'approbation.

Pourquoi est-ce que j'effectue cette modification ? L'ajout de cette stratégie d'approbation au rôle crée des autorisations pour EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Créez le JSON de stratégie d'autorisations, comme décrit ici, pour le pipeline nommé `MyFirstPipeline`. Nommez la stratégie d'autorisations `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilisez la commande suivante pour attacher la nouvelle stratégie d'autorisations `CodePipeline-Permissions-Policy-for-EB` au rôle `Role-for-MyRule` que vous avez créé.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Appelez la commande put-rule et incluez les paramètres `--name`, `--event-pattern` et `--role-arn`.

L'exemple de commande suivant crée une règle nommée `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. Pour ajouter CodePipeline en tant que cible, appelez la put-targets commande et incluez les `--targets` paramètres `--rule` et.

   La commande suivante spécifie que pour la règle nommée `EnabledS3SourceRule`, l'`Id` cible est composé du numéro un, ce qui indique qu'il s'agit de la règle 1 dans une liste de cibles pour la règle. La commande spécifie également un exemple d'ARN pour le pipeline. Le pipeline démarre lorsque des modifications sont effectuées dans le référentiel.

   ```
   aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-
   AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

1. Exécutez la commande get-pipeline pour copier la structure de pipeline dans un fichier JSON. Par exemple, pour un pipeline nommé `MyFirstPipeline`, exécutez la commande suivante :

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Cette commande ne renvoie rien, mais le fichier que vous avez créé doit apparaître dans le répertoire où vous avez exécuté la commande.

2. Ouvrez le fichier JSON dans un éditeur de texte brut et modifiez l'étape source en remplaçant la valeur du paramètre `PollForSourceChanges` pour un compartiment nommé `amzn-s3-demo-source-bucket` par `false`, comme illustré dans cet exemple.

   Pourquoi est-ce que j'effectue cette modification ? La définition de ce paramètre sur `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Si vous utilisez la structure de pipeline extraite à l'aide de la commande get-pipeline, vous devez supprimer les lignes `metadata` du fichier JSON. Sinon, la commande update-pipeline ne peut pas l'utiliser. Supprimez les lignes `"metadata": { }` et les champs `"updated"`, `"created"` et `"pipelineARN"`.

   Par exemple, supprimez les lignes suivantes de la structure :

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

   Enregistrez le fichier.

4. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline :

   > **Important**
   >
   > N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

Cette commande affiche toute la structure du pipeline mise à jour.

> **Note**
>
> La commande update-pipeline interrompt le pipeline. Si une révision est exécutée dans le pipeline lorsque vous exécutez la commande update-pipeline celle-ci est interrompue. Vous devez lancer manuellement le pipeline pour exécuter cette révision dans le pipeline mis à jour. Utilisez la commande start-pipeline-execution pour démarrer manuellement votre pipeline.

Migrez les pipelines de sondage avec une source S3 activée pour les événements (CloudFormation modèle)

Cette procédure concerne un pipeline dans lequel les événements sont activés dans le compartiment source.

Suivez ces étapes pour modifier votre pipeline à l'aide d'une source Amazon S3, du sondage à la détection des modifications basée sur les événements.

Pour créer un pipeline piloté par les événements avec Amazon S3, vous modifiez le `PollForSourceChanges` paramètre de votre pipeline, puis vous ajoutez les ressources suivantes à votre modèle :

- EventBridge règle et rôle IAM pour permettre à cet événement de démarrer votre pipeline.

Si vous l'utilisez CloudFormation pour créer et gérer vos pipelines, votre modèle inclut un contenu tel que le suivant.

> **Note**
>
> La propriété `Configuration` de l'étape source appelée `PollForSourceChanges`. Si votre modèle ne comprend pas cette propriété, `PollForSourceChanges` est défini sur `true` par défaut.

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
                        }
                    ]
                },


...
```

Pour créer une EventBridge règle avec Amazon S3 comme source et CodePipeline comme cible de l'événement et appliquer la politique d'autorisations

1. Dans le modèle, ci-dessous`Resources`, utilisez la `AWS::IAM::Role` CloudFormation ressource pour configurer le rôle IAM qui permet à votre événement de démarrer votre pipeline. Cette entrée crée un rôle qui utilise deux stratégies :

   - La première stratégie autorise le rôle à être endossé.
   - La deuxième stratégie fournit des autorisations pour démarrer le pipeline.

Pourquoi est-ce que j'effectue cette modification ? L'ajout d'une `AWS::IAM::Role` ressource CloudFormation permet de créer des autorisations pour EventBridge. Cette ressource est ajoutée à votre CloudFormation pile.

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. Utilisez la `AWS::Events::Rule` CloudFormation ressource pour ajouter une EventBridge règle. Ce modèle d'événement crée un événement qui surveille la création ou la suppression d'objets dans votre compartiment source Amazon S3. En outre, incluez une cible de votre pipeline. Lorsqu'un objet est créé, cette règle est invoquée `StartPipelineExecution` sur votre pipeline cible.

   Pourquoi est-ce que j'effectue cette modification ? L'ajout de la `AWS::Events::Rule` ressource CloudFormation permet de créer l'événement. Cette ressource est ajoutée à votre CloudFormation pile.

   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. Enregistrez le modèle mis à jour sur votre ordinateur local, puis ouvrez la console CloudFormation .
4. Choisissez votre pile, puis Créer un jeu de modifications pour la pile actuelle.
5. Chargez votre modèle mis à jour, puis affichez les modifications répertoriées dans CloudFormation. Il s'agit des modifications qui seront apportées à la pile. Vos nouvelles ressources doivent figurer dans la liste.
6. Sélectionnez Execute (Exécuter).

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

- Dans le modèle, remplacez la valeur du paramètre `PollForSourceChanges` par `false`. Si vous n'avez pas inclus `PollForSourceChanges` dans votre définition de pipeline, ajoutez ce paramètre et définissez-le sur `false`.

  Pourquoi est-ce que j'effectue cette modification ? Le remplacement de la valeur du paramètre `PollForSourceChanges` par `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

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

Lorsque vous créez CloudFormation ces ressources, votre pipeline est déclenché lorsque les fichiers de votre référentiel sont créés ou mis à jour.

> **Note**
>
> Ne vous arrêtez pas là. Bien que votre pipeline soit créé, vous devez créer un deuxième CloudFormation modèle pour votre pipeline Amazon S3. Si vous ne créez pas le second modèle, votre pipeline ne dispose d'aucune fonctionnalité de détection des modifications.

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"
        }
    ]
}
```

Migrer les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3

Pour un pipeline avec une source Amazon S3, modifiez le pipeline afin que la détection des modifications soit automatisée EventBridge. Choisissez l'une des méthodes suivantes pour implémenter la migration :

- Console : Migrer les pipelines de sondage (CodeCommit ou la source Amazon S3) (console)
- CLI : Migrer les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3 (CLI)
- CloudFormation: Migrez les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3 (CloudFormation modèle)

Migrer les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3 (CLI)

Suivez ces étapes pour modifier un pipeline qui utilise un sondage (contrôles périodiques) afin d'y utiliser un événement à la EventBridge place. Si vous voulez créer un pipeline, consultez Création d'un pipeline, d'étapes et d'actions.

Pour créer un pipeline piloté par les événements avec Amazon S3, vous modifiez le `PollForSourceChanges` paramètre de votre pipeline, puis vous créez les ressources suivantes :

- AWS CloudTrail politique de suivi, de compartiment et de compartiment qu'Amazon S3 peut utiliser pour enregistrer les événements.
- EventBridge événement
- Rôle IAM pour permettre à l' EventBridge événement de démarrer votre pipeline

Pour créer un AWS CloudTrail suivre et activer la journalisation

Pour utiliser le AWS CLI pour créer une trace, appelez la create-trail commande en spécifiant :

- Le nom du journal de suivi.

- Le compartiment auquel vous avez déjà appliqué la stratégie de compartiment pour AWS CloudTrail.

Pour plus d'informations, consultez la section Création d'un historique à l'aide de l'interface de ligne de AWS commande.

1. Appelez la commande create-trail et incluez les paramètres `--name` et `--s3-bucket-name`.

   Pourquoi est-ce que j'effectue cette modification ? Cela crée le journal de suivi CloudTrail requis pour votre compartiment source S3.

   La commande suivante utilise `--name` et `--s3-bucket-name` pour créer un journal de suivi nommé `my-trail` et un compartiment nommé `amzn-s3-demo-source-bucket`.

   ```
   aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-source-bucket
   ```

2. Appelez la commande start-logging et incluez le paramètre `--name`.

   Pourquoi est-ce que je fais ce changement ? Cette commande lance la CloudTrail journalisation de votre compartiment source et envoie les événements à EventBridge.

   Exemple :

   La commande suivante utilise `--name` pour démarrer la journalisation dans un journal de suivi nommé `my-trail`.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Appelez la commande put-event-selectors et incluez les paramètres `--trail-name` et `--event-selectors`. Utilisez des sélecteurs d'événements pour spécifier que vous souhaitez que votre journal enregistre les événements de données pour votre compartiment source et envoie les événements à la EventBridge règle.

   Pourquoi est-ce que je fais ce changement ? Cette commande filtre les événements.

   Exemple :

Dans l'exemple suivant, la commande utilise `--trail-name` et `--event-selectors` pour spécifier des événements de données pour un compartiment source et un préfixe nommés `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"] }] }]'
```

Pour créer une EventBridge règle avec Amazon S3 comme source et CodePipeline comme cible de l'événement et appliquer la politique d'autorisations

1. Accordez les autorisations nécessaires EventBridge CodePipeline pour invoquer la règle. Pour plus d'informations, consultez la section Utilisation de politiques basées sur les ressources pour Amazon. EventBridge

   a. Utilisez l'exemple suivant pour créer la politique de confiance permettant EventBridge d'assumer le rôle de service. Nommez-la `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Utilisez la commande suivante pour créer le rôle `Role-for-MyRule` et attachez la stratégie d'approbation.

      Pourquoi est-ce que j'effectue cette modification ? L'ajout de cette politique de confiance au rôle crée des autorisations pour EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Créez le JSON de stratégie d'autorisations, comme décrit ici, pour le pipeline nommé `MyFirstPipeline`. Nommez la stratégie d'autorisations `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilisez la commande suivante pour attacher la nouvelle stratégie d'autorisations `CodePipeline-Permissions-Policy-for-EB` au rôle `Role-for-MyRule` que vous avez créé.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Appelez la commande put-rule et incluez les paramètres `--name`, `--event-pattern` et `--role-arn`.

L'exemple de commande suivant crée une règle nommée `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. Pour ajouter CodePipeline en tant que cible, appelez la put-targets commande et incluez les `--targets` paramètres `--rule` et.

   La commande suivante spécifie que pour la règle nommée `MyS3SourceRule`, l'`Id` cible est composé du numéro un, ce qui indique qu'il s'agit de la règle 1 dans une liste de cibles pour la règle. La commande spécifie également un exemple d'ARN pour le pipeline. Le pipeline démarre lorsque des modifications sont effectuées dans le référentiel.

   ```
   aws events put-targets --rule MyS3SourceRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

4. (Facultatif) Pour configurer un transformateur d'entrée avec des remplacements de source pour un ID d'image spécifique, utilisez le JSON suivant dans votre commande CLI. L'exemple suivant configure une dérogation dans laquelle :

   - `Source`Dans cet exemple`actionName`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - `S3_OBJECT_VERSION_ID`Dans cet exemple`revisionType`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - Dans cet exemple`revisionValue`, < *`revisionValue`* > est dérivé de la variable d'événement source.

   ```
   {
       "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>"
                    }
                }
            }
        }
    ]
}
```

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

1. Exécutez la commande get-pipeline pour copier la structure de pipeline dans un fichier JSON. Par exemple, pour un pipeline nommé `MyFirstPipeline`, exécutez la commande suivante :

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Cette commande ne renvoie rien, mais le fichier que vous avez créé doit apparaître dans le répertoire où vous avez exécuté la commande.

2. Ouvrez le fichier JSON dans un éditeur de texte brut et modifiez l'étape source en remplaçant la valeur du paramètre `PollForSourceChanges` pour un compartiment nommé `amzn-s3-demo-source-bucket` par `false`, comme illustré dans cet exemple.

   Pourquoi est-ce que j'effectue cette modification ? La définition de ce paramètre sur `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
   ```

```
    "S3ObjectKey": "index.zip"
},
```

3. Si vous utilisez la structure de pipeline extraite à l'aide de la commande get-pipeline, vous devez supprimer les lignes `metadata` du fichier JSON. Sinon, la commande update-pipeline ne peut pas l'utiliser. Supprimez les lignes `"metadata": { }` et les champs `"updated"`, `"created"` et `"pipelineARN"`.

   Par exemple, supprimez les lignes suivantes de la structure :

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

   Enregistrez le fichier.

4. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline :

   > **Important**
   >
   > N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

   Cette commande affiche toute la structure du pipeline mise à jour.

   > **Note**
   >
   > La commande update-pipeline interrompt le pipeline. Si une révision est exécutée dans le pipeline lorsque vous exécutez la commande update-pipeline celle-ci est interrompue. Vous devez lancer manuellement le pipeline pour exécuter cette révision dans le pipeline mis à jour. Utilisez la commande start-pipeline-execution pour démarrer manuellement votre pipeline.

Migrez les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3 (CloudFormation modèle)

Suivez ces étapes pour modifier votre pipeline à l'aide d'une source Amazon S3, du sondage à la détection des modifications basée sur les événements.

Pour créer un pipeline piloté par les événements avec Amazon S3, vous modifiez le `PollForSourceChanges` paramètre de votre pipeline, puis vous ajoutez les ressources suivantes à votre modèle :

- EventBridge exige que tous les événements Amazon S3 soient enregistrés. Vous devez créer une politique AWS CloudTrail de suivi, de compartiment et de compartiment qu'Amazon S3 peut utiliser pour enregistrer les événements qui se produisent. Pour plus d'informations, consultez les sections Enregistrement des événements de données pour les sentiers et Enregistrement des événements de gestion pour les sentiers.
- EventBridge règle et rôle IAM pour permettre à cet événement de démarrer votre pipeline.

Si vous l'utilisez CloudFormation pour créer et gérer vos pipelines, votre modèle inclut un contenu tel que le suivant.

> **Note**
>
> La propriété `Configuration` de l'étape source appelée `PollForSourceChanges`. Si votre modèle ne comprend pas cette propriété, `PollForSourceChanges` est défini sur `true` par défaut.

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
                        }
                    ]
                },


...
```

Pour créer une EventBridge règle avec Amazon S3 comme source et CodePipeline comme cible de l'événement et appliquer la politique d'autorisations

1. Dans le modèle, ci-dessous`Resources`, utilisez la `AWS::IAM::Role` CloudFormation ressource pour configurer le rôle IAM qui permet à votre événement de démarrer votre pipeline. Cette entrée crée un rôle qui utilise deux stratégies :

   - La première stratégie autorise le rôle à être endossé.
   - La deuxième stratégie fournit des autorisations pour démarrer le pipeline.

   Pourquoi est-ce que j'effectue cette modification ? L'ajout d'une `AWS::IAM::Role` ressource CloudFormation permet de créer des autorisations pour EventBridge. Cette ressource est ajoutée à votre CloudFormation pile.

   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. Utilisez la `AWS::Events::Rule` CloudFormation ressource pour ajouter une EventBridge règle. Ce modèle d'événement crée un événement qui surveille `CopyObject` `PutObject` et `CompleteMultipartUpload` affecte votre compartiment source Amazon S3.
En outre, incluez une cible de votre pipeline. Lorsque `CopyObject`, `PutObject` ou `CompleteMultipartUpload` se produit, cette règle appelle `StartPipelineExecution` sur votre pipeline cible.

   Pourquoi est-ce que j'effectue cette modification ? L'ajout de la `AWS::Events::Rule` ressource CloudFormation permet de créer l'événement. Cette ressource est ajoutée à votre CloudFormation pile.

   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. Ajoutez cet extrait à votre premier modèle pour autoriser les fonctionnalités entre piles :

   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. (Facultatif) Pour configurer un transformateur d'entrée avec des remplacements de source pour un ID d'image spécifique, utilisez l'extrait YAML suivant. L'exemple suivant configure une dérogation dans laquelle :

- `SourceDans cet exemple``actionName`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
- `S3_OBJECT_VERSION_IDDans cet exemple``revisionType`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
- Dans cet exemple`revisionValue`, **<** *revisionValue* **>** est dérivé de la variable d'événement source.

```
---
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. Enregistrez le modèle mis à jour sur votre ordinateur local et ouvrez la CloudFormation console.
6. Choisissez votre pile, puis Créer un jeu de modifications pour la pile actuelle.
7. Chargez votre modèle mis à jour, puis affichez les modifications répertoriées dans CloudFormation. Il s'agit des modifications qui seront apportées à la pile. Vos nouvelles ressources doivent figurer dans la liste.
8. Sélectionnez Execute (Exécuter).

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon,

votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

- Dans le modèle, remplacez la valeur du paramètre `PollForSourceChanges` par `false`. Si vous n'avez pas inclus `PollForSourceChanges` dans votre définition de pipeline, ajoutez ce paramètre et définissez-le sur `false`.

  Pourquoi est-ce que j'effectue cette modification ? Le remplacement de la valeur du paramètre `PollForSourceChanges` par `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

  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
  }
```

Pour créer un deuxième modèle pour les CloudTrail ressources de votre pipeline Amazon S3

- Dans un modèle distinct, ci-dessous`Resources`, utilisez les `AWS::CloudTrail::Trail` CloudFormation ressources `AWS::S3::Bucket``AWS::S3::BucketPolicy`, et pour fournir une définition et un suivi simples du compartiment CloudTrail.

  Pourquoi est-ce que je fais ce changement ? Compte tenu de la limite actuelle de cinq pistes par compte, la CloudTrail piste doit être créée et gérée séparément. (Voir Limites dans AWS CloudTrail.) Cependant, vous pouvez inclure de nombreux compartiments Amazon S3 sur un seul trail, de sorte que vous puissiez créer le trail une seule fois, puis ajouter des compartiments Amazon S3 pour d'autres pipelines si nécessaire. Collez ce qui suit dans votre deuxième exemple de fichier de modèle.

  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

Lorsque vous créez CloudFormation ces ressources, votre pipeline est déclenché lorsque les fichiers de votre référentiel sont créés ou mis à jour.

> **Note**
>
> Ne vous arrêtez pas là. Bien que votre pipeline soit créé, vous devez créer un deuxième CloudFormation modèle pour votre pipeline Amazon S3. Si vous ne créez pas le second modèle, votre pipeline ne dispose d'aucune fonctionnalité de détection des modifications.

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"
            }
        }
    }
}


...
```

Migrer les pipelines de sondage pour une action source GitHub (via l'application OAuth) vers les connexions

Vous pouvez migrer une action source GitHub (via l'application OAuth) pour utiliser des connexions pour votre référentiel externe. Il s'agit de la méthode de détection des modifications recommandée pour les pipelines comportant une GitHub action source (via l'application OAuth).

Pour un pipeline avec une action source GitHub (via l'application OAuth), nous vous recommandons de modifier le pipeline pour utiliser une action GitHub (via GitHub l'application) afin que la détection des modifications soit automatisée. AWS CodeConnections Pour plus d'informations sur l'utilisation des connexions, consultezGitHub connexions.

Créer une connexion à GitHub (console)

Vous pouvez utiliser la console pour créer une connexion à GitHub.

Étape 1 : remplacez votre action GitHub (via l'application OAuth)

Utilisez la page de modification du pipeline pour remplacer votre action GitHub (via l'application OAuth) par une action GitHub (via GitHub l'application).

Pour remplacer votre action GitHub (via l'application OAuth)

1. Connectez-vous à la CodePipeline console.
2. Choisissez votre pipeline, puis sélectionnez Modifier. Choisissez Modifier l'étape sur votre scène source. Un message s'affiche pour vous recommander de mettre à jour votre action.
3. Dans le fournisseur d'actions, choisissez GitHub (via GitHub l'application).
4. Effectuez l'une des actions suivantes :

   - Sous Connexion, si vous n'avez pas encore créé de connexion avec votre fournisseur, choisissez Se connecter à GitHub. Passez à l'étape 2 : créez une connexion avec GitHub.
   - Sous Connexion, si vous avez déjà créé une connexion avec votre fournisseur, choisissez-la. Passez à l'étape 3 : enregistrez l'action source pour votre connexion.

Étape 2 : Création d'une connexion à GitHub

Une fois que vous avez choisi de créer la connexion, la GitHub page Se connecter à s'affiche.

Pour créer une connexion avec GitHub

1. Dans les paramètres de GitHub connexion, le nom de votre connexion apparaît dans Nom de la connexion.

   Sous GitHub Applications, choisissez l'installation d'une application ou choisissez Installer une nouvelle application pour en créer une.

   > **Note**
   >
   > Installez une application pour toutes vos connexions à un fournisseur particulier. Si vous avez déjà installé l' GitHub application, choisissez-la et ignorez cette étape.

2. Si la page d'autorisation GitHub s'affiche, connectez-vous avec vos informations d'identification, puis choisissez de continuer.
3. Sur la page d'installation de l'application, un message indique que l' AWS CodeStar application essaie de se connecter à votre GitHub compte.

> **Note**
>
> Vous n'installez l'application qu'une seule fois pour chaque GitHub compte. Si vous avez déjà installé l'application, vous pouvez choisir Configure (Configurer) pour passer à une page de modification pour l'installation de votre application, ou vous pouvez utiliser le bouton Précédent pour revenir à la console.

4. Sur la AWS CodeStar page Installer, choisissez Installer.
5. Sur la GitHub page Se connecter à, l'ID de connexion de votre nouvelle installation s'affiche. Choisissez Se connecter.

Étape 3 : Enregistrez votre action GitHub source

Effectuez vos mises à jour sur la page Modifier l'action pour enregistrer votre nouvelle action source.

Pour enregistrer votre action GitHub source

1. Dans Référentiel, entrez le nom de votre référentiel tiers. Dans Branche, entrez la branche dans laquelle vous souhaitez que votre pipeline détecte les changements de source.

> **Note**
>
> Dans Repository, tapez `owner-name/repository-name` comme indiqué dans cet exemple :
>
> ```
> my-account/my-repository
> ```

2. Dans Format d'artefact de sortie, choisissez le format de vos artefacts.
   - Pour stocker les artefacts de sortie de l' GitHub action à l'aide de la méthode par défaut, choisissez CodePipeline default. L'action accède aux fichiers depuis le GitHub référentiel et stocke les artefacts dans un fichier ZIP dans le magasin d'artefacts du pipeline.
   - Pour stocker un fichier JSON contenant une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git, choisissez Full clone (Clone complet). Cette option ne peut être utilisée que par des actions CodeBuild en aval.

Si vous choisissez cette option, vous devrez mettre à jour les autorisations pour votre rôle de service de CodeBuild projet, comme indiqué dansAjoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com. Pour un didacticiel qui explique comment utiliser l'option de clonage complet, consultezTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

3. Dans Artefacts de sortie, vous pouvez conserver le nom de l'artefact de sortie pour cette action, par exemple`SourceArtifact`. Choisissez Terminé pour fermer la page Modifier l'action.
4. Choisissez Terminé pour fermer la page d'édition de la scène. Choisissez Enregistrer pour fermer la page de modification du pipeline.

Créer une connexion à GitHub (CLI)

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour créer une connexion à GitHub.

Pour ce faire, utilisez la commande create-connection.

> **Important**
>
> Une connexion créée via le AWS CLI ou AWS CloudFormation est en `PENDING` statut par défaut. Après avoir créé une connexion à l'aide de l'interface de ligne de commande CloudFormation, utilisez la console pour modifier la connexion afin de définir son état`AVAILABLE`.

Pour créer une connexion avec GitHub

1. Ouvrez une invite de terminal (Linux, macOS ou Unix) ou de commande (Windows). Utilisez le AWS CLI pour exécuter la create-connection commande, en spécifiant le `--provider-type` et `--connection-name` pour votre connexion. Dans cet exemple, le nom du fournisseur tiers est `GitHub` et le nom de connexion spécifié est `MyConnection`.

   ```
   aws codeconnections create-connection --provider-type GitHub --connection-name
    MyConnection
   ```

   En cas de succès, cette commande renvoie les informations ARN de connexion semblables à ce qui suit.

```
{
    "ConnectionArn": "arn:aws:codeconnections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Utilisez la console pour terminer la connexion.

Migrer les pipelines de sondage pour une action source GitHub (via l'application OAuth) vers des webhooks

Vous pouvez migrer votre pipeline pour utiliser des webhooks afin de détecter les modifications apportées à votre référentiel GitHub source. Cette migration vers les webhooks est réservée à l'action GitHub (via l'application OAuth).

- Console : Migrer les pipelines de sondage vers les webhooks GitHub (via l'application OAuth) (actions source) (console)
- CLI : Migrer les pipelines de sondage vers les webhooks GitHub (via l'application OAuth) (actions source) (CLI)
- CloudFormation: Mettre à jour les pipelines pour les actions source des événements push GitHub (via l'application OAuth)) (CloudFormation modèle)

> **Important**
>
> Lorsque vous créez des CodePipeline webhooks, n'utilisez pas vos propres informations d'identification et ne réutilisez pas le même jeton secret sur plusieurs webhooks. Pour une sécurité optimale, générez un jeton secret unique pour chaque webhook que vous créez. Le jeton secret est une chaîne arbitraire que vous fournissez, qui est GitHub utilisée pour calculer et signer les charges utiles du webhook envoyées CodePipeline, afin de protéger l'intégrité et l'authenticité des charges utiles du webhook. L'utilisation de vos propres informations d'identification ou la réutilisation du même jeton sur plusieurs webhooks peut entraîner des failles de sécurité.

Migrer les pipelines de sondage vers les webhooks GitHub (via l'application OAuth) (actions source) (console)

Pour l'action source GitHub (via l'application OAuth), vous pouvez utiliser la CodePipeline console pour mettre à jour votre pipeline afin d'utiliser des webhooks pour détecter les modifications dans votre GitHub référentiel source.

Suivez ces étapes pour modifier un pipeline qui utilise le sondage (contrôles périodiques) à utiliser à la EventBridge place. Si vous voulez créer un pipeline, consultez Création d'un pipeline, d'étapes et d'actions.

Lorsque vous utilisez la console, le paramètre `PollForSourceChanges` de votre pipeline est modifié pour vous. Le GitHub webhook est créé et enregistré pour vous.

Pour modifier l'étape source du pipeline

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés.
2. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier. Une vue détaillée du pipeline s'affiche alors, laquelle indique notamment l'état de chaque action, dans chaque étape du pipeline.
3. Sur la page des détails du pipeline, choisissez Edit.
4. À l'étape Modifier, choisissez l'icône de modification sur l'action source.
5. Développez les options de détection des changements et choisissez Utiliser Amazon CloudWatch Events pour démarrer automatiquement mon pipeline lorsqu'un changement se produit (recommandé).

   Un message s'affiche pour indiquer que vous CodePipeline créez un webhook GitHub pour détecter les changements de source : AWS CodePipeline créera un webhook pour vous. Vous pouvez vous désinscrire dans les options ci-dessous. Choisissez Mettre à jour. Outre le webhook, CodePipeline crée les éléments suivants :

   - Un secret généré de manière aléatoire et utilisé pour autoriser la connexion à GitHub.
   - L'URL du webhook, générée à l'aide du point de terminaison public pour la région.

CodePipeline enregistre le webhook auprès GitHub de. Cette action abonne l'URL à la réception des événements du référentiel.

6. Lorsque vous avez terminé de modifier votre pipeline, cliquez sur Save pipeline changes pour revenir à la page récapitulative.

   Un message affiche le nom du webhook à créer pour votre pipeline. Choisissez Save and continue (Enregistrer et continuer).

7. Pour tester votre action, publiez une modification en utilisant le AWS CLI pour valider une modification de la source spécifiée dans l'étape source du pipeline.

Migrer les pipelines de sondage vers les webhooks GitHub (via l'application OAuth) (actions source) (CLI)

Suivez ces étapes pour modifier un pipeline qui utilise les vérifications périodiques afin d'utiliser plutôt un webhook. Si vous voulez créer un pipeline, consultez Création d'un pipeline, d'étapes et d'actions.

Pour créer un pipeline basé sur les événements, vous devez modifier le paramètre `PollForSourceChanges` de votre pipeline, puis créer manuellement les ressources suivantes :

- GitHub paramètres de webhook et d'autorisation

Pour créer et enregistrer votre webhook

> **Note**
>
> Lorsque vous utilisez la CLI ou CloudFormation pour créer un pipeline et ajouter un webhook, vous devez désactiver les contrôles périodiques. Pour désactiver les vérifications périodiques, vous devez ajouter explicitement le paramètre `PollForSourceChanges` et lui affectez la valeur Faux, comme indiqué dans la procédure finale ci-dessous. Sinon, la valeur par défaut d'une CLI ou d'un CloudFormation pipeline est true et ne s'affiche pas dans la sortie de la structure du pipeline. `PollForSourceChanges` Pour plus d'informations sur les PollForSourceChanges valeurs par défaut, consultezRéglages valides pour le `PollForSourceChanges` paramètre.

1. Dans un éditeur de texte, créez et enregistrez un fichier JSON pour le webhook que vous souhaitez créer. Utilisez cet exemple de fichier pour un webhook nommé `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. Appelez la commande put-webhook et incluez les paramètres `--cli-input` et `--region`.

   L'exemple de commande suivant crée un webhook avec le fichier JSON `webhook_json`.

```
aws codepipeline put-webhook --cli-input-json file://webhook_json.json --region
 "eu-central-1"
```

3. Dans la sortie illustrée dans cet exemple, l'URL et l'ARN sont renvoyés pour un webhook nommé `my-webhook`.

```
{
    "webhook": {
        "url": "https://webhooks.domain.com/
trigger111111111EXAMPLE1111111111111111",
        "definition": {
            "authenticationConfiguration": {
                "SecretToken": "secret"
            },
            "name": "my-webhook",
            "authentication": "GITHUB_HMAC",
            "targetPipeline": "pipeline_name",
            "targetAction": "Source",
            "filters": [
                {
```

```
                    "jsonPath": "$.ref",
                    "matchEquals": "refs/heads/{Branch}"
                }
            ]
        },
        "arn": "arn:aws:codepipeline:eu-central-1:ACCOUNT_ID:webhook:my-webhook"
    },
    "tags": [{
      "key": "Project",
      "value": "ProjectA"
    }]
}
```

Cet exemple ajoute le balisage du webhook en incluant la clé de balise `Project` et la valeur `ProjectA` sur le webhook. Pour plus d'informations sur le balisage des ressources CodePipeline, consultezBalisage des ressources.

4. Appelez la commande register-webhook-with-third-party et incluez le paramètre `--webhook-name`.

   L'exemple de commande suivant enregistre un webhook nommé `my-webhook`.

   ```
   aws codepipeline register-webhook-with-third-party --webhook-name my-webhook
   ```

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

1. Exécutez la commande get-pipeline pour copier la structure de pipeline dans un fichier JSON. Par exemple, pour un pipeline nommé `MyFirstPipeline`, saisissez la commande suivante :

```
aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
```

Cette commande ne renvoie rien, mais le fichier que vous avez créé doit apparaître dans le répertoire où vous avez exécuté la commande.

2. Ouvrez le fichier JSON dans un éditeur de texte brut et modifiez l'étape source en modifiant ou en ajoutant le paramètre `PollForSourceChanges`. Dans cet exemple, pour un référentiel nommé `UserGitHubRepo`, le paramètre est défini sur `false`.

   Pourquoi est-ce que je fais ce changement ? La modification de ce paramètre désactive les contrôles périodiques. Vous ne pouvez donc utiliser que la détection des modifications basée sur les événements.

   ```
   "configuration": {
       "Owner": "name",
       "Repo": "UserGitHubRepo",
       "PollForSourceChanges": "false",
       "Branch": "main",
       "OAuthToken": "****"
   },
   ```

3. Si vous utilisez la structure de pipeline récupérée à l'aide de la commande get-pipeline, vous devez modifier cette structure dans le fichier JSON en supprimant les lignes `metadata` du fichier. Sinon, la commande update-pipeline ne peut pas l'utiliser. Supprimez la section `"metadata"` de la structure de pipeline dans le fichier JSON, y compris les champs `{ }`, `"created"`, `"pipelineARN"` et `"updated"`.

   Par exemple, supprimez les lignes suivantes de la structure :

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

   Enregistrez le fichier.

4. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline d'une manière similaire à l'exemple suivant :

> **Important**
>
> N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

Cette commande affiche toute la structure du pipeline mise à jour.

> **Note**
>
> La commande update-pipeline interrompt le pipeline. Si une révision est exécutée dans le pipeline lorsque vous exécutez la commande update-pipeline celle-ci est interrompue. Vous devez lancer manuellement le pipeline pour exécuter cette révision dans le pipeline mis à jour. Utilisez la commande start-pipeline-execution pour démarrer manuellement votre pipeline.

Mettre à jour les pipelines pour les actions source des événements push GitHub (via l'application OAuth)) (CloudFormation modèle)

Suivez ces étapes pour mettre à jour votre pipeline (avec une GitHub source), depuis les contrôles périodiques (sondages) jusqu'à la détection des changements basée sur les événements à l'aide de webhooks.

Pour créer un pipeline piloté par des événements avec AWS CodeCommit, vous modifiez le `PollForSourceChanges` paramètre de votre pipeline, puis vous ajoutez une ressource GitHub Webhook à votre modèle.

Si vous l'utilisez CloudFormation pour créer et gérer vos pipelines, le contenu de votre modèle est le suivant.

> **Note**
>
> Notez la propriété de configuration `PollForSourceChanges` de l'étape source. Si votre modèle ne comprend pas cette propriété, `PollForSourceChanges` est défini sur `true` par défaut.

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
```

```
                        }
                    ]
                },


...
```

Pour ajouter des paramètres et créer un webhook dans votre modèle

Nous vous recommandons vivement de l'utiliser AWS Secrets Manager pour stocker vos informations d'identification. Si vous utilisez Secrets Manager, vous devez avoir déjà configuré et stocké vos paramètres secrets dans Secrets Manager. Cet exemple utilise des références dynamiques à Secrets Manager pour les GitHub informations d'identification de votre webhook. Pour de plus amples informations, veuillez consulter Utilisation de références dynamiques pour spécifier des valeurs de modèle.

> **Important**
>
> Lorsque vous transmettez des paramètres de secret, n'entrez pas la valeur directement dans le modèle. La valeur est rendue en texte brut ; elle est donc lisible. Pour des raisons de sécurité, n'utilisez pas de texte en clair dans votre CloudFormation modèle pour stocker vos informations d'identification.

Lorsque vous utilisez la CLI ou CloudFormation pour créer un pipeline et ajouter un webhook, vous devez désactiver les contrôles périodiques.

> **Note**
>
> Pour désactiver les vérifications périodiques, vous devez ajouter explicitement le paramètre `PollForSourceChanges` et lui affectez la valeur Faux, comme indiqué dans la procédure finale ci-dessous. Sinon, la valeur par défaut d'une CLI ou d'un CloudFormation pipeline est true et ne s'affiche pas dans la sortie de la structure du pipeline. `PollForSourceChanges` Pour plus d'informations sur les PollForSourceChanges valeurs par défaut, consultezRéglages valides pour le `PollForSourceChanges` paramètre.

1. Dans le modèle, sous `Resources`, ajoutez vos paramètres :

YAML

```
Parameters:
        GitHubOwner:
          Type: String

...
```

JSON

```
{
 "Parameters": {
  "BranchName": {
   "Description": "GitHub branch name",
   "Type": "String",
   "Default": "main"
  },
  "GitHubOwner": {
   "Type": "String"
  },

...
```

2. Utilisez la `AWS::CodePipeline::Webhook` CloudFormation ressource pour ajouter un webhook.

> **Note**
>
> La valeur de `TargetAction` que vous spécifiez doit correspondre à la propriété `Name` de l'action source définie dans le pipeline.

S'`RegisterWithThirdParty`il est défini sur`true`, assurez-vous que l'utilisateur associé au `OAuthToken` peut définir les étendues requises. GitHub Le jeton et le webhook nécessitent les GitHub étendues suivantes :

- `repo` - utilisée pour contrôler entièrement la lecture et l'extraction des artefacts dans un pipeline à partir de référentiels publics et privés.
- `admin:repo_hook` - utilisée pour contrôler entièrement les hooks de référentiel.

Dans le cas contraire, GitHub renvoie un 404. Pour plus d'informations sur l'erreur 404 renvoyée, consultez 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. Enregistrez le modèle mis à jour sur votre ordinateur local, puis ouvrez la CloudFormation console.
4. Choisissez votre pile, puis Créer un jeu de modifications pour la pile actuelle.
5. Chargez le modèle mis à jour, puis affichez les modifications répertoriées dans CloudFormation. Il s'agit des modifications apportées à la pile. Vos nouvelles ressources doivent figurer dans la liste.
6. Sélectionnez Execute (Exécuter).

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

- Dans le modèle, remplacez la valeur du paramètre `PollForSourceChanges` par `false`. Si vous n'avez pas inclus `PollForSourceChanges` dans votre définition de pipeline, ajoutez ce paramètre et définissez-le sur false.

Pourquoi est-ce que je fais ce changement ? Le remplacement de la valeur de ce paramètre par `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

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

Lorsque vous créez ces ressources avec CloudFormation, le webhook défini est créé dans le GitHub référentiel spécifié. Votre pipeline est déclenché à la validation.

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

...
```

Création du rôle CodePipeline de service

Lorsque vous créez un pipeline, vous créez un rôle de service ou vous utilisez un rôle de service existant.

Vous pouvez utiliser la CodePipeline console ou le AWS CLI pour créer un rôle CodePipeline de service. Un rôle de service est nécessaire pour créer un pipeline et le pipeline est toujours associé à ce rôle de service.

Avant de créer un pipeline à l'aide de l' AWS interface de ligne de commande, vous devez créer un rôle de CodePipeline service pour votre pipeline. Pour un exemple de CloudFormation modèle avec le rôle de service et la politique spécifiés, consultez les didacticiels dansTutoriel : Création d'un pipeline qui utilise des variables provenant de AWS CloudFormation actions de déploiement.

Le rôle de service n'est pas un rôle AWS géré mais il est créé initialement pour la création du pipeline, puis à mesure que de nouvelles autorisations sont ajoutées à la politique de rôle de service, vous devrez peut-être mettre à jour le rôle de service pour votre pipeline. Une fois votre pipeline créé avec un rôle de service, vous ne pouvez pas appliquer un autre rôle du service à ce pipeline. Attachez la stratégie recommandée au rôle de service.

Pour plus d'informations sur le rôle de service, consultez Gérer le rôle CodePipeline de service.

Création du rôle CodePipeline de service (console)

Lorsque vous utilisez la console pour créer un pipeline, vous créez le rôle de CodePipeline service à l'aide de l'assistant de création de pipeline.

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Choisissez Créer un pipeline et appliquez la procédure de la page Étape 1 : Choisir des paramètres de pipeline dans l'assistant de création de pipeline.

> **Note**
>
> Une fois le pipeline créé, vous ne pouvez plus modifier son nom. Pour plus d'informations sur les autres limitations, consultez Quotas dans AWS CodePipeline.

2. Dans Rôle de service, choisissez Nouveau rôle de service pour CodePipeline permettre de créer un nouveau rôle de service dans IAM.
3. Finalisez la création du pipeline. Votre rôle de service de pipeline peut être affiché dans votre liste de rôles IAM, et vous pouvez consulter l'ARN du rôle de service associé à un pipeline en exécutant la commande à l'aide de l' AWS interface de ligne de `get-pipeline` commande.

Création du rôle CodePipeline de service (CLI)

Avant de créer un pipeline à l'aide de l' AWS interface de ligne de commande ou CloudFormation, vous devez créer un rôle de CodePipeline service pour votre pipeline et y associer la politique de rôle de service et la politique de confiance. Pour utiliser l'interface de ligne de commande afin de créer votre rôle de service, suivez les étapes ci-dessous pour créer d'abord une politique de confiance JSON et la politique de rôle JSON sous forme de fichiers distincts dans le répertoire où vous allez exécuter les commandes de l'interface de ligne de commande.

> **Note**
>
> Nous vous recommandons de n'autoriser que les utilisateurs administratifs à créer un rôle de service. Une personne disposant des autorisations nécessaires pour créer un rôle et attacher n'importe quelle politique peut augmenter ses propres autorisations. Au lieu de cela, créez une politique qui l'autorise à créer uniquement les rôles dont elle a besoin ou de demander à un administrateur de créer la fonction du service en son nom.

1. Dans une fenêtre de terminal, entrez la commande suivante pour créer un fichier nomméTrustPolicy.json, dans lequel vous allez coller la politique de rôle JSON. Cet exemple utilise VIM.

```
vim TrustPolicy.json
```

2. Collez le JSON suivant dans le fichier.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codepipeline.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Pour enregistrer et quitter le fichier, entrez la commande VIM suivante :

```
:wq
```

3. Dans une fenêtre de terminal, entrez la commande suivante pour créer un fichier nomméRolePolicy.json, dans lequel vous allez coller la politique de rôle JSON. Cet exemple utilise VIM.

```
vim RolePolicy.json
```

4. Collez la politique JSON dans le fichier. Utilisez la politique relative aux rôles de service minimum, comme indiqué dansCodePipeline politique relative aux rôles de service. En outre, ajoutez les autorisations appropriées à votre rôle de service en fonction des actions que vous prévoyez d'utiliser. Pour obtenir la liste des actions et un lien vers les autorisations de rôle de service requises pour chaque action, consultezAjouter des autorisations au rôle CodePipeline de service.

   Assurez-vous de limiter les autorisations autant que possible en les limitant au niveau des ressources Resource sur le terrain.

   Pour enregistrer et quitter le fichier, entrez la commande VIM suivante :

```
:wq
```

5. Entrez la commande suivante pour créer le rôle et y associer la politique de rôle de confiance. Le format du nom de la stratégie est normalement identique à celui du nom du rôle. Cet exemple utilise le nom du rôle `MyRole` et la politique `TrustPolicy` qui ont été créés dans un fichier distinct.

```
aws iam create-role --role-name MyRole --assume-role-policy-document file://
TrustPolicy.json
```

6. Entrez la commande suivante pour créer la politique de rôle et l'associer au rôle. Le format du nom de la stratégie est normalement identique à celui du nom du rôle. Cet exemple utilise le nom du rôle `MyRole` et la politique `MyRole` qui ont été créés dans un fichier distinct.

```
aws iam put-role-policy --role-name MyRole --policy-name RolePolicy --policy-
document file://RolePolicy.json
```

7. Pour afficher le nom du rôle et la politique de confiance créés, entrez la commande suivante pour le rôle nommé `MyRole` :

```
aws iam get-role --role-name MyRole
```

8. Utilisez l'ARN du rôle de service lorsque vous créez votre pipeline à l'aide de l' AWS interface de ligne de commande ou CloudFormation.

Balisage des ressources

Une étiquette est une étiquette d'attribut personnalisée que vous attribuez ou AWS attribuez à une AWS ressource. Chaque AWS étiquette comporte deux parties :

- Une clé de balise (par exemple, `CostCenter`, `Environment`, `Project` ou `Secret`). Les touches des tags distinguent les majuscules et minuscules.
- Un champ facultatif appelé valeur de balise (par exemple, `111122223333`, `Production` ou le nom d'une équipe). Omettre la valeur de balise équivaut à l'utilisation d'une chaîne vide. Les valeurs de balise sont sensibles à la casse, tout comme les clés de balise.

Ensemble, ces éléments sont connus sous le nom de paires clé-valeur.

Les balises vous aident à identifier et à organiser vos AWS ressources. Beaucoup Services AWS prennent en charge le balisage. Vous pouvez donc attribuer le même tag à des ressources provenant

de différents services pour indiquer que les ressources sont liées. Par exemple, vous pouvez attribuer la même balise à un pipeline que vous attribuez à un compartiment source Amazon S3.

Pour obtenir des conseils sur l'utilisation des balises, consultez l'article Stratégies de balisage AWS sur le blog AWS Answers.

Vous pouvez baliser les types de ressources suivants dans CodePipeline :

- Marquer un pipeline dans CodePipeline
- Marquer une action personnalisée dans CodePipeline

Vous pouvez utiliser les CodePipeline API ou AWS CLI les kits de AWS développement logiciel (SDK) pour :

- ajouter des balises à un pipeline, à une action personnalisée ou à un webhook lors de leur création ;
- ajouter, gérer et supprimer des balises pour un pipeline, une action personnalisée ou un webhook.

Vous pouvez également utiliser la console pour ajouter, gérer et supprimer des balises pour un pipeline.

Outre l'identification, l'organisation et le suivi de votre ressource à l'aide de balises, vous pouvez utiliser des balises dans les politiques IAM pour contrôler qui peut consulter votre ressource et interagir avec elle. Pour obtenir des exemples de stratégies d'accès basées sur les balises, consultez Utilisation de balises pour contrôler l'accès aux CodePipeline ressources.

Marquer un pipeline dans CodePipeline

Les balises sont des paires clé-valeur associées à des AWS ressources. Vous pouvez appliquer des balises à vos pipelines dans CodePipeline. Pour plus d'informations sur le balisage CodePipeline des ressources, les cas d'utilisation, les contraintes de clé et de valeur de balise et les types de ressources pris en charge, consultezBalisage des ressources.

Vous pouvez utiliser l'interface de ligne de commande pour spécifier des balises lorsque vous créez un pipeline. Vous pouvez utiliser la console ou l'interface de ligne de commande pour ajouter ou supprimer des balises ainsi que pour mettre à jour les valeurs des balises dans un pipeline. Vous pouvez ajouter jusqu'à 50 balises à chaque pipeline.

Rubriques

- Balisage de pipelines (console)
- Balisage de pipelines (interface de ligne de commande)

Balisage de pipelines (console)

Vous pouvez utiliser la console ou la CLI pour baliser des ressources. Les pipelines constituent la seule ressource CodePipeline pouvant être gérée tant à l'aide de la console que de l'interface de ligne de commande.

Rubriques

- Ajout de balises à un pipeline (console)
- Affichage des balises d'un pipeline (console)
- Modification des balises d'un pipeline (console)
- Suppression des balises d'un pipeline (console)

Ajout de balises à un pipeline (console)

Vous pouvez utiliser la console pour ajouter des balises à un pipeline existant.

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Sur la page Pipelines, sélectionnez le pipeline dans lequel vous souhaitez ajouter des balises.
3. Choisissez Paramètres dans le volet de navigation.
4. Sous Balises de pipeline, choisissez Modifier.
5. Dans les champs Clé et Valeur, entrez une paire de clés pour chaque ensemble de balises que vous souhaitez ajouter. (Le champ Valeur est facultatif.) Par exemple, dans Clé, saisissez **`Project`**. Dans Value (Valeur), entrez **`ProjectA`**.
6. (Facultatif) Choisissez Add tag (Ajouter une balise) pour ajouter d'autres lignes et saisir d'autres balises.
7. Sélectionnez Soumettre. Les balises sont répertoriées sous les paramètres des pipelines.

Affichage des balises d'un pipeline (console)

Vous pouvez utiliser la console pour afficher la liste des balises pour les pipelines existants.

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Sur la page Pipelines, sélectionnez le pipeline dans lequel vous souhaitez afficher les balises.
3. Choisissez Paramètres dans le volet de navigation.
4. Sous Balises de pipeline, affichez les balises du pipeline dans les colonnes Clé et Valeur.

Modification des balises d'un pipeline (console)

Vous pouvez utiliser la console pour modifier les balises qui ont été ajoutées aux pipelines.

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Sur la page Pipelines, sélectionnez le pipeline dans lequel vous souhaitez mettre à jour des balises.
3. Choisissez Paramètres dans le volet de navigation.
4. Sous Balises de pipeline, choisissez Modifier.
5. Dans les champs Clé et Valeur, mettez à jour les valeurs dans chaque champ selon vos besoins. Par exemple, pour la clé **`Project`**, dans Valeur, remplacez **`ProjectA`** par **`ProjectB`**.
6. Sélectionnez Soumettre.

Suppression des balises d'un pipeline (console)

Vous pouvez utiliser la console pour supprimer des balises de pipelines. Lorsque vous supprimez des balises de la ressource associée, les balises sont supprimées.

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Sur la page Pipelines, sélectionnez le pipeline dans lequel vous souhaitez supprimer des balises.
3. Choisissez Paramètres dans le volet de navigation.
4. Sous Balises de pipeline, choisissez Modifier.

5. En regard de la clé et de la valeur de chaque balise que vous souhaitez supprimer, choisissez Supprimer la balise.
6. Sélectionnez Soumettre.

Balisage de pipelines (interface de ligne de commande)

Vous pouvez utiliser l'interface de ligne de commande pour baliser des ressources. Vous devez utiliser la console pour gérer les balises dans les pipelines.

Rubriques

- Ajout de balises à un pipeline (interface de ligne de commande)
- Affichage des balises d'un pipeline (interface de ligne de commande)
- Modification des balises d'un pipeline (interface de ligne de commande)
- Suppression des balises d'un pipeline (interface de ligne de commande)

Ajout de balises à un pipeline (interface de ligne de commande)

Vous pouvez utiliser la console ou le AWS CLI pour baliser les pipelines.

Pour ajouter une balise à un pipeline lors de sa création, veuillez consulter Création d'un pipeline, d'étapes et d'actions.

Dans ces étapes, nous supposons que vous avez déjà installé une version récente de l' AWS CLI ou que vous avez procédé à une mise à jour vers la version actuelle. Pour plus d'informations, consultez Installing the AWS Command Line Interface (Installation de).

Depuis le terminal ou la ligne de commande, exécutez la commande tag-resource, en spécifiant l'ARN (Amazon Resource Name) du pipeline dans lequel vous souhaitez ajouter des balises ainsi que la clé et la valeur de la balise que vous souhaitez ajouter. Vous pouvez ajouter plusieurs balises à un pipeline. Par exemple, pour baliser un pipeline nommé *MyPipeline* avec deux balises, une clé de balise nommée *DeploymentEnvironment* avec la valeur de *Test* balise et une clé de balise nommée *IscontainerBased* avec la valeur de balise de *true* :

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tags key=Project,value=ProjectA key=IscontainerBased,value=true
```

Si elle aboutit, cette commande ne renvoie rien.

Affichage des balises d'un pipeline (interface de ligne de commande)

Procédez comme suit pour utiliser le AWS CLI afin d'afficher les AWS balises d'un pipeline. Si aucune balise n'a été ajoutée, la liste renvoyée est vide.

Depuis le terminal ou la ligne de commande, exécutez la commande list-tags-for-resource. Par exemple, pour afficher la liste des clés de balise et des valeurs de balise pour un pipeline nommé *MyPipeline* avec la valeur `arn:aws:codepipeline:`*us-west-2*`:`*account-id*`:`*MyPipeline* ARN :

```
aws codepipeline list-tags-for-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:MyPipeline
```

Si elle aboutit, cette commande renvoie des informations similaires à ce qui suit :

```
{
    "tags": {
        "Project": "ProjectA",
        "IscontainerBased": "true"
    }
}
```

Modification des balises d'un pipeline (interface de ligne de commande)

Procédez comme suit pour utiliser le AWS CLI pour modifier une balise pour un pipeline. Vous pouvez modifier la valeur d'une clé existante ou ajouter une autre clé. Vous pouvez également supprimer des balises d'un pipeline, comme indiqué dans la section suivante.

Depuis le terminal ou la ligne de commande, exécutez la commande tag-resource, en spécifiant l'ARN du pipeline dans lequel vous souhaitez mettre à jour une balise et spécifiez la clé de balise et la valeur de balise :

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-
id:MyPipeline --tags key=Project,value=ProjectA
```

Si elle aboutit, cette commande ne renvoie rien.

Suppression des balises d'un pipeline (interface de ligne de commande)

Procédez comme suit pour utiliser le AWS CLI pour supprimer une balise d'un pipeline. Lorsque vous supprimez des balises de la ressource associée, les balises sont supprimées.

> **Note**
>
> Si vous supprimez un pipeline, toutes les associations de balises sont supprimées du pipeline supprimé. Vous n'avez pas besoin de supprimer les balises avant de supprimer un pipeline.

Depuis le terminal ou la ligne de commande, exécutez la commande untag-resource, en spécifiant l'ARN du pipeline dans lequel vous souhaitez supprimer des balises et la clé de balise de la balise que vous souhaitez supprimer. Par exemple, pour supprimer plusieurs balises sur un pipeline nommé *MyPipeline* avec les clés de balise *Project* et *IscontainerBased* :

```
aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tag-keys Project IscontainerBased
```

Si elle aboutit, cette commande ne renvoie rien. Pour vérifier quelles balises sont associées au pipeline, exécutez la commande list-tags-for-resource.

Création d'une règle de notification

Vous pouvez utiliser des règles de notification pour informer les utilisateurs de modifications importantes, par exemple, lors du démarrage de l'exécution d'un pipeline. Les règles de notification spécifient à la fois les événements et le sujet Amazon SNS utilisé pour envoyer des notifications. Pour plus d'informations, consultez Que sont les notifications ?

Vous pouvez utiliser la console ou le AWS CLI pour créer des règles de notification pour AWS CodePipeline.

Pour créer une règle de notification (console)

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Choisissez Pipelines, puis choisissez un pipeline dans lequel vous souhaitez ajouter des notifications.
3. Sur la page du pipeline, choisissez Notify (Notifier), puis Create notification rule (Créer une règle de notification). Vous pouvez également accéder à la page Settings (Paramètres) du pipeline et choisir Create notification rule (Créer une règle de notification).

4. Dans Notification name (Nom de la notification), saisissez le nom de la règle.
5. Dans Type de détail, choisissez Basic si vous souhaitez que seules les informations fournies à Amazon soient EventBridge incluses dans la notification. Choisissez Complet si vous souhaitez inclure les informations fournies à Amazon EventBridge et les informations susceptibles d'être fournies par le CodePipeline ou le gestionnaire de notifications.

   Pour plus d'informations, consultez Présentation du contenu des notifications et de la sécurité.
6. Dans Événements qui déclenchent des notifications, sélectionnez les événements pour lesquels vous souhaitez envoyer des notifications. Pour de plus amples informations, veuillez consulter Événements pour les règles de notification sur les pipelines.
7. Dans Targets (Cibles), effectuez l'une des actions suivantes :

   - Si vous avez déjà configuré une ressource à utiliser avec les notifications, dans Choisir le type de cible, sélectionnez Amazon Q Developer dans les applications de chat (Slack) ou le sujet SNS. Dans Choisir une cible, choisissez le nom du client (pour un client Slack configuré dans Amazon Q Developer dans les applications de chat) ou le nom de ressource Amazon (ARN) du sujet Amazon SNS (pour les sujets Amazon SNS déjà configurés avec la politique requise pour les notifications).
   - Si vous n'avez pas configuré de ressource à utiliser avec les notifications, choisissez Create target (Créer une cible), puis SNS topic (Rubrique SNS). Donnez un nom à la rubrique après codestar-notifications-, puis choisissez Create (Créer).

   > **Note**
   >
   > - Si vous créez la rubrique Amazon SNS dans le cadre de la création de la règle de notification, la stratégie qui permet à la fonctionnalité de notifications de publier des événements dans la rubrique est appliquée automatiquement. L'utilisation d'une rubrique créée pour les règles de notification vous permet de vous assurer que vous n'abonnez que les utilisateurs qui doivent recevoir des notifications sur ce référentiel.
   > - Vous ne pouvez pas créer un client Amazon Q Developer dans les applications de chat dans le cadre de la création d'une règle de notification. Si vous choisissez Amazon Q Developer dans les applications de chat (Slack), vous verrez un bouton vous demandant de configurer un client dans Amazon Q Developer dans les applications de chat. Le choix de cette option ouvre la console Amazon Q Developer dans les applications de chat. Pour plus d'informations, consultez Configurer les

intégrations entre les notifications et Amazon Q Developer dans les applications de chat.

- Si vous souhaitez utiliser une rubrique Amazon SNS existante comme cible, vous devez ajouter la politique requise pour les AWS CodeStar notifications en plus de toute autre politique susceptible d'exister pour cette rubrique. Pour de plus amples informations, veuillez consulter Configurer les rubriques Amazon SNS existantes pour les notifications et Présentation du contenu des notifications et de la sécurité.

8. Pour terminer la création de la règle, choisissez Submit (Soumettre).
9. Vous devez inscrire les utilisateurs à la rubrique Amazon SNS relative à la règle pour qu'ils puissent recevoir des notifications. Pour plus d'informations, consultez Abonner des utilisateurs aux rubriques Amazon SNS qui sont des cibles. Vous pouvez également configurer l'intégration entre les notifications et Amazon Q Developer dans les applications de chat afin d'envoyer des notifications aux forums de discussion Amazon Chime ou aux canaux Slack. Pour plus d'informations, consultez Configurer l'intégration entre les notifications et Amazon Q Developer dans les applications de chat.

Pour créer une règle de notification (AWS CLI)

1. Dans un terminal ou une invite de commandes, exécutez la commande create-notification rule pour générer le squelette JSON :

```
aws codestar-notifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

Vous pouvez donner au fichier le nom de votre choix. Dans cet exemple, le fichier est nommé *rule.json*.

2. Ouvrez le fichier JSON dans un éditeur de texte brut et modifiez-le pour y inclure la ressource, les types d'événements et la cible que vous souhaitez pour la règle. L'exemple suivant montre une règle de notification nommée **MyNotificationRule** d'après un pipeline nommé *MyDemoPipeline* dans un AWS compte avec cet ID*123456789012*. Les notifications sont envoyées avec le type de détail complet à une rubrique Amazon SNS nommée *codestar-notifications-MyNotificationTopic* when pipeline execution start :

```
{
    "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"
}
```

Enregistrez le fichier.

3. À l'aide du fichier que vous venez de modifier, à partir du terminal ou de la ligne de commande, exécutez à nouveau la commande create-notification-rule pour créer la règle de notification :

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. En cas de réussite, la commande renvoie l'ARN de la règle de notification, comme suit :

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

Actions à la source et méthodes de détection des modifications

Lorsque vous ajoutez une action source à votre pipeline, les actions fonctionnent avec les ressources supplémentaires décrites dans le tableau.

> **Note**
>
> Les actions source CodeCommit et S3 nécessitent soit une ressource de détection des modifications configurée (une EventBridge règle), soit l'option permettant d'interroger le référentiel pour connaître les modifications de source. Pour les pipelines dotés d'une action source Bitbucket ou GitHub Enterprise Server, il n'est pas nécessaire de configurer un webhook ou d'effectuer un sondage par défaut. GitHub L'action Connexions gère pour vous la détection des modifications.

Source	Utilise des ressources supplémentaires ?	Étapes
Amazon S3 avec CloudTrail ressources	Cette action source utilise une règle d'événement et des CloudTrail ressources supplémentaires. Lorsque vous utilisez la CLI ou CloudFormation que vous créez cette action, vous créez et gérez également ces ressources.	Voir Création d'un pipeline, d'étapes et d'actions et Connexion à la source Amazon S3 : actions qui utilisent EventBridge et AWS CloudTrail
Amazon S3 sans CloudTrail ressources	Cette action source utilise un compartiment activé pour les événements dotés d'une règle d'événement sans nécessiter de CloudTrail ressources	Voir Création d'un pipeline, d'étapes et d'actions et Connexion aux actions de source Amazon S3 avec une source activée pour les événements

Source	Utilise des ressources supplémentaires ?	Étapes	
	supplémentaires. Lorsque vous utilisez la CLI ou CloudFormation que vous créez cette action, vous créez et gérez également ces ressources.		
Bitbucket Cloud	Cette action source utilise une ressource de connexion.	Consultez Connexions Bitbucket Cloud	
AWS CodeCommit	Amazon EventBridge (recommandé). Il s'agit de la valeur par défaut pour les pipelines avec une source CodeCommit créés ou modifiés dans la console.	Voir Création d'un pipeline, d'étapes et d'actions et CodeCommit actions à la source et EventBridge	
Amazon ECR	Amazon EventBridge. Ceci est créé par l'assistant pour les pipelines avec une source Amazon ECR créée ou modifiée dans la console.	Consultez Création d'un pipeline, d'étapes et d'actions et Actions et ressources relatives aux sources Amazon ECR EventBridge.	
GitHub ou GitHub Enterprise Cloud	Cette action source utilise une ressource de connexion.	Consultez GitHub connexions	
GitHub Serveur d'entreprise	Cette action source utilise une ressource de connexion et une ressource hôte.	Consultez GitHub Connexions aux serveurs d'entreprise	

Source	Utilise des ressources supplémentaires ?	Étapes	
GitLab.com	Cette action source utilise une ressource de connexion.	Consultez GitLab.com connexions	
GitLab autogéré	Cette action source utilise une ressource de connexion et une ressource hôte.	Consultez Connexions pour l' GitLab autogestion	

Si vous disposez d'un pipeline qui utilise l'interrogation, vous pouvez le mettre à jour pour qu'il utilise la méthode de détection recommandée. Pour de plus amples informations, veuillez consulter Mise à jour de pipelines d'interrogation vers la méthode recommandée de détection des modifications.

Si vous souhaitez désactiver la détection des modifications pour une action source utilisant des connexions, consultezCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

Connectez-vous à des fournisseurs de sources propriétaires à l'aide d'actions de source

Vous pouvez utiliser la AWS CodePipeline console ou le AWS CLI pour vous connecter à des fournisseurs d'actions source, tels que CodeCommit S3.

> **Note**
>
> Lorsque vous utilisez la console pour créer ou modifier un pipeline, les ressources de détection des modifications sont créées pour vous. Si vous utilisez l' AWS interface de ligne de commande pour créer le pipeline, vous devez créer vous-même les ressources supplémentaires. Pour de plus amples informations, veuillez consulter CodeCommit actions à la source et EventBridge.

Rubriques

- Actions et ressources relatives aux sources Amazon ECR EventBridge
- Connexion aux actions de source Amazon S3 avec une source activée pour les événements
- Connexion à la source Amazon S3 : actions qui utilisent EventBridge et AWS CloudTrail
- CodeCommit actions à la source et EventBridge

Actions et ressources relatives aux sources Amazon ECR EventBridge

Pour ajouter une action source Amazon ECR dans CodePipeline, vous pouvez choisir de :

- Utilisez l'assistant de création de pipeline (Création d'un pipeline personnalisé (console)) de la CodePipeline console ou la page Modifier l'action pour choisir l'option fournisseur Amazon ECR. La console crée une EventBridge règle qui démarre votre pipeline lorsque la source change.
- Utilisez l'interface de ligne de commande pour ajouter la configuration de l'ECRaction et créer des ressources supplémentaires comme suit :
  - Utilisez l'ECRexemple de configuration d'action Référence des actions source Amazon ECR pour créer votre action comme indiqué dansCréation d'un pipeline (interface de ligne de commande).

- La méthode de détection des modifications démarre par défaut le pipeline en interrogeant la source. Vous devez désactiver les contrôles périodiques et créer la règle de détection des modifications manuellement. Utilisez l'une des méthodes suivantes : Créer une EventBridge règle pour une source Amazon ECR (console)Création d'une EventBridge règle pour une source Amazon ECR (CLI), ouCréez une EventBridge règle pour une source Amazon ECR (CloudFormation modèle).

Rubriques

- Créer une EventBridge règle pour une source Amazon ECR (console)
- Création d'une EventBridge règle pour une source Amazon ECR (CLI)
- Créez une EventBridge règle pour une source Amazon ECR (CloudFormation modèle)

Créer une EventBridge règle pour une source Amazon ECR (console)

Pour créer une EventBridge règle à utiliser dans le CodePipeline cadre des opérations (source Amazon ECR)

1. Ouvrez la EventBridge console Amazon à l'adresse https://console.aws.amazon.com/events/.
2. Dans le panneau de navigation, sélectionnez Events.
3. Choisissez Créer une règle, puis sous Source de l'événement, dans Nom du service, choisissez Elastic Container Registry (ECR).
4. Dans Source d'événement, choisissez Modèle d'événement.

   Choisissez Edit (Modifier), puis collez l'exemple de modèle d'événement suivant dans la fenêtre Event Source (Source de l'événement) pour un référentiel `eb-test` avec une balise d'image `cli-testing` :

```
{
    "detail-type": [
        "ECR Image Action"
    ],
    "source": [
        "aws.ecr"
    ],
    "detail": {
        "action-type": [
            "PUSH"
```

```
        ],
        "image-tag": [
            "latest"
        ],
        "repository-name": [
            "eb-test"
        ],
        "result": [
            "SUCCESS"
        ]
    }
}
```

> **Note**
>
> Pour consulter le modèle d'événement complet pris en charge pour les événements Amazon ECR, consultez Amazon ECR Events EventBridge et/ou Amazon Elastic Container Registry Events.

5. Choisissez Enregistrer.

   Dans le volet Aperçu du modèle d'événement, prévisualisez la règle.

6. Dans Targets, choisissez CodePipeline.
7. Entrez l'ARN du pipeline pour le pipeline à démarrer selon cette règle.

> **Note**
>
> Vous trouverez l'ARN de pipeline dans la sortie des métadonnées après avoir exécuté la commande get-pipeline. L'ARN de pipeline est élaboré dans ce format : arn:aws:codepipeline : : : *region account pipeline-name*
> Exemple d'ARN de pipeline : `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

8. Créez ou spécifiez un rôle de service IAM qui accorde EventBridge des autorisations pour invoquer la cible associée à votre EventBridge règle (dans ce cas, la cible est CodePipeline).

   - Choisissez Créer un nouveau rôle pour cette ressource spécifique afin de créer un rôle de service qui donne EventBridge les autorisations nécessaires pour démarrer les exécutions de votre pipeline.

- Choisissez Utiliser le rôle existant pour saisir un rôle de service qui autorise EventBridge le démarrage des exécutions de votre pipeline.

9. (Facultatif) Pour spécifier des remplacements de source avec un ID d'image spécifique, utilisez le transformateur d'entrée pour transmettre les données sous forme de paramètres JSON. Vous pouvez également utiliser le transformateur d'entrée pour transmettre les variables du pipeline.

   - Développer Additional settings (Paramètres supplémentaires).

     Sous Configurer l'entrée cible, choisissez Configurer le transformateur d'entrée.

     Dans la fenêtre de dialogue, choisissez Enter my own. Dans la zone Chemin d'entrée, tapez les paires clé-valeur suivantes.

     ```
     {"revisionValue": "$.detail.image-digest"}
     ```

   - Dans la zone Modèle, tapez les paires clé-valeur suivantes.

     ```
     {
         "sourceRevisions": [
             {
                 "actionName": "Source",
                 "revisionType": "IMAGE_DIGEST",
                 "revisionValue": "<revisionValue>"
             }
         ],
          "variables": [
             {
                 "name": "Variable_Name",
                 "value": "Variable_Value"
             }
         ]
     }
     ```

   - Choisissez Confirmer.

10. Passez en revue la configuration de votre règle pour vous assurer qu'elle correspond à vos besoins.
11. Choisissez Configure details (Configurer les détails).
12. Sur la page Configurer les détails de la règle, entrez un nom et une description pour la règle, puis cochez État pour activer la règle.

13. Si la règle vous convient, choisissez Créer une règle.

Création d'une EventBridge règle pour une source Amazon ECR (CLI)

Appelez la commande put-rule, en spécifiant les éléments suivants :

- Un nom qui identifie de façon unique la règle que vous créez. Ce nom doit être unique pour tous les pipelines que vous créez et qui CodePipeline sont associés à votre AWS compte.
- Le modèle d'événement pour la source et les champs de détails utilisés par la règle. Pour plus d'informations, consultez Amazon EventBridge et Event Patterns.

Pour créer une EventBridge règle avec Amazon ECR comme source d'événement et CodePipeline comme cible

1. Ajoutez des autorisations EventBridge à utiliser CodePipeline pour invoquer la règle. Pour plus d'informations, consultez la section Utilisation de politiques basées sur les ressources pour Amazon. EventBridge

   a. Utilisez l'exemple suivant pour créer la stratégie d'approbation qui permet à EventBridge d'assumer le rôle de service. Nommez la stratégie d'approbation `trustpolicyforEB.json`.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   b. Utilisez la commande suivante pour créer le rôle `Role-for-MyRule` et attachez la stratégie d'approbation.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Créez le JSON de stratégie d'autorisations comme indiqué dans cet exemple pour le pipeline nommé `MyFirstPipeline`. Nommez la stratégie d'autorisations `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilisez la commande suivante pour attacher au rôle `Role-for-MyRule` la stratégie d'autorisations `CodePipeline-Permissions-Policy-for-EB`.

Pourquoi est-ce que j'effectue cette modification ? L'ajout de cette politique au rôle crée des autorisations pour EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Appelez la commande put-rule et incluez les paramètres `--name`, `--event-pattern` et `--role-arn`.

Pourquoi est-ce que j'effectue cette modification ? Vous devez créer un événement avec une règle qui spécifie comment un push d'image doit être effectué et une cible qui nomme le pipeline devant être démarré par l'événement.

L'exemple de commande suivant crée une règle nommée `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**
>
> Pour consulter le modèle d'événement complet pris en charge pour les événements Amazon ECR, consultez Amazon ECR Events EventBridge et/ou Amazon Elastic Container Registry Events.

3. Pour ajouter CodePipeline en tant que cible, appelez la put-targets commande et incluez les paramètres suivants :

   - Le paramètre `--rule` s'utilise avec le la règle `rule_name` que vous avez créée à l'aide de la commande put-rule.
   - Le paramètre `--targets` s'utilise avec l'ID de liste `Id` de la cible figurant dans la liste des cibles et l'ARN du pipeline cible.

   L'exemple de commande suivant spécifie que pour la règle appelée `MyECRRepoRule`, l'`Id` cible est composé du numéro un, ce qui indique qu'il s'agit de la règle 1 dans une liste de cibles pour la règle. L'exemple de commande spécifie également un exemple d'`Arn` pour le pipeline et l'exemple de `RoleArn` pour la règle. Le pipeline démarre lorsque des modifications sont effectuées dans le référentiel.

   ```
   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. (Facultatif) Pour configurer un transformateur d'entrée avec des remplacements de source pour un ID d'image spécifique, utilisez le JSON suivant dans votre commande CLI. L'exemple suivant configure une dérogation dans laquelle :

   - `Source`Dans cet exemple`actionName`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.

- `IMAGE_DIGEST`Dans cet exemple`revisionType`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
- Dans cet exemple`revisionValue`, **<** *`revisionValue`* **>** est dérivé de la variable d'événement source.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "ARN",
            "InputTransformer": {
                "InputPathsMap": {
                    "revisionValue": "$.detail.image-digest"
                },
                "InputTemplate": {
                    "sourceRevisions": [
                        {
                            "actionName": "Source",
                            "revisionType": "IMAGE_DIGEST",
                            "revisionValue": "<revisionValue>"
                        }
                    ]
                }
            }
        }
    ]
}
```

Créez une EventBridge règle pour une source Amazon ECR (CloudFormation modèle)

CloudFormation Pour créer une règle, utilisez l'extrait de modèle comme indiqué ici.

Pour mettre à jour votre pipeline CloudFormation modèle et création de EventBridge règle

1. Dans le modèle, ci-dessous`Resources`, utilisez la `AWS::IAM::Role` CloudFormation ressource pour configurer le rôle IAM qui permet à votre événement de démarrer votre pipeline. Cette entrée crée un rôle qui utilise deux stratégies :

   - La première stratégie autorise le rôle à être endossé.
   - La deuxième stratégie fournit des autorisations pour démarrer le pipeline.

   Pourquoi est-ce que j'effectue cette modification ? Vous devez créer un rôle qui peut être assumé par EventBridge pour démarrer une exécution dans notre 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. Dans le modèle, ci-dessous`Resources`, utilisez la `AWS::Events::Rule` CloudFormation ressource pour ajouter une EventBridge règle pour la source Amazon ECR. Ce modèle d'événement crée un événement qui surveille les validations dans votre référentiel. Lorsqu'un changement d'état du référentiel est EventBridge détecté, la règle est invoquée `StartPipelineExecution` sur votre pipeline cible.

   Pourquoi est-ce que je fais ce changement ? Vous devez créer un événement avec une règle qui spécifie comment un push d'image doit être effectué et une cible qui nomme le pipeline devant être démarré par l'événement.

   Cet extrait de code utilise une image nommée `eb-test` avec une balise `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**
>
> Pour consulter le modèle d'événement complet pris en charge pour les événements Amazon ECR, consultez Amazon ECR Events EventBridge et/ou Amazon Elastic Container Registry Events.

3. (Facultatif) Pour configurer un transformateur d'entrée avec des remplacements de source pour un ID d'image spécifique, utilisez l'extrait YAML suivant. L'exemple suivant configure une dérogation dans laquelle :

   - `SourceDans` cet exemple`actionName`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - `IMAGE_DIGESTDans` cet exemple`revisionType`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - Dans cet exemple`revisionValue`, **<** *`revisionValue`* **>** est dérivé de la variable d'événement source.

```
---
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: ARN
  InputTransformer:
    InputPathsMap:
      revisionValue: "$.detail.image-digest"
    InputTemplate:
      sourceRevisions:
        actionName: Source
        revisionType: IMAGE_DIGEST
        revisionValue: '<revisionValue>'
```

4. Enregistrez le modèle mis à jour sur votre ordinateur local, puis ouvrez la console CloudFormation .
5. Choisissez votre pile, puis Créer un jeu de modifications pour la pile actuelle.

6. Chargez le modèle mis à jour, puis affichez les modifications répertoriées dans CloudFormation. Il s'agit des modifications apportées à la pile. Vos nouvelles ressources doivent figurer dans la liste.
7. Sélectionnez Execute (Exécuter).

Connexion aux actions de source Amazon S3 avec une source activée pour les événements

Les instructions de cette section décrivent les étapes à suivre pour créer l'action source S3 qui ne nécessite pas la création ou la gestion de AWS CloudTrail ressources.

> **Important**
>
> La procédure permettant de créer cette action sans AWS CloudTrail ressources n'est pas disponible dans la console. Pour utiliser l'interface de ligne de commande, consultez les procédures ici ou consultezMigrer les pipelines de sondage avec une source S3 activée pour les événements.

Pour un pipeline avec une source Amazon S3, modifiez le pipeline afin que la détection des modifications soit automatisée via EventBridge et avec un compartiment source activé pour les notifications d'événements. Il s'agit de la méthode recommandée si vous utilisez l'interface de ligne de commande ou si vous CloudFormation souhaitez migrer votre pipeline.

> **Note**
>
> Cela inclut l'utilisation d'un compartiment activé pour les notifications d'événements, dans lequel vous n'avez pas besoin de créer un historique CloudTrail distinct. Si vous utilisez la console, une règle d'événement et CloudTrail un suivi sont configurés pour vous. Pour ces étapes, voirMigrer les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3.

- CLI : Migrer les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3 (CLI)
- CloudFormation: Migrez les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3 (CloudFormation modèle)

Créez des pipelines avec une source S3 activée pour les événements (CLI)

Suivez ces étapes pour créer un pipeline avec une source S3 qui utilise un événement dans EventBridge pour la détection des modifications. Pour connaître les étapes complètes de création d'un pipeline à l'aide de l'interface de ligne de commande, consultezCréation d'un pipeline, d'étapes et d'actions.

Pour créer un pipeline piloté par les événements avec Amazon S3, vous modifiez le `PollForSourceChanges` paramètre de votre pipeline, puis vous créez les ressources suivantes :

- EventBridge règle de l'événement
- Rôle IAM pour permettre à l' EventBridge événement de démarrer votre pipeline

Pour créer une EventBridge règle avec Amazon S3 comme source et CodePipeline comme cible de l'événement et appliquer la politique d'autorisations

1. Accordez les autorisations nécessaires EventBridge CodePipeline pour invoquer la règle. Pour plus d'informations, consultez la section Utilisation de politiques basées sur les ressources pour Amazon. EventBridge

   a. Utilisez l'exemple suivant pour créer la stratégie d'approbation qui permet à EventBridge d'assumer le rôle de service. Nommez-la `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Utilisez la commande suivante pour créer le rôle `Role-for-MyRule` et attachez la stratégie d'approbation.

Pourquoi est-ce que j'effectue cette modification ? L'ajout de cette stratégie d'approbation au rôle crée des autorisations pour EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Créez le JSON de stratégie d'autorisations, comme décrit ici, pour le pipeline nommé `MyFirstPipeline`. Nommez la stratégie d'autorisations `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilisez la commande suivante pour attacher la nouvelle stratégie d'autorisations `CodePipeline-Permissions-Policy-for-EB` au rôle `Role-for-MyRule` que vous avez créé.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Appelez la commande put-rule et incluez les paramètres `--name`, `--event-pattern` et `--role-arn`.

L'exemple de commande suivant crée une règle nommée `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. Pour ajouter CodePipeline en tant que cible, appelez la put-targets commande et incluez les `--targets` paramètres `--rule` et.

   La commande suivante spécifie que pour la règle nommée `EnabledS3SourceRule`, l'`Id` cible est composé du numéro un, ce qui indique qu'il s'agit de la règle 1 dans une liste de cibles pour la règle. La commande spécifie également un exemple d'ARN pour le pipeline. Le pipeline démarre lorsque des modifications sont effectuées dans le référentiel.

   ```
   aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-
   AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

1. Exécutez la commande get-pipeline pour copier la structure de pipeline dans un fichier JSON. Par exemple, pour un pipeline nommé `MyFirstPipeline`, exécutez la commande suivante :

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Cette commande ne renvoie rien, mais le fichier que vous avez créé doit apparaître dans le répertoire où vous avez exécuté la commande.

2. Ouvrez le fichier JSON dans un éditeur de texte brut et modifiez l'étape source en remplaçant la valeur du paramètre `PollForSourceChanges` pour un compartiment nommé `amzn-s3-demo-source-bucket` par `false`, comme illustré dans cet exemple.

   Pourquoi est-ce que j'effectue cette modification ? La définition de ce paramètre sur `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Si vous utilisez la structure de pipeline extraite à l'aide de la commande get-pipeline, vous devez supprimer les lignes `metadata` du fichier JSON. Sinon, la commande update-pipeline ne peut pas l'utiliser. Supprimez les lignes `"metadata": { }` et les champs `"updated"`, `"created"` et `"pipelineARN"`.

   Par exemple, supprimez les lignes suivantes de la structure :

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

   Enregistrez le fichier.

4. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline :

   > **Important**
   >
   > N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

Cette commande affiche toute la structure du pipeline mise à jour.

> **Note**
>
> La commande update-pipeline interrompt le pipeline. Si une révision est exécutée dans le pipeline lorsque vous exécutez la commande update-pipeline celle-ci est interrompue. Vous devez lancer manuellement le pipeline pour exécuter cette révision dans le pipeline mis à jour. Utilisez la commande start-pipeline-execution pour démarrer manuellement votre pipeline.

Créez des pipelines avec une source S3 activée pour les événements (CloudFormation modèle)

Cette procédure concerne un pipeline dans lequel les événements sont activés dans le compartiment source.

Suivez ces étapes pour créer un pipeline avec une source Amazon S3 pour la détection des modifications basée sur les événements.

Pour créer un pipeline piloté par les événements avec Amazon S3, vous modifiez le `PollForSourceChanges` paramètre de votre pipeline, puis vous ajoutez les ressources suivantes à votre modèle :

- EventBridge règle et rôle IAM pour permettre à cet événement de démarrer votre pipeline.

Si vous l'utilisez CloudFormation pour créer et gérer vos pipelines, votre modèle inclut un contenu tel que le suivant.

> **Note**
>
> La propriété `Configuration` de l'étape source appelée `PollForSourceChanges`. Si votre modèle ne comprend pas cette propriété, `PollForSourceChanges` est défini sur `true` par défaut.

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
                        }
                    ]
                },


...
```

Pour créer une EventBridge règle avec Amazon S3 comme source et CodePipeline comme cible de l'événement et appliquer la politique d'autorisations

1. Dans le modèle, ci-dessous`Resources`, utilisez la `AWS::IAM::Role` CloudFormation ressource pour configurer le rôle IAM qui permet à votre événement de démarrer votre pipeline. Cette entrée crée un rôle qui utilise deux stratégies :

   - La première stratégie autorise le rôle à être endossé.
   - La deuxième stratégie fournit des autorisations pour démarrer le pipeline.

Pourquoi est-ce que j'effectue cette modification ? L'ajout d'une `AWS::IAM::Role` ressource CloudFormation permet de créer des autorisations pour EventBridge. Cette ressource est ajoutée à votre CloudFormation pile.

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. Utilisez la `AWS::Events::Rule` CloudFormation ressource pour ajouter une EventBridge règle. Ce modèle d'événement crée un événement qui surveille la création ou la suppression d'objets dans votre compartiment source Amazon S3. En outre, incluez une cible de votre pipeline. Lorsqu'un objet est créé, cette règle est invoquée `StartPipelineExecution` sur votre pipeline cible.

   Pourquoi est-ce que j'effectue cette modification ? L'ajout de la `AWS::Events::Rule` ressource CloudFormation permet de créer l'événement. Cette ressource est ajoutée à votre CloudFormation pile.

   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. Enregistrez le modèle mis à jour sur votre ordinateur local, puis ouvrez la console CloudFormation .
4. Choisissez votre pile, puis Créer un jeu de modifications pour la pile actuelle.
5. Chargez votre modèle mis à jour, puis affichez les modifications répertoriées dans CloudFormation. Il s'agit des modifications qui seront apportées à la pile. Vos nouvelles ressources doivent figurer dans la liste.
6. Sélectionnez Execute (Exécuter).

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

- Dans le modèle, remplacez la valeur du paramètre `PollForSourceChanges` par `false`. Si vous n'avez pas inclus `PollForSourceChanges` dans votre définition de pipeline, ajoutez ce paramètre et définissez-le sur `false`.

  Pourquoi est-ce que j'effectue cette modification ? Le remplacement de la valeur du paramètre `PollForSourceChanges` par `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

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

Lorsque vous créez CloudFormation ces ressources, votre pipeline est déclenché lorsque les fichiers de votre référentiel sont créés ou mis à jour.

> **Note**
>
> Ne vous arrêtez pas là. Bien que votre pipeline soit créé, vous devez créer un deuxième CloudFormation modèle pour votre pipeline Amazon S3. Si vous ne créez pas le second modèle, votre pipeline ne dispose d'aucune fonctionnalité de détection des modifications.

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"
        }
    ]
}
```

Connexion à la source Amazon S3 : actions qui utilisent EventBridge et AWS CloudTrail

Les instructions de cette section décrivent les étapes à suivre pour créer l'action source S3 qui utilise AWS CloudTrail les ressources que vous devez créer et gérer. Pour utiliser l'action source S3 EventBridge qui ne nécessite pas de AWS CloudTrail ressources supplémentaires, utilisez les instructions CLI à l'adresseMigrer les pipelines de sondage avec une source S3 activée pour les événements.

> **Important**
>
> Cette procédure décrit les étapes à suivre pour créer l'action source S3 qui utilise AWS CloudTrail les ressources que vous devez créer et gérer. La procédure permettant de créer cette action sans AWS CloudTrail ressources n'est pas disponible dans la console. Pour utiliser l'interface de ligne de commande, consultezMigrer les pipelines de sondage avec une source S3 activée pour les événements.

Pour ajouter une action source Amazon S3 dans CodePipeline, vous pouvez choisir l'une des options suivantes :

- Utilisez l'assistant de création de pipeline (Création d'un pipeline personnalisé (console)) de la CodePipeline console ou la page Modifier l'action pour choisir l'option du fournisseur S3. La console crée une EventBridge règle et un CloudTrail journal qui démarrent votre pipeline lorsque la source change.
- Utilisez le AWS CLI pour ajouter la configuration d'action pour l'S3action et créer des ressources supplémentaires comme suit :
  - Utilisez l'S3exemple de configuration d'action Référence des actions source d'Amazon S3 pour créer votre action comme indiqué dansCréation d'un pipeline (interface de ligne de commande).

- La méthode de détection des modifications démarre par défaut le pipeline en interrogeant la source. Vous devez désactiver les contrôles périodiques et créer la règle de détection des modifications et le suivi manuellement. Utilisez l'une des méthodes suivantes : Création d'une EventBridge règle pour une source Amazon S3 (console)Création d'une EventBridge règle pour une source Amazon S3 (CLI), ouCréez une EventBridge règle pour une source Amazon S3 (CloudFormation modèle).

AWS CloudTrail est un service qui enregistre et filtre les événements sur votre compartiment source Amazon S3. Le trail envoie les modifications de source filtrées à la EventBridge règle. La EventBridge règle détecte le changement de source, puis démarre votre pipeline.

Prérequis:

- Si vous ne créez pas de journal, utilisez un historique existant AWS CloudTrail pour enregistrer les événements dans votre compartiment source Amazon S3 et envoyer les événements filtrés à la EventBridge règle.
- Créez ou utilisez un compartiment S3 existant dans lequel AWS CloudTrail vous pouvez stocker ses fichiers journaux. AWS CloudTrail doit disposer des autorisations requises pour transmettre des fichiers journaux à un compartiment Amazon S3. Le compartiment ne peut pas être configuré en tant que compartiment Paiement par le demandeur. Lorsque vous créez un compartiment Amazon S3 dans le cadre de la création ou de la mise à jour d'un journal dans la console AWS CloudTrail , les autorisations requises sont associées à un compartiment pour vous. Pour plus d'informations, consultez Amazon S3 Bucket Policy pour CloudTrail.

Création d'une EventBridge règle pour une source Amazon S3 (console)

Avant de configurer une règle dans EventBridge, vous devez créer une AWS CloudTrail trace. Pour plus d'informations, consultez Création d'un journal de suivi dans la console.

> **Important**
>
> Si vous utilisez la console pour créer ou modifier votre pipeline, votre EventBridge règle et votre AWS CloudTrail historique sont créés pour vous.

Pour créer un journal de suivi

1. Ouvrez la AWS CloudTrail console.
2. Dans le panneau de navigation, choisissez Journaux d'activité.
3. Choisissez Créer un journal d'activité). Pour Trail name (Nom du journal de suivi), entrez un nom pour votre journal de suivi.
4. Sous Storage location (Emplacement de stockage), créez ou spécifiez le compartiment à utiliser pour stocker les fichiers journaux. Par défaut, les objets et les compartiments Amazon S3 sont privés. Seul le propriétaire de la ressource (le AWS compte qui a créé le bucket) peut accéder au bucket et à ses objets. Le compartiment doit disposer d'une politique de ressources AWS CloudTrail qui autorise l'accès aux objets du compartiment.
5. Sous Compartiment et dossier du journal Trail, spécifiez un compartiment Amazon S3 et le préfixe de l'objet (nom du dossier) pour enregistrer les événements liés aux données pour tous les objets du dossier. Pour chaque journal de suivi, vous pouvez ajouter jusqu'à 250 objets Amazon S3. Complétez les informations de clé de chiffrement requises et choisissez Suivant.
6. Pour Type d'événement, choisissez Gestion des événements.
7. Pour Gestion des événements, choisissez Écrire. Le journal enregistre l'activité de l'API Amazon S3 au niveau de l'objet (par exemple, `GetObject` et`PutObject`) sur le compartiment et le préfixe spécifiés.
8. Choisissez Write (Écrire).
9. Si vous êtes satisfait du parcours, choisissez Créer un parcours.

Pour créer une EventBridge règle qui cible votre pipeline avec une source Amazon S3

1. Ouvrez la EventBridge console Amazon à l'adresse https://console.aws.amazon.com/events/.
2. Dans le panneau de navigation, choisissez Rules. Laissez le bus par défaut sélectionné ou choisissez un bus d'événements. Choisissez Créer une règle.
3. Dans Nom, entrez le nom de votre règle.
4. Sous Type de règle, choisissez Règle avec un modèle d'événement. Choisissez Suivant.
5. Sous Source de l'événement, choisissez AWS des événements ou des événements EventBridge partenaires.
6. Dans Exemple de type d'événement, choisissez AWS événements.
7. Dans Exemples d'événements, saisissez S3 comme mot clé sur lequel filtrer. Choisissez AWS l'appel API via CloudTrail.

8. Sous Méthode de création, choisissez Customer pattern (éditeur JSON).

   Collez le modèle d'événement fourni ci-dessous. Assurez-vous d'ajouter le nom du compartiment et la clé d'objet S3 (ou nom de clé) qui identifient de manière unique l'objet du compartiment comme étant`requestParameters`. Dans cet exemple, une règle est créée pour un compartiment nommé `amzn-s3-demo-source-bucket` et une clé d'objet de`my-files.zip`. Lorsque vous utilisez la fenêtre Modifier pour spécifier les ressources, votre règle est mise à jour pour utiliser un modèle événement personnalisé.

   Vous trouverez ci-dessous un exemple de modèle d'événement à copier et coller :

```
{
    "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. Choisissez Suivant.
10. Dans Types de cibles, choisissez AWS service.

11. Dans Sélectionnez une cible, choisissez CodePipeline. Dans Pipeline ARN, entrez l'ARN du pipeline pour le pipeline à démarrer selon cette règle.

> **Note**
>
> Pour l'obtenir l'ARN de pipeline, exécutez la commande get-pipeline. L'ARN de pipeline apparaît dans la sortie. Il est créé dans ce format :
> arn:aws:codepipeline : : : *region account pipeline-name*
> Exemple d'ARN de pipeline :
> arn:aws:codepipeline:us-east- 2:80398 EXAMPLE:MyFirstPipeline

12. Pour créer ou spécifier un rôle de service IAM qui accorde EventBridge des autorisations pour appeler la cible associée à votre EventBridge règle (dans ce cas, la cible est CodePipeline) :

    - Choisissez Créer un nouveau rôle pour cette ressource spécifique afin de créer un rôle de service qui donne EventBridge les autorisations nécessaires pour démarrer les exécutions de votre pipeline.
    - Choisissez Utiliser le rôle existant pour saisir un rôle de service qui autorise EventBridge le démarrage des exécutions de votre pipeline.

13. (Facultatif) Pour spécifier des remplacements de source avec un ID d'image spécifique, utilisez le transformateur d'entrée pour transmettre les données sous forme de paramètres JSON. Vous pouvez également utiliser le transformateur d'entrée pour transmettre les variables du pipeline.

    - Développer Additional settings (Paramètres supplémentaires).

      Sous Configurer l'entrée cible, choisissez Configurer le transformateur d'entrée.

      Dans la fenêtre de dialogue, choisissez Enter my own. Dans la zone Chemin d'entrée, tapez les paires clé-valeur suivantes.

```
{"revisionValue": "$.detail.object.version-id"}
```

    - Dans la zone Modèle, tapez les paires clé-valeur suivantes.

```
                                {
    "sourceRevisions": [
        {
```

```
            "actionName": "Source",
            "revisionType": "S3_OBJECT_VERSION_ID",
            "revisionValue": "<revisionValue>"
        }
    ],
     "variables": [
        {
            "name": "Variable_Name",
            "value": "Variable_Value"
        }
    ]
}
```

- Choisissez Confirmer.

14. Choisissez Suivant.
15. Sur la page Tags, choisissez Next.
16. Sur la page Réviser et créer, passez en revue la configuration des règles. Si la règle vous convient, choisissez Créer une règle.

Création d'une EventBridge règle pour une source Amazon S3 (CLI)

Pour créer un AWS CloudTrail suivre et activer la journalisation

Pour utiliser le AWS CLI pour créer une trace, appelez la create-trail commande en spécifiant :

- Le nom du journal de suivi.
- Le compartiment auquel vous avez déjà appliqué la stratégie de compartiment pour AWS CloudTrail.

Pour plus d'informations, consultez la section Création d'un historique à l'aide de l'interface de ligne de AWS commande.

1. Appelez la commande create-trail et incluez les paramètres `--name` et `--s3-bucket-name`.

   Pourquoi est-ce que j'effectue cette modification ? Cela crée le journal de suivi CloudTrail requis pour votre compartiment source S3.

La commande suivante utilise `--name` et `--s3-bucket-name` pour créer un journal de suivi nommé `my-trail` et un compartiment nommé `amzn-s3-demo-source-bucket`.

```
aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-source-bucket
```

2. Appelez la commande start-logging et incluez le paramètre `--name`.

   Pourquoi est-ce que je fais ce changement ? Cette commande lance la CloudTrail journalisation de votre compartiment source et envoie les événements à EventBridge.

   Exemple :

   La commande suivante utilise `--name` pour démarrer la journalisation dans un journal de suivi nommé `my-trail`.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Appelez la commande put-event-selectors et incluez les paramètres `--trail-name` et `--event-selectors`. Utilisez des sélecteurs d'événements pour spécifier que vous souhaitez que votre journal enregistre les événements de données pour votre compartiment source et envoie les événements à la EventBridge règle.

   Pourquoi est-ce que je fais ce changement ? Cette commande filtre les événements.

   Exemple :

   Dans l'exemple suivant, la commande utilise `--trail-name` et `--event-selectors` pour spécifier des événements de données pour un compartiment source et un préfixe nommés `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"] }] }]'
   ```

Pour créer une EventBridge règle avec Amazon S3 comme source et CodePipeline comme cible de l'événement et appliquer la politique d'autorisations

1. Accordez les autorisations nécessaires EventBridge CodePipeline pour invoquer la règle. Pour plus d'informations, consultez la section Utilisation de politiques basées sur les ressources pour Amazon. EventBridge

   a. Utilisez l'exemple suivant pour créer la politique de confiance permettant EventBridge d'assumer le rôle de service. Nommez-la `trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Utilisez la commande suivante pour créer le rôle `Role-for-MyRule` et attachez la stratégie d'approbation.

      Pourquoi est-ce que j'effectue cette modification ? L'ajout de cette politique de confiance au rôle crée des autorisations pour EventBridge.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Créez le JSON de stratégie d'autorisations, comme décrit ici, pour le pipeline nommé `MyFirstPipeline`. Nommez la stratégie d'autorisations `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilisez la commande suivante pour attacher la nouvelle stratégie d'autorisations `CodePipeline-Permissions-Policy-for-EB` au rôle `Role-for-MyRule` que vous avez créé.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Appelez la commande put-rule et incluez les paramètres `--name`, `--event-pattern` et `--role-arn`.

   L'exemple de commande suivant crée une règle nommée `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. Pour ajouter CodePipeline en tant que cible, appelez la put-targets commande et incluez les `--targets` paramètres `--rule` et.

La commande suivante spécifie que pour la règle nommée `MyS3SourceRule`, l'`Id` cible est composé du numéro un, ce qui indique qu'il s'agit de la règle 1 dans une liste de cibles pour la règle. La commande spécifie également un exemple d'ARN pour le pipeline. Le pipeline démarre lorsque des modifications sont effectuées dans le référentiel.

```
aws events put-targets --rule MyS3SourceRule --targets
 Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

4. (Facultatif) Pour configurer un transformateur d'entrée avec des remplacements de source pour un ID d'image spécifique, utilisez le JSON suivant dans votre commande CLI. L'exemple suivant configure une dérogation où :

   - `Source`Dans cet exemple`actionName`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - `S3_OBJECT_VERSION_ID`Dans cet exemple`revisionType`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - Dans cet exemple`revisionValue`, **<** *`revisionValue`* **>** est dérivé de la variable d'événement source.

```
{
    "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>"
                    }
                }
            }
        }
    ]
```

```
}
```

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

1. Exécutez la commande get-pipeline pour copier la structure de pipeline dans un fichier JSON. Par exemple, pour un pipeline nommé `MyFirstPipeline`, exécutez la commande suivante :

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Cette commande ne renvoie rien, mais le fichier que vous avez créé doit apparaître dans le répertoire où vous avez exécuté la commande.

2. Ouvrez le fichier JSON dans un éditeur de texte brut et modifiez l'étape source en remplaçant la valeur du paramètre `PollForSourceChanges` pour un compartiment nommé `amzn-s3-demo-source-bucket` par `false`, comme illustré dans cet exemple.

   Pourquoi est-ce que j'effectue cette modification ? La définition de ce paramètre sur `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Si vous utilisez la structure de pipeline extraite à l'aide de la commande get-pipeline, vous devez supprimer les lignes `metadata` du fichier JSON. Sinon, la commande update-pipeline ne peut

pas l'utiliser. Supprimez les lignes `"metadata": { }` et les champs `"updated"`, `"created"` et `"pipelineARN"`.

Par exemple, supprimez les lignes suivantes de la structure :

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

Enregistrez le fichier.

4. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline :

> **Important**
>
> N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

Cette commande affiche toute la structure du pipeline mise à jour.

> **Note**
>
> La commande update-pipeline interrompt le pipeline. Si une révision est exécutée dans le pipeline lorsque vous exécutez la commande update-pipeline celle-ci est interrompue. Vous devez lancer manuellement le pipeline pour exécuter cette révision dans le pipeline mis à jour. Utilisez la commande start-pipeline-execution pour démarrer manuellement votre pipeline.

Créez une EventBridge règle pour une source Amazon S3 (CloudFormation modèle)

CloudFormation Pour créer une règle, mettez à jour votre modèle comme indiqué ici.

Pour créer une EventBridge règle avec Amazon S3 comme source et CodePipeline comme cible de l'événement et appliquer la politique d'autorisations

1. Dans le modèle, ci-dessous`Resources`, utilisez la `AWS::IAM::Role` CloudFormation ressource pour configurer le rôle IAM qui permet à votre événement de démarrer votre pipeline. Cette entrée crée un rôle qui utilise deux stratégies :

   - La première stratégie autorise le rôle à être endossé.
   - La deuxième stratégie fournit des autorisations pour démarrer le pipeline.

   Pourquoi est-ce que j'effectue cette modification ? L'ajout d'une `AWS::IAM::Role` ressource CloudFormation permet de créer des autorisations pour EventBridge. Cette ressource est ajoutée à votre CloudFormation pile.

   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. Utilisez la `AWS::Events::Rule` CloudFormation ressource pour ajouter une EventBridge règle. Ce modèle d'événement crée un événement qui surveille `CopyObject` `PutObject` et `CompleteMultipartUpload` affecte votre compartiment source Amazon S3.
En outre, incluez une cible de votre pipeline. Lorsque `CopyObject`, `PutObject` ou `CompleteMultipartUpload` se produit, cette règle appelle `StartPipelineExecution` sur votre pipeline cible.

   Pourquoi est-ce que j'effectue cette modification ? L'ajout de la `AWS::Events::Rule` ressource CloudFormation permet de créer l'événement. Cette ressource est ajoutée à votre CloudFormation pile.

   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. Ajoutez cet extrait à votre premier modèle pour autoriser les fonctionnalités entre piles :

   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. (Facultatif) Pour configurer un transformateur d'entrée avec des remplacements de source pour un ID d'image spécifique, utilisez l'extrait YAML suivant. L'exemple suivant configure une dérogation dans laquelle :

   - `SourceDans cet exemple``actionName`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - `S3_OBJECT_VERSION_IDDans cet exemple``revisionType`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - Dans cet exemple`revisionValue`, < *`revisionValue`* > est dérivé de la variable d'événement source.

   ```
   ---
   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. Enregistrez le modèle mis à jour sur votre ordinateur local et ouvrez la CloudFormation console.
6. Choisissez votre pile, puis Créer un jeu de modifications pour la pile actuelle.
7. Chargez votre modèle mis à jour, puis affichez les modifications répertoriées dans CloudFormation. Il s'agit des modifications qui seront apportées à la pile. Vos nouvelles ressources doivent figurer dans la liste.
8. Sélectionnez Execute (Exécuter).

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

- Dans le modèle, remplacez la valeur du paramètre `PollForSourceChanges` par `false`. Si vous n'avez pas inclus `PollForSourceChanges` dans votre définition de pipeline, ajoutez ce paramètre et définissez-le sur `false`.

  Pourquoi est-ce que j'effectue cette modification ? Le remplacement de la valeur du paramètre `PollForSourceChanges` par `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

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
  }
```

Pour créer un deuxième modèle pour les CloudTrail ressources de votre pipeline Amazon S3

- Dans un modèle distinct, ci-dessous`Resources`, utilisez les `AWS::CloudTrail::Trail` CloudFormation ressources `AWS::S3::BucketAWS::S3::BucketPolicy`, et pour fournir une définition et un suivi simples du compartiment CloudTrail.

  Pourquoi est-ce que je fais ce changement ? Compte tenu de la limite actuelle de cinq pistes par compte, la CloudTrail piste doit être créée et gérée séparément. (Voir Limites dans AWS CloudTrail.) Cependant, vous pouvez inclure de nombreux compartiments Amazon S3 sur un seul trail, de sorte que vous puissiez créer le trail une seule fois, puis ajouter des compartiments Amazon S3 pour d'autres pipelines si nécessaire. Collez ce qui suit dans votre deuxième exemple de fichier de modèle.

  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 actions à la source et EventBridge

Pour ajouter une action CodeCommit source dans CodePipeline, vous pouvez choisir l'une des options suivantes :

- Utilisez l'assistant de création de pipeline (Création d'un pipeline personnalisé (console)) de la CodePipeline console ou la page Modifier l'action pour choisir l'option du CodeCommit fournisseur. La console crée une EventBridge règle qui démarre votre pipeline lorsque la source change.
- Utilisez le AWS CLI pour ajouter la configuration d'action pour l'`CodeCommit`action et créer des ressources supplémentaires comme suit :
  - Utilisez l'`CodeCommit`exemple de configuration d'action CodeCommit référence de l'action source pour créer votre action comme indiqué dansCréation d'un pipeline (interface de ligne de commande).
  - La méthode de détection des modifications démarre par défaut le pipeline en interrogeant la source. Vous devez désactiver les contrôles périodiques et créer la règle de détection des modifications manuellement. Utilisez l'une des méthodes suivantes : Création d'une EventBridge règle pour une CodeCommit source (console)Création d'une EventBridge règle pour une CodeCommit source (CLI), ouCréez une EventBridge règle pour une CodeCommit source (CloudFormation modèle).

Rubriques

- Création d'une EventBridge règle pour une CodeCommit source (console)
- Création d'une EventBridge règle pour une CodeCommit source (CLI)
- Créez une EventBridge règle pour une CodeCommit source (CloudFormation modèle)

Création d'une EventBridge règle pour une CodeCommit source (console)

> **Important**
>
> Si vous utilisez la console pour créer ou modifier votre pipeline, votre EventBridge règle est créée pour vous.

Pour créer une EventBridge règle à utiliser dans les CodePipeline opérations

1. Ouvrez la EventBridge console Amazon à l'adresse https://console.aws.amazon.com/events/.
2. Dans le panneau de navigation, choisissez Rules. Laissez le bus par défaut sélectionné ou choisissez un bus d'événements. Choisissez Créer une règle.
3. Dans Nom, entrez le nom de votre règle.

4. Sous Type de règle, choisissez Règle avec un modèle d'événement. Choisissez Suivant.
5. Sous Source de l'événement, choisissez AWS des événements ou des événements EventBridge partenaires.
6. Dans Exemple de type d'événement, choisissez AWS événements.
7. Dans Exemples d'événements, saisissez CodeCommit le mot clé sur lequel vous souhaitez filtrer. Choisissez CodeCommit Repository State Change.
8. Sous Méthode de création, choisissez Customer pattern (éditeur JSON).

   Collez le modèle d'événement fourni ci-dessous. L'exemple suivant est un exemple de modèle d'événement CodeCommit dans la fenêtre Événement pour un référentiel `MyTestRepo` avec une branche nommée `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. Dans Targets, choisissez CodePipeline.
10. Entrez l'ARN du pipeline pour le pipeline à démarrer selon cette règle.

> **Note**
>
> Vous trouverez l'ARN de pipeline dans la sortie des métadonnées après avoir exécuté la commande get-pipeline. L'ARN de pipeline est élaboré dans ce format :

arn:aws:codepipeline : : *region account pipeline-name*
Exemple d'ARN de pipeline :
`arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

11. Pour créer ou spécifier un rôle de service IAM qui accorde EventBridge des autorisations pour appeler la cible associée à votre EventBridge règle (dans ce cas, la cible est CodePipeline) :

    - Choisissez Créer un nouveau rôle pour cette ressource spécifique afin de créer un rôle de service qui donne EventBridge les autorisations nécessaires pour démarrer les exécutions de votre pipeline.
    - Choisissez Utiliser le rôle existant pour saisir un rôle de service qui autorise EventBridge le démarrage des exécutions de votre pipeline.

12. (Facultatif) Pour spécifier des remplacements de source avec un ID d'image spécifique, utilisez le transformateur d'entrée pour transmettre les données sous forme de paramètres JSON. Vous pouvez également utiliser le transformateur d'entrée pour transmettre les variables du pipeline.

    - Développer Additional settings (Paramètres supplémentaires).

      Sous Configurer l'entrée cible, choisissez Configurer le transformateur d'entrée.

      Dans la fenêtre de dialogue, choisissez Enter my own. Dans la zone Chemin d'entrée, tapez les paires clé-valeur suivantes.

      ```
      {"revisionValue": "$.detail.image-digest"}
      ```

    - Dans la zone Modèle, tapez les paires clé-valeur suivantes.

      ```
      {
          "sourceRevisions": [
              {
                  "actionName": "Source",
                  "revisionType": "COMMIT_ID",
                  "revisionValue": "<revisionValue>"
              }
          ],
          "variables": [
              {
                  "name": "Branch_Name",
                  "value": "value"
              }
      ```

```
    ]
}
```

- Choisissez Confirmer.

13. Choisissez Suivant.
14. Sur la page Tags, choisissez Next.
15. Sur la page Réviser et créer, passez en revue la configuration des règles. Si la règle vous convient, choisissez Créer une règle.

Création d'une EventBridge règle pour une CodeCommit source (CLI)

Appelez la commande put-rule, en spécifiant les éléments suivants :

- Un nom qui identifie de façon unique la règle que vous créez. Ce nom doit être unique pour tous les pipelines que vous créez et qui CodePipeline sont associés à votre AWS compte.
- Le modèle d'événement pour la source et les champs de détails utilisés par la règle. Pour plus d'informations, consultez Amazon EventBridge et Event Patterns.

Pour créer une EventBridge règle avec CodeCommit comme source d'événement et CodePipeline comme cible

1. Ajoutez des autorisations EventBridge à utiliser CodePipeline pour invoquer la règle. Pour plus d'informations, consultez la section Utilisation de politiques basées sur les ressources pour Amazon. EventBridge

   a. Utilisez l'exemple suivant pour créer la politique de confiance qui permet EventBridge d'assumer le rôle de service. Nommez la stratégie d'approbation `trustpolicyforEB.json`.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "events.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Utilisez la commande suivante pour créer le rôle `Role-for-MyRule` et attachez la stratégie d'approbation.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Créez le JSON de stratégie d'autorisations comme indiqué dans cet exemple pour le pipeline nommé `MyFirstPipeline`. Nommez la stratégie d'autorisations `permissionspolicyforEB.json`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilisez la commande suivante pour attacher au rôle `Role-for-MyRule` la stratégie d'autorisations `CodePipeline-Permissions-Policy-for-EB`.

Pourquoi est-ce que j'effectue cette modification ? L'ajout de cette politique au rôle crée des autorisations pour EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Appelez la commande put-rule et incluez les paramètres `--name`, `--event-pattern` et `--role-arn`.

   Pourquoi est-ce que j'effectue cette modification ? Cette commande permet à CloudFormation de créer l'événement.

   L'exemple de commande suivant crée une règle nommée `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. Pour ajouter CodePipeline en tant que cible, appelez la put-targets commande et incluez les paramètres suivants :

   - Le paramètre `--rule` s'utilise avec le la règle `rule_name` que vous avez créée à l'aide de la commande put-rule.
   - Le paramètre `--targets` s'utilise avec l'ID de liste `Id` de la cible figurant dans la liste des cibles et l'ARN du pipeline cible.

   L'exemple de commande suivant spécifie que pour la règle appelée `MyCodeCommitRepoRule`, l'`Id` cible est composé du numéro un, ce qui indique qu'il s'agit de la règle 1 dans une liste de cibles pour la règle. L'exemple de commande spécifie également un exemple d'ARN pour le pipeline. Le pipeline démarre lorsque des modifications sont effectuées dans le référentiel.

```
aws events put-targets --rule MyCodeCommitRepoRule --targets
 Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

4. (Facultatif) Pour configurer un transformateur d'entrée avec des remplacements de source pour un ID d'image spécifique, utilisez le JSON suivant dans votre commande CLI. L'exemple suivant configure une dérogation dans laquelle :

   - `SourceDans cet exemple``actionName`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.

- `COMMIT_ID`Dans cet exemple`revisionType`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
- Dans cet exemple`revisionValue`, < *`revisionValue`* > est dérivé de la variable d'événement source.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "pipeline-ARN",
            "InputTransformer": {
                "sourceRevisions": {
                    "actionName": "Source",
                    "revisionType": "COMMIT_ID",
                    "revisionValue": "<revisionValue>"
                },
                "variables": [
                    {
                        "name": "Branch_Name",
                        "value": "value"
                    }
                ]
            }
        }
    ]
}
```

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Lorsque vous créez un pipeline avec cette méthode, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut s'il n'est pas explicitement défini sur Faux. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

1. Exécutez la commande get-pipeline pour copier la structure de pipeline dans un fichier JSON. Par exemple, pour un pipeline nommé `MyFirstPipeline`, exécutez la commande suivante :

```
aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
```

   Cette commande ne renvoie rien, mais le fichier que vous avez créé doit apparaître dans le répertoire où vous avez exécuté la commande.

2. Ouvrez le fichier JSON dans un éditeur de texte brut et modifiez l'étape source en remplaçant la valeur du paramètre `PollForSourceChanges` par `false`, comme illustré dans cet exemple.

   Pourquoi est-ce que j'effectue cette modification ? Le remplacement de la valeur de ce paramètre par `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

```
"configuration": {
    "PollForSourceChanges": "false",
    "BranchName": "main",
    "RepositoryName": "MyTestRepo"
},
```

3. Si vous utilisez la structure de pipeline extraite à l'aide de la commande get-pipeline, supprimez les lignes `metadata` du fichier JSON. Sinon, la commande update-pipeline ne peut pas l'utiliser. Supprimez les lignes `"metadata": { }` et les champs `"updated"`, `"created"` et `"pipelineARN"`.

   Par exemple, supprimez les lignes suivantes de la structure :

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

   Enregistrez le fichier.

4. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline :

> **Important**
>
> N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

Cette commande affiche toute la structure du pipeline mise à jour.

> **Note**
>
> La commande update-pipeline interrompt le pipeline. Si une révision est exécutée dans le pipeline lorsque vous exécutez la commande update-pipeline celle-ci est interrompue. Vous devez lancer manuellement le pipeline pour exécuter cette révision dans le pipeline mis à jour. Utilisez la commande **`start-pipeline-execution`** pour démarrer manuellement votre pipeline.

Créez une EventBridge règle pour une CodeCommit source (CloudFormation modèle)

CloudFormation Pour créer une règle, mettez à jour votre modèle comme indiqué ici.

Pour mettre à jour votre pipeline CloudFormation modèle et création de EventBridge règle

1. Dans le modèle, ci-dessous`Resources`, utilisez la `AWS::IAM::Role` CloudFormation ressource pour configurer le rôle IAM qui permet à votre événement de démarrer votre pipeline. Cette entrée crée un rôle qui utilise deux stratégies :

   - La première stratégie autorise le rôle à être endossé.
   - La deuxième stratégie fournit des autorisations pour démarrer le pipeline.

Pourquoi est-ce que j'effectue cette modification ? L'ajout de la `AWS::IAM::Role` ressource permet CloudFormation de créer des autorisations pour EventBridge. Cette ressource est ajoutée à votre CloudFormation pile.

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. Dans le modèle, ci-dessous`Resources`, utilisez la `AWS::Events::Rule` CloudFormation ressource pour ajouter une EventBridge règle. Ce modèle d'événement crée un événement qui surveille les modifications push apportées à votre référentiel. Lorsqu'un changement d'état du

référentiel est EventBridge détecté, la règle est invoquée `StartPipelineExecution` sur votre pipeline cible.

Pourquoi est-ce que je fais ce changement ? L'ajout de la `AWS::Events::Rule` ressource CloudFormation permet de créer l'événement. Cette ressource est ajoutée à votre CloudFormation pile.

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. (Facultatif) Pour configurer un transformateur d'entrée avec des remplacements de source pour un ID d'image spécifique, utilisez l'extrait YAML suivant. L'exemple suivant configure une dérogation dans laquelle :

   - `SourceDans cet exemple actionName`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - `COMMIT_IDDans cet exemple revisionType`, il s'agit de la valeur dynamique, définie lors de la création du pipeline, qui n'est pas dérivée de l'événement source.
   - Dans cet exemple`revisionValue`, < *`revisionValue`* > est dérivé de la variable d'événement source.
   - Les variables de sortie pour `BranchName` et `Value` sont spécifiées.

```
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
    sourceRevisions:
      actionName: Source
      revisionType: COMMIT_ID
      revisionValue: <revisionValue>
    variables:
    - name: BranchName
      value: value
```

4. Enregistrez le modèle mis à jour sur votre ordinateur local, puis ouvrez la CloudFormation console.
5. Choisissez votre pile, puis Créer un jeu de modifications pour la pile actuelle.
6. Chargez le modèle mis à jour, puis affichez les modifications répertoriées dans CloudFormation. Il s'agit des modifications apportées à la pile. Vos nouvelles ressources doivent figurer dans la liste.
7. Sélectionnez Execute (Exécuter).

Pour modifier le PollForSourceChanges paramètre de votre pipeline

> **Important**
>
> Dans de nombreux cas, le paramètre `PollForSourceChanges` prend la valeur Vrai par défaut lorsque vous créez un pipeline. Lorsque vous ajoutez la détection des modifications basée sur les événements, vous devez ajouter le paramètre à votre sortie et le configurer sur Faux pour désactiver l'interrogation. Sinon, votre pipeline démarre deux fois pour une seule modification de source. Pour en savoir plus, consultez Réglages valides pour le `PollForSourceChanges` paramètre.

- Dans le modèle, remplacez la valeur du paramètre `PollForSourceChanges` par `false`. Si vous n'avez pas inclus `PollForSourceChanges` dans votre définition de pipeline, ajoutez ce paramètre et définissez-le sur `false`.

Pourquoi est-ce que j'effectue cette modification ? Le remplacement de la valeur de ce paramètre par `false` désactive les vérifications périodiques, ce qui vous permet d'utiliser la détection des modifications basée sur les événements uniquement.

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
    }
  ]
},
```

Ajoutez des fournisseurs de sources tiers aux pipelines à l'aide de AWS CodeConnections

Vous pouvez utiliser la AWS CodePipeline console ou le AWS CLI pour connecter votre pipeline à des référentiels tiers.

> **Note**
>
> Lorsque vous utilisez la console pour créer ou modifier un pipeline, les ressources de détection des modifications sont créées pour vous. Si vous utilisez la AWS CLI pour créer le pipeline, vous devez créer vous-même les ressources supplémentaires. Pour plus d'informations, consultez CodeCommit actions à la source et EventBridge.

Rubriques

- DevOps Connexions Azure
- Connexions Bitbucket Cloud
- GitHub connexions
- GitHub Connexions aux serveurs d'entreprise
- GitLab.com connexions
- Connexions pour l' GitLab autogestion
- Utiliser une connexion partagée avec un autre compte

DevOps Connexions Azure

Les connexions vous permettent d'autoriser et d'établir des configurations qui associent votre fournisseur tiers à vos AWS ressources. Pour associer votre référentiel tiers en tant que source de votre pipeline, vous devez utiliser une connexion.

> **Note**
>
> Au lieu de créer ou d'utiliser une connexion existante dans votre compte, vous pouvez utiliser une connexion partagée entre un autre Compte AWS. Consultez Utiliser une connexion partagée avec un autre compte.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans les régions Asie-Pacifique (Hong Kong), Asie-Pacifique (Hyderabad), Asie-Pacifique (Jakarta), Asie-Pacifique (Melbourne), Asie-Pacifique (Osaka), Afrique (Le Cap), Moyen-Orient (Bahreïn), Moyen-Orient (Émirats arabes unis), Europe (Espagne), Europe (Zurich), Israël (Tel Aviv) ou (). AWS GovCloud US-West Pour faire référence aux autres actions disponibles, voirIntégrations de produits et de services avec CodePipeline. Pour les considérations relatives à cette action dans la région Europe (Milan), voir la note dansCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

Pour ajouter une action DevOps source Azure CodePipeline, vous pouvez choisir de :

- Utilisez l'assistant de création de pipeline ou la page d'action Modifier de la CodePipeline console pour choisir l'option du DevOps fournisseur Azure. Création d'une connexion à Bitbucket Cloud (console)Reportez-vous à la section pour ajouter l'action. La console vous permet de créer une ressource de connexions.
- Utilisez la CLI pour ajouter la configuration de l'`CreateSourceConnection`action avec le `Azure DevOps` fournisseur comme suit :
  - Pour créer vos ressources de connexions, reportez-vous Création d'une connexion à Azure DevOps (CLI) à la section Création d'une ressource de connexions avec la CLI.
  - Utilisez l'`CreateSourceConnection`exemple de configuration d'action CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées pour ajouter votre action comme indiqué dansCréation d'un pipeline (interface de ligne de commande).

> **Note**
>
> Vous pouvez également créer une connexion à l'aide de la console Developer Tools sous Paramètres. Voir Créer une connexion.

Avant de commencer :

- Vous devez avoir créé un compte auprès du fournisseur du référentiel tiers, tel qu'Azure DevOps

- Vous devez déjà avoir créé un référentiel de code tiers, tel qu'un DevOps référentiel Azure.

> **Note**
>
> DevOps Les connexions Azure fournissent uniquement l'accès aux référentiels détenus par le DevOps compte Azure qui a été utilisé pour créer la connexion.
> Pour installer la connexion, vous devez disposer des autorisations d'administration pour le compte Azure.

Rubriques

- Création d'une connexion à Azure DevOps (console)
- Création d'une connexion à Azure DevOps (CLI)

Création d'une connexion à Azure DevOps (console)

Suivez ces étapes pour utiliser la CodePipeline console afin d'ajouter une action de connexion pour votre DevOps référentiel Azure.

Étape 1 : créer ou modifier votre pipeline

Pour créer ou modifier votre pipeline

1. Connectez-vous à la CodePipeline console.
2. Choisissez l'une des options suivantes.
   - Choisissez de créer un pipeline. Suivez les étapes décrites dans Créer un pipeline pour terminer le premier écran et choisissez Next. Sur la page Source, sous Source Provider, sélectionnez Azure DevOps.
   - Choisissez de modifier un pipeline existant. Choisissez Modifier, puis sélectionnez Modifier l'étape. Choisissez d'ajouter ou de modifier votre action source. Sur la page Modifier l'action, sous Nom de l'action, entrez le nom de votre action. Dans Action provider, choisissez Azure DevOps.
3. Effectuez l'une des actions suivantes :

- Sous Connexion, si vous n'avez pas encore créé de connexion avec votre fournisseur, choisissez Connect to Azure DevOps. Passez à l'étape 2 : créer une connexion à Azure DevOps.
- Sous Connexion, si vous avez déjà créé une connexion avec votre fournisseur, choisissez-la. Passez à l'étape 3 : Enregistrer l'action source pour votre connexion.

Étape 2 : créer une connexion à Azure DevOps

Pour créer une connexion à Azure DevOps

1. Pour créer une connexion à un DevOps référentiel Azure, sous Sélectionnez un fournisseur, choisissez Azure DevOps. Dans Connection name (Nom de la connexion), saisissez le nom de la connexion que vous souhaitez créer. Choisissez Connect to Azure DevOps, puis passez à l'étape 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. Sur la page des DevOps paramètres Connect to Azure, le nom de votre connexion s'affiche.

   Sous DevOps Applications Azure, choisissez une installation d'application ou choisissez Installer une nouvelle application pour en créer une.

   > **Note**
   >
   > Vous n'installez l'application qu'une seule fois pour chaque DevOps compte Azure. Si vous avez déjà installé l'application Connector, choisissez-la et passez à la dernière étape de cette section.

3. Si la page de connexion de Microsoft s'affiche, connectez-vous avec vos informations d'identification, puis choisissez de continuer.
4. Sur la page d'installation de l'application, un message indique que l'application du connecteur essaie de se connecter à votre DevOps compte Azure.

   Choisissez Grant access (Accorder l'accès).
5. Sur la page de connexion, l'identifiant de connexion de votre nouvelle installation est affiché. Choisissez Se connecter. La connexion créée s'affiche dans la liste des connexions.

Étape 3 : enregistrer votre action DevOps source Azure

Suivez ces étapes dans l'assistant ou sur la page Modifier l'action pour enregistrer votre action source avec vos informations de connexion.

Pour terminer et enregistrer votre action source avec votre connexion

1. Dans Nom du référentiel, choisissez le nom de votre référentiel tiers.
2. Sous Déclencheurs du pipeline, vous pouvez ajouter des déclencheurs si votre action est une AWS CodeConnections action. Pour configurer la configuration des déclencheurs du pipeline et pour éventuellement filtrer à l'aide de déclencheurs, reportez-vous à la sectionAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ».
3. Dans Output artifact format (Format d'artefact de sortie), vous devez choisir le format de vos artefacts.

   - Pour stocker les artefacts de sortie de l' DevOps action Azure à l'aide de la méthode par défaut, choisissez CodePipeline default. L'action accède aux fichiers depuis le référentiel et stocke les artefacts dans un fichier ZIP dans le magasin d'artefacts du pipeline.

- Pour stocker un fichier JSON contenant une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git, choisissez Full clone (Clone complet). Cette option ne peut être utilisée que par des actions CodeBuild en aval.

  Si vous choisissez cette option, vous devrez mettre à jour les autorisations associées à votre rôle de service de CodeBuild projet, comme indiqué dansAjoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com.

4. Choisissez Suivant dans l'assistant ou Enregistrer sur la page d'action Modifier.

Création d'une connexion à Azure DevOps (CLI)

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour créer une connexion.

Pour ce faire, utilisez la commande create-connection.

> **Important**
>
> Une connexion créée via le AWS CLI ou AWS CloudFormation est en `PENDING` statut par défaut. Après avoir créé une connexion avec la CLI CloudFormation, utilisez la console pour modifier la connexion afin de définir son état`AVAILABLE`.

Pour créer une connexion

1. Ouvrez une invite de terminal (Linux, macOS ou Unix) ou de commande (Windows). Utilisez le AWS CLI pour exécuter la create-connection commande, en spécifiant le `--provider-type` et `--connection-name` pour votre connexion. Dans cet exemple, le nom du fournisseur tiers est `AzureDevOps` et le nom de connexion spécifié est `MyConnection`.

   ```
   aws codeconnections create-connection --provider-type AzureDevOps --connection-name
    MyConnection
   ```

   En cas de succès, cette commande renvoie les informations ARN de connexion semblables à ce qui suit.

   ```
   {
   ```

```
    "ConnectionArn": "arn:aws:codeconnections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Utilisez la console pour terminer la connexion. Pour plus d'informations, voir Mettre à jour une connexion en attente.
3. Le pipeline détecte par défaut les modifications lors de l'envoi du code vers le référentiel des sources de connexion. Pour configurer la configuration du déclencheur du pipeline pour le lancement manuel ou pour les balises Git, effectuez l'une des opérations suivantes :

   - Pour configurer la configuration du déclencheur du pipeline de manière à ce qu'elle commence par un déverrouillage manuel uniquement, ajoutez la ligne suivante à la configuration :

     ```
     "DetectChanges": "false",
     ```

   - Pour configurer la configuration des déclencheurs du pipeline afin de filtrer avec des déclencheurs, voir plus de détails dansAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ». Par exemple, ce qui suit ajoute des balises Git au niveau du pipeline de la définition JSON du pipeline. Dans cet exemple, `release-v0` et `release-v1` sont les balises Git à inclure et `release-v2` les balises Git à exclure.

     ```
     "triggers": [
                 {
                     "providerType": "CodeStarSourceConnection",
                     "gitConfiguration": {
                         "sourceActionName": "Source",
                         "push": [
                             {
                                 "tags": {
                                     "includes": [
                                         "release-v0", "release-v1"
                                     ],
                                     "excludes": [
                                         "release-v2"
                                     ]
                                 }
                             }
                         ]
                     }
                 }
             ]
     ```

Connexions Bitbucket Cloud

Les connexions vous permettent d'autoriser et d'établir des configurations qui associent votre fournisseur tiers à vos AWS ressources. Pour associer votre référentiel tiers en tant que source de votre pipeline, vous devez utiliser une connexion.

> **Note**
>
> Au lieu de créer ou d'utiliser une connexion existante dans votre compte, vous pouvez utiliser une connexion partagée entre un autre Compte AWS. Consultez Utiliser une connexion partagée avec un autre compte.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans les régions Asie-Pacifique (Hong Kong), Asie-Pacifique (Hyderabad), Asie-Pacifique (Jakarta), Asie-Pacifique (Melbourne), Asie-Pacifique (Osaka), Afrique (Le Cap), Moyen-Orient (Bahreïn), Moyen-Orient (Émirats arabes unis), Europe (Espagne), Europe (Zurich), Israël (Tel Aviv) ou (). AWS GovCloud US-West Pour faire référence aux autres actions disponibles, voirIntégrations de produits et de services avec CodePipeline. Pour les considérations relatives à cette action dans la région Europe (Milan), voir la note dansCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

Pour ajouter une action source Bitbucket Cloud CodePipeline, vous pouvez choisir de :

- Utilisez l'assistant de création de pipeline ou la page d'action Modifier de la CodePipeline console pour choisir l'option du fournisseur Bitbucket. Création d'une connexion à Bitbucket Cloud (console)Reportez-vous à la section pour ajouter l'action. La console vous permet de créer une ressource de connexions.

  > **Note**
  >
  > Vous pouvez créer des connexions à un référentiel Bitbucket Cloud. Les types de fournisseurs Bitbucket installés, tels que Bitbucket Server, ne sont pas pris en charge.

- Utilisez la CLI pour ajouter la configuration de l'`CreateSourceConnection`action avec le `Bitbucket` fournisseur comme suit :
  - Pour créer vos ressources de connexions, reportez-vous Création d'une connexion à Bitbucket Cloud (CLI) à la section Création d'une ressource de connexions avec la CLI.
  - Utilisez l'`CreateSourceConnection`exemple de configuration d'action CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées pour ajouter votre action comme indiqué dansCréation d'un pipeline (interface de ligne de commande).

> **Note**
>
> Vous pouvez également créer une connexion à l'aide de la console Developer Tools sous Paramètres. Voir Créer une connexion.

Avant de commencer :

- Vous devez avoir créé un compte auprès du fournisseur du référentiel tiers, tel que Bitbucket Cloud.
- Vous devez déjà avoir créé un référentiel de code tiers, tel qu'un référentiel Bitbucket Cloud.

> **Note**
>
> Les connexions Bitbucket Cloud fournissent uniquement l'accès aux référentiels détenus par le compte Bitbucket Cloud qui a été utilisé pour créer la connexion.
> Si l'application est installée dans un espace de travail Bitbucket Cloud, vous devez disposer des autorisations d'administration de l'espace de travail. Dans le cas contraire, l'option d'installation de l'application ne s'affichera pas.

Rubriques

- Création d'une connexion à Bitbucket Cloud (console)
- Création d'une connexion à Bitbucket Cloud (CLI)

Création d'une connexion à Bitbucket Cloud (console)

Suivez ces étapes pour utiliser la CodePipeline console afin d'ajouter une action de connexion à votre référentiel Bitbucket.

> **Note**
>
> Vous pouvez créer des connexions à un référentiel Bitbucket Cloud. Les types de fournisseurs Bitbucket installés, tels que Bitbucket Server, ne sont pas pris en charge.

Étape 1 : créer ou modifier votre pipeline

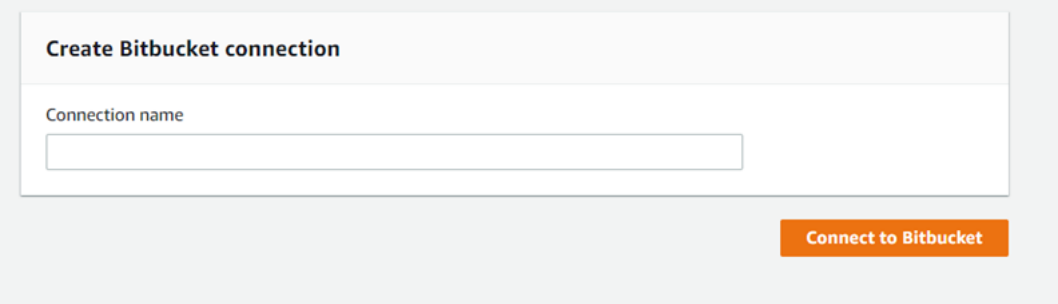
Pour créer ou modifier votre pipeline

1. Connectez-vous à la CodePipeline console.
2. Choisissez l'une des options suivantes.
   - Choisissez de créer un pipeline. Suivez les étapes décrites dans Créer un pipeline pour terminer le premier écran et choisissez Next. Sur la page Source, sous Source Provider, sélectionnez Bitbucket.
   - Choisissez de modifier un pipeline existant. Choisissez Modifier, puis sélectionnez Modifier l'étape. Choisissez d'ajouter ou de modifier votre action source. Sur la page Modifier l'action, sous Nom de l'action, entrez le nom de votre action. Dans Action provider, choisissez Bitbucket.
3. Effectuez l'une des actions suivantes :
   - Sous Connexion, si vous n'avez pas encore créé de connexion avec votre fournisseur, choisissez Connect to Bitbucket. Passez à l'étape 2 : créer une connexion à Bitbucket.
   - Sous Connexion, si vous avez déjà créé une connexion avec votre fournisseur, choisissez-la. Passez à l'étape 3 : Enregistrer l'action source pour votre connexion.

Étape 2 : créer une connexion à Bitbucket Cloud

Pour créer une connexion à Bitbucket Cloud

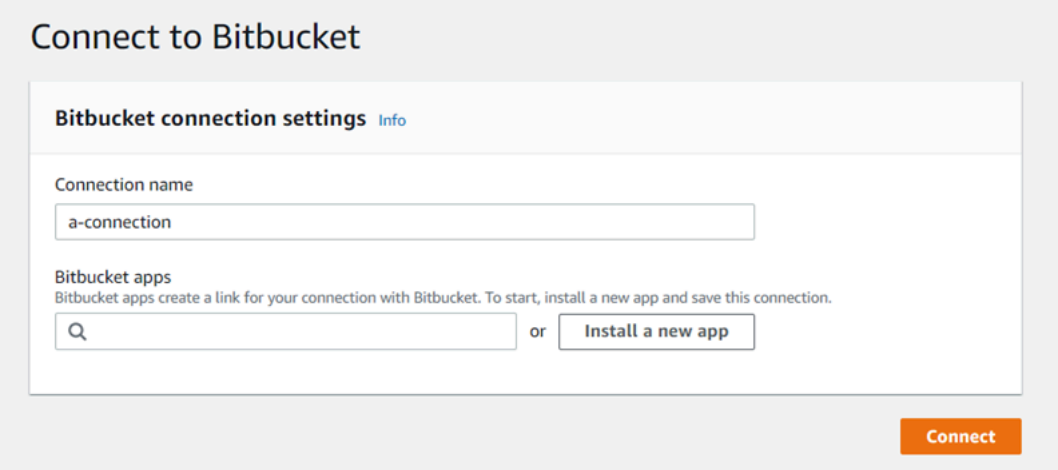
1. Sur la page des paramètres de Connect to Bitbucket, entrez le nom de votre connexion et choisissez Connect to Bitbucket.



Le champ des applications Bitbucket apparaît.

2. Sous Bitbucket apps (Applications Bitbucket), choisissez une installation d'application ou choisissez Install a new app (Installer une nouvelle application) pour en créer une.

> **Note**
>
> Vous n'installez l'application qu'une seule fois pour chaque espace de travail ou compte Bitbucket Cloud. Si vous avez déjà installé l'application Bitbucket, choisissez-la et passez à l'étape 4.

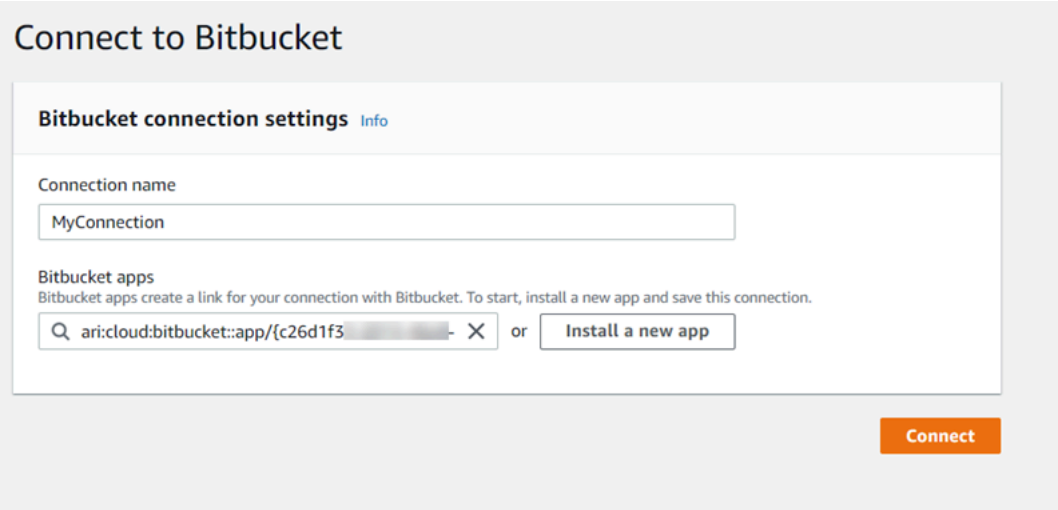


3. Si la page de connexion de Bitbucket Cloud s'affiche, connectez-vous avec vos informations d'identification, puis choisissez de continuer.
4. Sur la page d'installation de l'application, un message indique que l' AWS CodeStar application essaie de se connecter à votre compte Bitbucket.

   Si vous utilisez un espace de travail Bitbucket, modifiez l'option Autoriser pour pour l'espace de travail. Seuls les espaces de travail auxquels vous avez accès en tant qu'administrateur s'affichent.

Choisissez Grant access (Accorder l'accès).

5. Dans Bitbucket apps (Applications Bitbucket), l'ID de connexion de votre nouvelle installation s'affiche. Choisissez Se connecter. La connexion créée s'affiche dans la liste des connexions.



Étape 3 : enregistrer votre action source Bitbucket Cloud

Suivez ces étapes dans l'assistant ou sur la page Modifier l'action pour enregistrer votre action source avec vos informations de connexion.

Pour terminer et enregistrer votre action source avec votre connexion

1. Dans Nom du référentiel, choisissez le nom de votre référentiel tiers.
2. Sous Déclencheurs du pipeline, vous pouvez ajouter des déclencheurs si votre action est une AWS CodeConnections action. Pour configurer la configuration des déclencheurs du pipeline et pour éventuellement filtrer à l'aide de déclencheurs, reportez-vous à la sectionAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ».
3. Dans Output artifact format (Format d'artefact de sortie), vous devez choisir le format de vos artefacts.

   - Pour stocker les artefacts de sortie issus de l'action Bitbucket Cloud à l'aide de la méthode par défaut, choisissez CodePipeline par défaut. L'action accède aux fichiers depuis le référentiel Bitbucket Cloud et stocke les artefacts dans un fichier ZIP dans le magasin d'artefacts du pipeline.

- Pour stocker un fichier JSON contenant une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git, choisissez Full clone (Clone complet). Cette option ne peut être utilisée que par des actions CodeBuild en aval.

  Si vous choisissez cette option, vous devrez mettre à jour les autorisations associées à votre rôle de service de CodeBuild projet, comme indiqué dansAjoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com.

4. Choisissez Suivant dans l'assistant ou Enregistrer sur la page d'action Modifier.

Création d'une connexion à Bitbucket Cloud (CLI)

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour créer une connexion.

> **Note**
>
> Vous pouvez créer des connexions à un référentiel Bitbucket Cloud. Les types de fournisseurs Bitbucket installés, tels que Bitbucket Server, ne sont pas pris en charge.

Pour ce faire, utilisez la commande create-connection.

> **Important**
>
> Une connexion créée via le AWS CLI ou AWS CloudFormation est en `PENDING` statut par défaut. Après avoir créé une connexion avec la CLI CloudFormation, utilisez la console pour modifier la connexion afin de définir son état`AVAILABLE`.

Pour créer une connexion

1. Ouvrez une invite de terminal (Linux, macOS ou Unix) ou de commande (Windows). Utilisez le AWS CLI pour exécuter la create-connection commande, en spécifiant le `--provider-type` et `--connection-name` pour votre connexion. Dans cet exemple, le nom du fournisseur tiers est `Bitbucket` et le nom de connexion spécifié est `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type Bitbucket --connection-name MyConnection
   ```

En cas de succès, cette commande renvoie les informations ARN de connexion semblables à ce qui suit.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Utilisez la console pour terminer la connexion. Pour plus d'informations, voir Mettre à jour une connexion en attente.
3. Le pipeline détecte par défaut les modifications lors de l'envoi du code vers le référentiel des sources de connexion. Pour configurer la configuration du déclencheur du pipeline pour le lancement manuel ou pour les balises Git, effectuez l'une des opérations suivantes :

   - Pour configurer la configuration du déclencheur du pipeline afin qu'elle commence par un déverrouillage manuel uniquement, ajoutez la ligne suivante à la configuration :

     ```
     "DetectChanges": "false",
     ```

   - Pour configurer la configuration des déclencheurs du pipeline afin de filtrer avec des déclencheurs, voir plus de détails dansAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ». Par exemple, ce qui suit ajoute des balises Git au niveau du pipeline de la définition JSON du pipeline. Dans cet exemple, `release-v0` et `release-v1` sont les balises Git à inclure et `release-v2` les balises Git à exclure.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
     ```

```
                    }
                ]
            }
        }
    ]
```

GitHub connexions

Vous utilisez les connexions pour autoriser et établir des configurations qui associent votre fournisseur tiers à vos AWS ressources.

> **Note**
>
> Au lieu de créer ou d'utiliser une connexion existante dans votre compte, vous pouvez utiliser une connexion partagée entre un autre Compte AWS. Consultez Utiliser une connexion partagée avec un autre compte.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans les régions Asie-Pacifique (Hong Kong), Asie-Pacifique (Hyderabad), Asie-Pacifique (Jakarta), Asie-Pacifique (Melbourne), Asie-Pacifique (Osaka), Afrique (Le Cap), Moyen-Orient (Bahreïn), Moyen-Orient (Émirats arabes unis), Europe (Espagne), Europe (Zurich), Israël (Tel Aviv) ou (). AWS GovCloud US-West Pour faire référence aux autres actions disponibles, voirIntégrations de produits et de services avec CodePipeline. Pour les considérations relatives à cette action dans la région Europe (Milan), voir la note dansCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

Pour ajouter une action source pour votre référentiel GitHub ou GitHub celui de votre référentiel Enterprise Cloud CodePipeline, vous pouvez choisir de :

- Utilisez l'assistant de création de pipeline ou la page d'action Modifier de la CodePipeline console pour choisir l'option fournisseur GitHub (via GitHub l'application). Consultez Création d'une connexion à GitHub (console) pour ajouter l'action. La console vous permet de créer une ressource de connexions.

> **Note**
>
> Pour un didacticiel expliquant comment ajouter une GitHub connexion et utiliser l'option de clonage complet dans votre pipeline pour cloner des métadonnées, consultezTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

- Utilisez la CLI pour ajouter la configuration de l'`CodeStarSourceConnection`action avec le `GitHub` fournisseur en suivant les étapes de la CLI indiquées dansCréation d'un pipeline (interface de ligne de commande).

> **Note**
>
> Vous pouvez également créer une connexion à l'aide de la console Developer Tools sous Paramètres. Voir Créer une connexion.

Avant de commencer :

- Vous devez avoir créé un compte auprès de GitHub.
- Vous devez déjà avoir créé un dépôt de GitHub code.
- Si votre rôle CodePipeline de service a été créé avant le 18 décembre 2019, vous devrez peut-être mettre à jour ses autorisations afin de pouvoir l'utiliser `codestar-connections:UseConnection` pour AWS CodeStar les connexions. Pour obtenir des instructions, consultez Ajouter des autorisations au rôle CodePipeline de service.

> **Note**
>
> Pour créer la connexion, vous devez être le propriétaire de GitHub l'organisation. Pour les référentiels qui ne font pas partie d'une organisation, vous devez être le propriétaire du référentiel.

Rubriques

- Création d'une connexion à GitHub (console)
- Création d'une connexion à GitHub (CLI)

Création d'une connexion à GitHub (console)

Suivez ces étapes pour utiliser la CodePipeline console afin d'ajouter une action de connexion pour votre référentiel GitHub ou GitHub celui d'Enterprise Cloud.

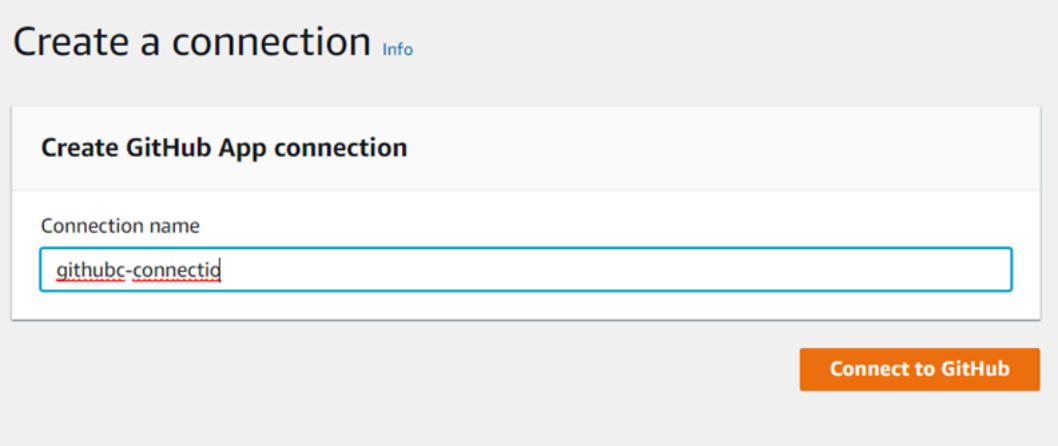
> **Note**
>
> Au cours de ces étapes, vous pouvez sélectionner des référentiels spécifiques sous Accès au référentiel. Les référentiels non sélectionnés ne seront ni accessibles ni visibles par CodePipeline.

Étape 1 : créer ou modifier votre pipeline

1. Connectez-vous à la CodePipeline console.
2. Choisissez l'une des options suivantes.
   - Choisissez de créer un pipeline. Suivez les étapes décrites dans Créer un pipeline pour terminer le premier écran et choisissez Next. Sur la page Source, sous Source Provider, sélectionnez GitHub (via GitHub App).
   - Choisissez de modifier un pipeline existant. Choisissez Modifier, puis sélectionnez Modifier l'étape. Choisissez d'ajouter ou de modifier votre action source. Sur la page Modifier l'action, sous Nom de l'action, entrez le nom de votre action. Dans Action provider, choisissez GitHub (via GitHub App).
3. Effectuez l'une des actions suivantes :
   - Sous Connexion, si vous n'avez pas encore créé de connexion avec votre fournisseur, choisissez Se connecter à GitHub. Passez à l'étape 2 : créer une connexion à GitHub.
   - Sous Connexion, si vous avez déjà créé une connexion avec votre fournisseur, choisissez-la. Passez à l'étape 3 : enregistrer l'action source pour votre connexion.

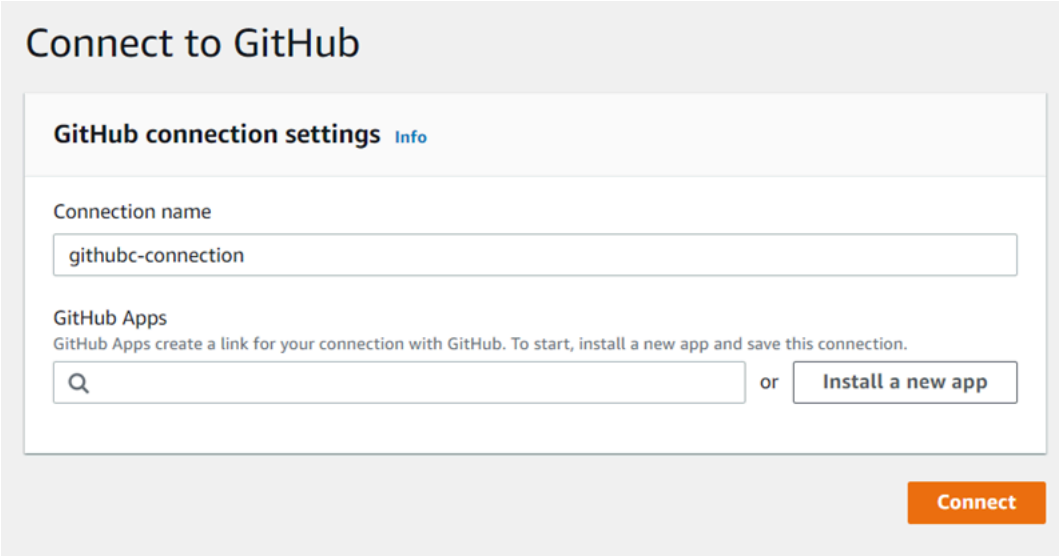
Étape 2 : créer une connexion avec GitHub

Une fois que vous avez choisi de créer la connexion, la GitHub page Connect to apparaît.



Pour créer une connexion avec GitHub

1. Dans les paramètres de GitHub connexion, le nom de votre connexion apparaît dans Nom de la connexion. Choisissez Connect to GitHub. La page de demande d'accès s'affiche.
2. Choisissez Autoriser le AWS connecteur pour GitHub. La page de connexion affiche et affiche le champ GitHub Applications.



3. Sous GitHub Applications, choisissez une installation d'application ou choisissez Installer une nouvelle application pour en créer une.

   Installez une application pour toutes vos connexions à un fournisseur particulier. Si vous avez déjà installé le AWS Connector for GitHub app, choisissez-le et ignorez cette étape.

> **Note**
>
> Si vous souhaitez créer un jeton d'accès utilisateur, assurez-vous d'avoir déjà installé le AWS connecteur pour l' GitHub application, puis laissez le champ Installation de l'application vide. AWS CodeConnections utilisera le jeton d'accès utilisateur pour la connexion.

4. Sur la GitHub page Installer le AWS connecteur pour, choisissez le compte sur lequel vous souhaitez installer l'application.

> **Note**
>
> Vous n'installez l'application qu'une seule fois pour chaque GitHub compte. Si vous avez déjà installé l'application, vous pouvez choisir Configure (Configurer) pour passer à une page de modification pour l'installation de votre application, ou vous pouvez utiliser le bouton Précédent pour revenir à la console.

5. Sur la GitHub page Installer le AWS connecteur pour, laissez les valeurs par défaut et choisissez Installer.
6. Sur la GitHub page Connect to, l'ID de connexion de votre nouvelle installation apparaît dans GitHub Apps. Choisissez Se connecter.

Étape 3 : Enregistrez votre action GitHub source

Suivez ces étapes sur la page Modifier l'action pour enregistrer votre action source avec vos informations de connexion.

Pour enregistrer votre action GitHub source

1. Dans Nom du référentiel, choisissez le nom de votre référentiel tiers.
2. Sous Déclencheurs du pipeline, vous pouvez ajouter des déclencheurs si votre action est une AWS CodeConnections action. Pour configurer la configuration des déclencheurs du pipeline et pour éventuellement filtrer à l'aide de déclencheurs, reportez-vous à la sectionAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ».
3. Dans Output artifact format (Format d'artefact de sortie), vous devez choisir le format de vos artefacts.

- Pour stocker les artefacts de sortie de l' GitHub action à l'aide de la méthode par défaut, choisissez CodePipeline par défaut. L'action accède aux fichiers depuis le GitHub référentiel et stocke les artefacts dans un fichier ZIP dans le magasin d'artefacts du pipeline.
- Pour stocker un fichier JSON contenant une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git, choisissez Full clone (Clone complet). Cette option ne peut être utilisée que par des actions CodeBuild en aval.

  Si vous choisissez cette option, vous devrez mettre à jour les autorisations associées à votre rôle de service de CodeBuild projet, comme indiqué dansAjoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com. Pour consulter un didacticiel expliquant comment utiliser l'option de clonage complet, voirTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

4. Choisissez Suivant dans l'assistant ou Enregistrer sur la page d'action Modifier.

Création d'une connexion à GitHub (CLI)

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour créer une connexion.

Pour ce faire, utilisez la commande create-connection.

> **Important**
>
> Une connexion créée via le AWS CLI ou AWS CloudFormation est en `PENDING` statut par défaut. Après avoir créé une connexion avec la CLI CloudFormation, utilisez la console pour modifier la connexion afin de définir son état`AVAILABLE`.

Pour créer une connexion

1. Ouvrez une invite de terminal (Linux, macOS ou Unix) ou de commande (Windows). Utilisez le AWS CLI pour exécuter la create-connection commande, en spécifiant le `--provider-type` et `--connection-name` pour votre connexion. Dans cet exemple, le nom du fournisseur tiers est `GitHub` et le nom de connexion spécifié est `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type GitHub --connection-name
    MyConnection
   ```

En cas de succès, cette commande renvoie les informations ARN de connexion semblables à ce qui suit.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Utilisez la console pour terminer la connexion. Pour plus d'informations, voir Mettre à jour une connexion en attente.
3. Le pipeline détecte par défaut les modifications lors de l'envoi du code vers le référentiel des sources de connexion. Pour configurer la configuration du déclencheur du pipeline pour le lancement manuel ou pour les balises Git, effectuez l'une des opérations suivantes :

   - Pour configurer la configuration du déclencheur du pipeline de manière à ce qu'elle commence par un déverrouillage manuel uniquement, ajoutez la ligne suivante à la configuration :

     ```
     "DetectChanges": "false",
     ```

   - Pour configurer la configuration des déclencheurs du pipeline afin de filtrer avec des déclencheurs, voir plus de détails dansAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ». Par exemple, ce qui suit ajoute au niveau du pipeline de la définition JSON du pipeline. Dans cet exemple, `release-v0` et `release-v1` sont les balises Git à inclure et `release-v2` les balises Git à exclure.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
     ```

```
                    }
                ]
            }
        }
    ]
```

GitHub Connexions aux serveurs d'entreprise

Les connexions vous permettent d'autoriser et d'établir des configurations qui associent votre fournisseur tiers à vos AWS ressources. Pour associer votre référentiel tiers en tant que source de votre pipeline, vous utilisez une connexion.

> **Note**
>
> Au lieu de créer ou d'utiliser une connexion existante dans votre compte, vous pouvez utiliser une connexion partagée entre un autre Compte AWS. Consultez Utiliser une connexion partagée avec un autre compte.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans les régions Asie-Pacifique (Hong Kong), Asie-Pacifique (Hyderabad), Asie-Pacifique (Jakarta), Asie-Pacifique (Melbourne), Asie-Pacifique (Osaka), Afrique (Le Cap), Moyen-Orient (Bahreïn), Moyen-Orient (Émirats arabes unis), Europe (Espagne), Europe (Zurich), Israël (Tel Aviv) ou (). AWS GovCloud US-West Pour faire référence aux autres actions disponibles, voirIntégrations de produits et de services avec CodePipeline. Pour les considérations relatives à cette action dans la région Europe (Milan), voir la note dansCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

Pour ajouter une action source GitHub Enterprise Server CodePipeline, vous pouvez choisir de :

- Utilisez l'assistant de création de pipeline ou la page d'action Modifier de la CodePipeline console pour choisir l'option du fournisseur de serveurs GitHub d'entreprise. Consultez Création d'une connexion à GitHub Enterprise Server (console) pour ajouter l'action. La console vous permet de créer une ressource hôte et une ressource de connexions.

- Utilisez la CLI pour ajouter la configuration de l'`CreateSourceConnection`action avec le `GitHubEnterpriseServer` fournisseur et créer vos ressources :
  - Pour créer vos ressources de connexion, reportez-vous Création d'un hôte et d'une connexion à GitHub Enterprise Server (CLI) à la section Création d'une ressource hôte et d'une ressource de connexions avec la CLI.
  - Utilisez l'`CreateSourceConnection`exemple de configuration d'action CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées pour ajouter votre action comme indiqué dansCréation d'un pipeline (interface de ligne de commande).

> **Note**
>
> Vous pouvez également créer une connexion à l'aide de la console Developer Tools sous Paramètres. Voir Créer une connexion.

Avant de commencer :

- Vous devez avoir créé un compte auprès GitHub d'Enterprise Server et installé l'instance GitHub d'Enterprise Server sur votre infrastructure.

  > **Note**
  >
  > Chaque VPC ne peut être associé qu'à un seul hôte (instance de serveur GitHub d'entreprise) à la fois.

- Vous devez déjà avoir créé un référentiel de code avec GitHub Enterprise Server.

Rubriques

- Création d'une connexion à GitHub Enterprise Server (console)
- Création d'un hôte et d'une connexion à GitHub Enterprise Server (CLI)

Création d'une connexion à GitHub Enterprise Server (console)

Suivez ces étapes pour utiliser la CodePipeline console afin d'ajouter une action de connexion pour votre référentiel GitHub Enterprise Server.

> **Note**
>
> GitHub Les connexions Enterprise Server fournissent uniquement l'accès aux référentiels détenus par le compte GitHub Enterprise Server qui a été utilisé pour créer la connexion.

Avant de commencer :

Pour une connexion hôte à GitHub Enterprise Server, vous devez avoir effectué les étapes de création d'une ressource hôte pour votre connexion. Consultez la section Gérer les hôtes pour les connexions.

Étape 1 : créer ou modifier votre pipeline

Pour créer ou modifier votre pipeline

1. Connectez-vous à la CodePipeline console.
2. Choisissez l'une des options suivantes.
   - Choisissez de créer un pipeline. Suivez les étapes décrites dans Créer un pipeline pour terminer le premier écran et choisissez Next. Sur la page Source, sous Fournisseur de source, choisissez GitHub Enterprise Server.
   - Choisissez de modifier un pipeline existant. Choisissez Modifier, puis sélectionnez Modifier l'étape. Choisissez d'ajouter ou de modifier votre action source. Sur la page Modifier l'action, sous Nom de l'action, entrez le nom de votre action. Dans Action provider, sélectionnez GitHub Enterprise Server.
3. Effectuez l'une des actions suivantes :
   - Sous Connexion, si vous n'avez pas encore créé de connexion avec votre fournisseur, choisissez Connect to GitHub Enterprise Server. Passez à l'étape 2 : créer une connexion au serveur GitHub d'entreprise.
   - Sous Connexion, si vous avez déjà créé une connexion avec votre fournisseur, choisissez-la. Passez à l'étape 3 : Enregistrer l'action source pour votre connexion.

Création d'une connexion à GitHub Enterprise Server

Une fois que vous avez choisi de créer la connexion, la page Connect to GitHub Enterprise Server s'affiche.

> **Important**
>
> AWS CodeConnections ne prend pas en charge la version 2.22.0 d' GitHub Enterprise Server en raison d'un problème connu dans cette version. Pour vous connecter, effectuez une mise à niveau vers la version 2.22.1 ou vers la dernière version disponible.

Pour vous connecter à GitHub Enterprise Server

1. Dans Connection name (Nom de la connexion), saisissez le nom de votre connexion.
2. Dans URL, saisissez le point de terminaison de votre serveur.

   > **Note**
   >
   > Si l'URL fournie a déjà été utilisée pour configurer un serveur d' GitHubentreprise pour une connexion, vous serez invité à choisir l'ARN de la ressource hôte créé précédemment pour ce point de terminaison.

3. Si vous avez lancé votre serveur dans un VPC Amazon et que vous souhaitez vous connecter à votre VPC, choisissez Use a VPC (Utilisation d'un VPC) et complétez ce qui suit.

   a. Dans ID du VPC, choisissez votre ID de VPC. Assurez-vous de choisir le VPC pour l'infrastructure sur laquelle votre instance de serveur GitHub d'entreprise est installée ou un VPC avec accès à votre instance de serveur GitHub d'entreprise via VPN ou Direct Connect.

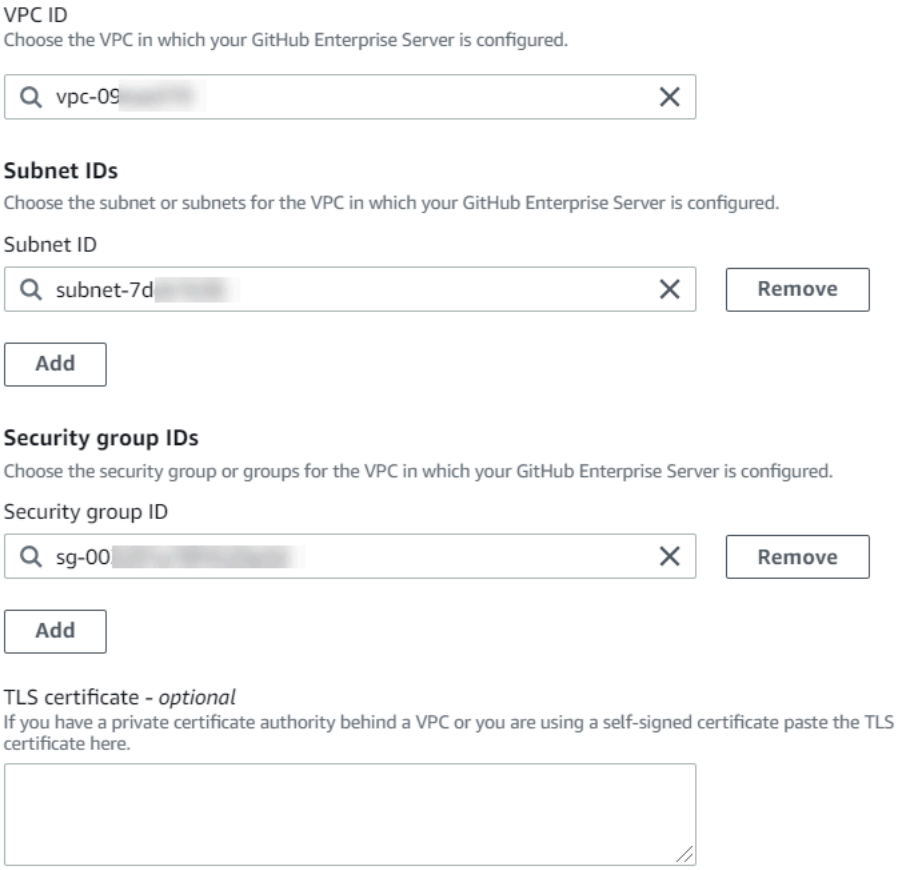
   b. Sous Subnet ID (ID de sous-réseau), choisissez Add (Ajouter). Dans le champ, choisissez l'ID de sous-réseau que vous souhaitez utiliser pour votre hôte. Vous pouvez choisir jusqu'à 10 sous-réseaux.

      Assurez-vous de choisir le sous-réseau pour l'infrastructure dans laquelle votre instance de serveur GitHub d'entreprise est installée ou un sous-réseau avec accès à votre instance de serveur GitHub d'entreprise installée via VPN ou Direct Connect.

   c. Sous Security group IDs (ID de groupe de sécurité), choisissez Add (Ajouter). Dans le champ, choisissez le groupe de sécurité que vous souhaitez utiliser pour votre hôte. Vous pouvez choisir jusqu'à 10 groupes de sécurité.

      Assurez-vous de choisir le groupe de sécurité pour l'infrastructure sur laquelle votre instance de serveur GitHub d'entreprise est installée ou un groupe de sécurité ayant accès à votre instance de serveur GitHub d'entreprise installée via un VPN ou Direct Connect.

d. Si vous avez configuré un VPC privé et que vous avez configuré votre instance de serveur GitHub d'entreprise pour effectuer une validation TLS à l'aide d'une autorité de certification non publique, dans Certificat TLS, entrez votre ID de certificat. La valeur du certificat TLS doit être la clé publique du certificat.



4. Choisissez Connect to GitHub Enterprise Server. La connexion créée s'affiche avec le statut En attente. Une ressource hôte est créée pour la connexion avec les informations de serveur que vous avez fournies. Pour le nom d'hôte, l'URL est utilisée.
5. Choisissez Update pending connection (Mettre à jour la connexion en attente).
6. Si vous y êtes invité, sur la page de connexion GitHub Enterprise, connectez-vous avec vos informations GitHub d'identification Enterprise.
7. Sur la page Créer une GitHub application, choisissez un nom pour votre application.
8. Sur la page GitHub d'autorisation, choisissez Autoriser<app-name>.
9. Sur la page d'installation de l'application, un message indique que l'application Connector est prête à être installée. Si vous avez plusieurs organisations, vous pouvez être invité à choisir l'organisation dans laquelle vous souhaitez installer l'application.

Choisissez les paramètres du référentiel dans lesquels vous souhaitez installer l'application. Choisissez Installer.

10. La page de connexion affiche la connexion créée avec un statut Disponible.

Étape 3 : enregistrer l'action source de votre serveur d' GitHub entreprise

Suivez ces étapes dans l'assistant ou sur la page Modifier l'action pour enregistrer votre action source avec vos informations de connexion.

Pour terminer et enregistrer votre action source avec votre connexion

1. Dans Nom du référentiel, choisissez le nom de votre référentiel tiers.
2. Sous Déclencheurs du pipeline, vous pouvez ajouter des déclencheurs si votre action est une AWS CodeConnections action. Pour configurer la configuration des déclencheurs du pipeline et pour éventuellement filtrer à l'aide de déclencheurs, reportez-vous à la sectionAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ».
3. Dans Output artifact format (Format d'artefact de sortie), vous devez choisir le format de vos artefacts.

   - Pour stocker les artefacts de sortie issus de l'action GitHub Enterprise Server à l'aide de la méthode par défaut, choisissez CodePipelinepar défaut. L'action accède aux fichiers depuis le référentiel GitHub Enterprise Server et stocke les artefacts dans un fichier ZIP dans le magasin d'artefacts du pipeline.
   - Pour stocker un fichier JSON contenant une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git, choisissez Full clone (Clone complet). Cette option ne peut être utilisée que par des actions CodeBuild en aval.

4. Choisissez Suivant dans l'assistant ou Enregistrer sur la page d'action Modifier.

Création d'un hôte et d'une connexion à GitHub Enterprise Server (CLI)

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour créer une connexion.

Pour ce faire, utilisez la commande create-connection.

> **Important**
>
> Une connexion créée via le AWS CLI ou AWS CloudFormation est en `PENDING` état par défaut. Après avoir créé une connexion avec la CLI CloudFormation, utilisez la console pour modifier la connexion afin de définir son état`AVAILABLE`.

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour créer un hôte pour les connexions installées.

> **Note**
>
> Vous ne créez un hôte qu'une seule fois par compte GitHub Enterprise Server. Toutes vos connexions à un compte GitHub Enterprise Server spécifique utiliseront le même hôte.

Vous utilisez un hôte pour représenter le point de terminaison de l'infrastructure sur laquelle votre fournisseur tiers est installé. Une fois que vous avez terminé la création de l'hôte avec la CLI, l'hôte est en attente. Vous configurez ou enregistrez ensuite l'hôte pour le faire passer au statut Disponible. Une fois l'hôte disponible, procédez comme suit pour créer une connexion.

Pour ce faire, utilisez la commande create-host.

> **Important**
>
> Un hôte créé via le AWS CLI est en `Pending` statut par défaut. Après avoir créé un hôte à l'aide de la CLI, utilisez la console ou l'interface de ligne de commande pour configurer l'hôte afin qu'il définisse son statut`Available`.

Pour créer un hôte

1. Ouvrez une invite de terminal (Linux, macOS ou Unix) ou de commande (Windows). Utilisez le AWS CLI pour exécuter la create-host commande, en spécifiant le `--name` `--provider-type`, et `--provider-endpoint` pour votre connexion. Dans cet exemple, le nom du fournisseur tiers est `GitHubEnterpriseServer` et le point de terminaison est `my-instance.dev`.

```
aws codestar-connections create-host --name MyHost --provider-type
 GitHubEnterpriseServer --provider-endpoint "https://my-instance.dev"
```

En cas de succès, cette commande renvoie les informations Amazon Resource Name (ARN) hôte semblables à ce qui suit.

```
{
    "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
Host-28aef605"
}
```

Après cette étape, l'hôte présente l'état `PENDING`.

2. Utilisez la console pour terminer la configuration de l'hôte et passer l'hôte vers l'état `Available`.

Pour créer une connexion à GitHub Enterprise Server

1. Ouvrez une invite de terminal (Linux, macOS ou Unix) ou de commande (Windows). Utilisez le AWS CLI pour exécuter la create-connection commande, en spécifiant le `--host-arn` et `--connection-name` pour votre connexion.

```
aws codestar-connections create-connection --host-arn arn:aws:codestar-
connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
 MyConnection
```

En cas de succès, cette commande renvoie les informations ARN de connexion semblables à ce qui suit.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad"
}
```

2. Utilisez la console pour configurer la connexion en attente.
3. Le pipeline détecte par défaut les modifications lors de l'envoi du code vers le référentiel des sources de connexion. Pour configurer la configuration du déclencheur du pipeline pour le lancement manuel ou pour les balises Git, effectuez l'une des opérations suivantes :

- Pour configurer la configuration du déclencheur du pipeline de manière à ce qu'elle commence par un déverrouillage manuel uniquement, ajoutez la ligne suivante à la configuration :

```
"DetectChanges": "false",
```

- Pour configurer la configuration des déclencheurs du pipeline afin de filtrer avec des déclencheurs, voir plus de détails dansAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ». Par exemple, ce qui suit ajoute au niveau du pipeline de la définition JSON du pipeline. Dans cet exemple, `release-v0` et `release-v1` sont les balises Git à inclure et `release-v2` les balises Git à exclure.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "tags": {
                                "includes": [
                                    "release-v0", "release-v1"
                                ],
                                "excludes": [
                                    "release-v2"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
```

GitLab.com connexions

Les connexions vous permettent d'autoriser et d'établir des configurations qui associent votre fournisseur tiers à vos AWS ressources. Pour associer votre référentiel tiers en tant que source de votre pipeline, vous devez utiliser une connexion.

> **Note**
>
> Au lieu de créer ou d'utiliser une connexion existante dans votre compte, vous pouvez utiliser une connexion partagée entre un autre Compte AWS. Consultez Utiliser une connexion partagée avec un autre compte.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans les régions Asie-Pacifique (Hong Kong), Asie-Pacifique (Hyderabad), Asie-Pacifique (Jakarta), Asie-Pacifique (Melbourne), Asie-Pacifique (Osaka), Afrique (Le Cap), Moyen-Orient (Bahreïn), Moyen-Orient (Émirats arabes unis), Europe (Espagne), Europe (Zurich), Israël (Tel Aviv) ou (). AWS GovCloud US-West Pour faire référence aux autres actions disponibles, voirIntégrations de produits et de services avec CodePipeline. Pour les considérations relatives à cette action dans la région Europe (Milan), voir la note dansCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

Pour ajouter une action GitLab.com source CodePipeline, vous pouvez choisir de :

- Utilisez l'assistant de création de pipeline ou la page d'action Modifier de la CodePipeline console pour choisir l'option du GitLabfournisseur. Création d'une connexion à GitLab.com (console)Reportez-vous à la section pour ajouter l'action. La console vous permet de créer une ressource de connexions.
- Utilisez la CLI pour ajouter la configuration de l'`CreateSourceConnection`action avec le `GitLab` fournisseur comme suit :
  - Pour créer vos ressources de connexions, reportez-vous Création d'une connexion à GitLab.com (CLI) à la section Création d'une ressource de connexions avec la CLI.
  - Utilisez l'`CreateSourceConnection`exemple de configuration d'action CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées pour ajouter votre action comme indiqué dansCréation d'un pipeline (interface de ligne de commande).

> **Note**
>
> Vous pouvez également créer une connexion à l'aide de la console Developer Tools sous Paramètres. Voir Créer une connexion.

> **Note**
>
> En autorisant l'installation de cette connexion GitLab.com, vous autorisez notre service à traiter vos données en accédant à votre compte, et vous pouvez révoquer ces autorisations à tout moment en désinstallant l'application.

Avant de commencer :

- Vous devez déjà avoir créé un compte auprès de GitLab.com.

  > **Note**
  >
  > Les connexions fournissent uniquement l'accès aux référentiels appartenant au compte qui a été utilisé pour créer et autoriser la connexion.

  > **Note**
  >
  > Vous pouvez créer des connexions à un référentiel dans lequel vous avez le rôle de propriétaire GitLab, puis la connexion peut être utilisée avec le référentiel avec des ressources telles que CodePipeline. Pour les référentiels dans des groupes, il n'est pas nécessaire d'être le propriétaire du groupe.

- Pour spécifier une source pour votre pipeline, vous devez déjà avoir créé un dépôt sur gitlab.com.

Rubriques

- Création d'une connexion à GitLab.com (console)
- Création d'une connexion à GitLab.com (CLI)

Création d'une connexion à GitLab.com (console)

Suivez ces étapes pour utiliser la CodePipeline console afin d'ajouter une action de connexion pour votre projet (référentiel) dans GitLab.

Pour créer ou modifier votre pipeline

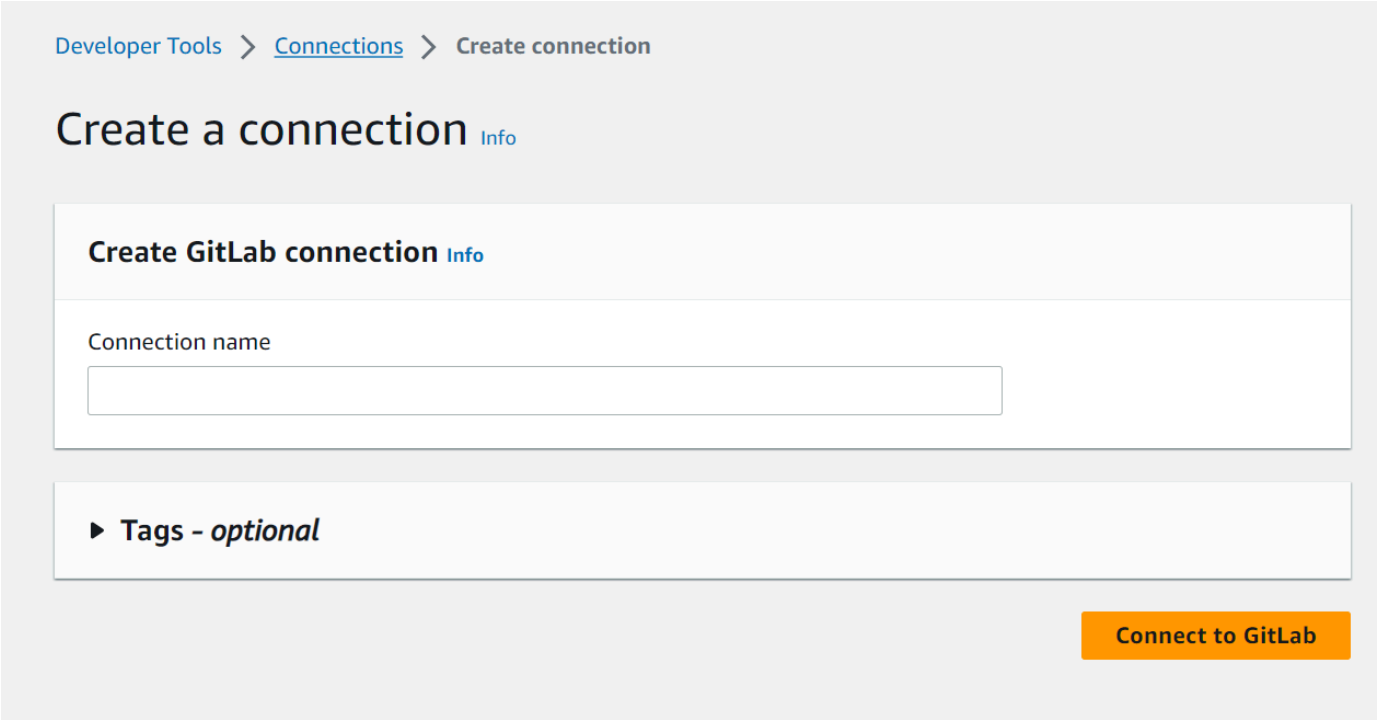
1. Connectez-vous à la CodePipeline console.
2. Choisissez l'une des options suivantes.

   - Choisissez de créer un pipeline. Suivez les étapes décrites dans Créer un pipeline pour terminer le premier écran et choisissez Next. Sur la page Source, sous Source Provider, sélectionnez GitLab.
   - Choisissez de modifier un pipeline existant. Choisissez Modifier, puis sélectionnez Modifier l'étape. Choisissez d'ajouter ou de modifier votre action source. Sur la page Modifier l'action, sous Nom de l'action, entrez le nom de votre action. Dans Action provider, sélectionnez GitLab.
3. Effectuez l'une des actions suivantes :

   - Sous Connexion, si vous n'avez pas encore créé de connexion avec votre fournisseur, choisissez Se connecter à GitLab. Passez à l'étape 4 pour créer la connexion.
   - Sous Connexion, si vous avez déjà créé une connexion avec votre fournisseur, choisissez-la. Passez à l'étape 9.

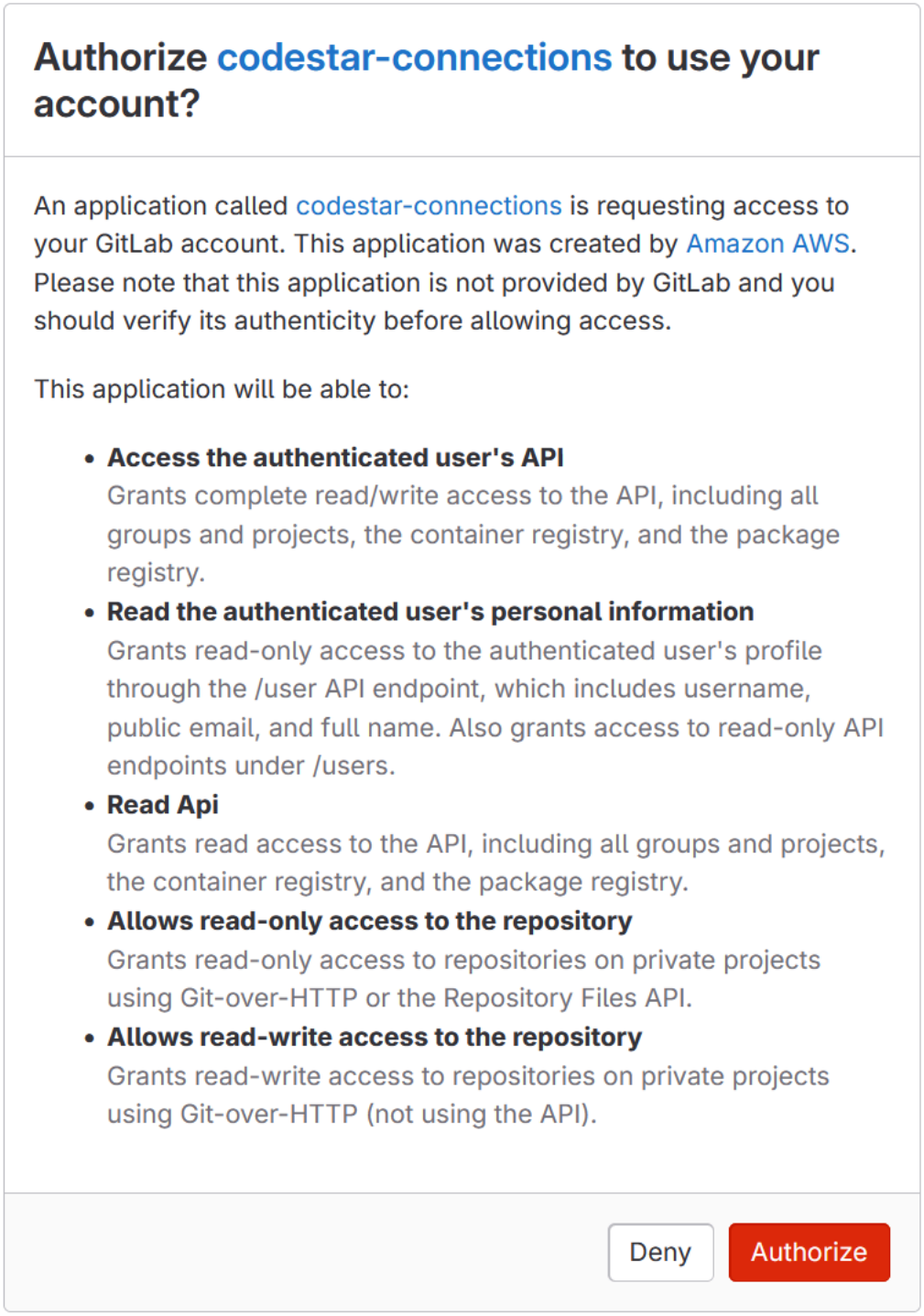
   > **Note**
   >
   > Si vous fermez la fenêtre contextuelle avant de créer une GitLab.com connexion, vous devez actualiser la page.

4. Pour créer une connexion à un GitLab.com référentiel, sous Sélectionnez un fournisseur, sélectionnez GitLab. Dans Connection name (Nom de la connexion), saisissez le nom de la connexion que vous souhaitez créer. Choisissez Connect to GitLab.



5. Lorsque la page de connexion GitLab.com s'affiche, connectez-vous avec vos informations d'identification, puis choisissez Se connecter.
6. Si c'est la première fois que vous autorisez la connexion, une page d'autorisation s'affiche avec un message demandant l'autorisation de connexion pour accéder à votre GitLab.com compte.

   Choisissez Authorize (Autoriser).



7. Le navigateur revient à la page de la console des connexions. Sous Créer une GitLab connexion, la nouvelle connexion est affichée dans Nom de la connexion.
8. Choisissez Connect to GitLab.

   Vous serez renvoyé à la CodePipeline console.

> **Note**
>
> Une fois la GitLab.com connexion créée avec succès, une bannière de réussite s'affiche dans la fenêtre principale.
> Si vous ne vous êtes pas déjà connecté GitLab à l'ordinateur actuel, vous devez fermer manuellement la fenêtre contextuelle.

9. Dans Nom du référentiel, choisissez le nom de votre projet en GitLab spécifiant le chemin du projet avec l'espace de noms. Par exemple, pour un référentiel au niveau du groupe, entrez le nom du référentiel au format suivant : `group-name/repository-name` Pour plus d'informations sur le chemin et l'espace de noms, consultez le `path_with_namespace` champ dans https://docs.gitlab.com/ee/api/projects.html#get -single-project. Pour plus d'informations sur l'espace de noms dans GitLab, consultez https://docs.gitlab.com/ee/user/namespace/.

> **Note**
>
> Pour les groupes dans GitLab, vous devez spécifier manuellement le chemin du projet avec l'espace de noms. Par exemple, pour un référentiel nommé `myrepo` dans un groupe`mygroup`, entrez ce qui suit : `mygroup/myrepo` Vous pouvez trouver le chemin du projet avec l'espace de noms dans l'URL dans GitLab.

10. Sous Déclencheurs du pipeline, vous pouvez ajouter des déclencheurs si votre action est une AWS CodeConnections action. Pour configurer la configuration des déclencheurs du pipeline et pour éventuellement filtrer à l'aide de déclencheurs, reportez-vous à la sectionAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ».
11. Dans Nom de branche, choisissez la branche où vous souhaitez que votre pipeline détecte les modifications de source.

> **Note**
>
> Si le nom de la branche n'est pas renseigné automatiquement, cela signifie que vous n'avez pas accès au référentiel en tant que propriétaire. Soit le nom du projet n'est pas valide, soit la connexion utilisée n'a pas accès au project/repository.

12. Dans Output artifact format (Format d'artefact de sortie), vous devez choisir le format de vos artefacts.

- Pour stocker les artefacts de sortie de l' GitLab.com action à l'aide de la méthode par défaut, choisissez CodePipeline par défaut. L'action accède aux fichiers depuis le GitLab.com référentiel et stocke les artefacts dans un fichier ZIP dans le magasin d'artefacts du pipeline.
- Pour stocker un fichier JSON contenant une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git, choisissez Full clone (Clone complet). Cette option ne peut être utilisée que par les actions CodeBuild en aval.

  Si vous choisissez cette option, vous devrez mettre à jour les autorisations associées à votre rôle de service de CodeBuild projet, comme indiqué dansAjoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com. Pour consulter un didacticiel expliquant comment utiliser l'option de clonage complet, voirTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

13. Choisissez d'enregistrer l'action source et de continuer.

Création d'une connexion à GitLab.com (CLI)

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour créer une connexion.

Pour ce faire, utilisez la commande create-connection.

> **Important**
>
> Une connexion créée via le AWS CLI ou AWS CloudFormation est en `PENDING` statut par défaut. Après avoir créé une connexion avec la CLI CloudFormation, utilisez la console pour modifier la connexion afin de définir son état`AVAILABLE`.

Pour créer une connexion

1. Ouvrez une invite de terminal (Linux, macOS ou Unix) ou de commande (Windows). Utilisez le AWS CLI pour exécuter la create-connection commande, en spécifiant le `--provider-type` et `--connection-name` pour votre connexion. Dans cet exemple, le nom du fournisseur tiers est `GitLab` et le nom de connexion spécifié est `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type GitLab --connection-name
    MyConnection
   ```

En cas de succès, cette commande renvoie les informations ARN de connexion semblables à ce qui suit.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Utilisez la console pour terminer la connexion. Pour plus d'informations, voir Mettre à jour une connexion en attente.
3. Le pipeline détecte par défaut les modifications lors de l'envoi du code vers le référentiel des sources de connexion. Pour configurer la configuration du déclencheur du pipeline pour le lancement manuel ou pour les balises Git, effectuez l'une des opérations suivantes :

   - Pour configurer la configuration du déclencheur du pipeline de manière à ce qu'elle commence par un déverrouillage manuel uniquement, ajoutez la ligne suivante à la configuration :

     ```
     "DetectChanges": "false",
     ```

   - Pour configurer la configuration des déclencheurs du pipeline afin de filtrer avec des déclencheurs, voir plus de détails dansAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ». Par exemple, ce qui suit ajoute au niveau du pipeline de la définition JSON du pipeline. Dans cet exemple, `release-v0` et `release-v1` sont les balises Git à inclure et `release-v2` les balises Git à exclure.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
     ```

```
                    }
                ]
            }
        }
    ]
```

Connexions pour l' GitLab autogestion

Les connexions vous permettent d'autoriser et d'établir des configurations qui associent votre fournisseur tiers à vos AWS ressources. Pour associer votre référentiel tiers en tant que source de votre pipeline, vous devez utiliser une connexion.

> **Note**
>
> Au lieu de créer ou d'utiliser une connexion existante dans votre compte, vous pouvez utiliser une connexion partagée entre un autre Compte AWS. Consultez Utiliser une connexion partagée avec un autre compte.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans les régions Asie-Pacifique (Hong Kong), Asie-Pacifique (Hyderabad), Asie-Pacifique (Jakarta), Asie-Pacifique (Melbourne), Asie-Pacifique (Osaka), Afrique (Le Cap), Moyen-Orient (Bahreïn), Moyen-Orient (Émirats arabes unis), Europe (Espagne), Europe (Zurich), Israël (Tel Aviv) ou (). AWS GovCloud US-West Pour faire référence aux autres actions disponibles, voirIntégrations de produits et de services avec CodePipeline. Pour les considérations relatives à cette action dans la région Europe (Milan), voir la note dansCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

Pour ajouter une action source GitLab autogérée CodePipeline, vous pouvez choisir de :

- Utilisez l'assistant de création de pipeline ou la page d'action Modifier de la CodePipeline console pour choisir l'option fournisseur GitLabautogéré. Création d'une connexion à l' GitLab autogestion (console)Reportez-vous à la section pour ajouter l'action. La console vous permet de créer une ressource hôte et une ressource de connexions.

- Utilisez la CLI pour ajouter la configuration de l'`CreateSourceConnection`action avec le `GitLabSelfManaged` fournisseur et créer vos ressources :
  - Pour créer vos ressources de connexion, reportez-vous Création d'un hôte et d'une connexion à une interface GitLab autogérée (CLI) à la section Création d'une ressource hôte et d'une ressource de connexions avec la CLI.
  - Utilisez l'`CreateSourceConnection`exemple de configuration d'action CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées pour ajouter votre action comme indiqué dansCréation d'un pipeline (interface de ligne de commande).

> **Note**
>
> Vous pouvez également créer une connexion à l'aide de la console Developer Tools sous Paramètres. Voir Créer une connexion.

Avant de commencer :

- Vous devez déjà avoir créé un compte GitLab et disposer de l'édition GitLab Enterprise ou de l'édition GitLab Community avec une installation autogérée. Pour plus d'informations, consultez https://docs.gitlab.com/ee/subscriptions/self_managed/.

  > **Note**
  >
  > Les connexions fournissent uniquement l'accès au compte qui a été utilisé pour créer et autoriser la connexion.

  > **Note**
  >
  > Vous pouvez créer des connexions à un référentiel dans lequel vous avez le rôle de propriétaire GitLab, puis la connexion peut être utilisée avec des ressources telles que CodePipeline. Pour les référentiels dans des groupes, il n'est pas nécessaire d'être le propriétaire du groupe.

- Vous devez déjà avoir créé un jeton d'accès GitLab personnel (PAT) avec le `api` scope. Si le mode administrateur est activé sur votre GitLab instance, le PAT doit également inclure le

`admin_mode` champ d'application. Pour plus d'informations, consultez les sections Jetons d'accès personnels et Étendue des jetons d'accès sur le GitLab site Web. Vous devez être administrateur pour créer et utiliser le PAT.

> **Note**
>
> Votre PAT est utilisé pour autoriser l'hôte et n'est pas stocké ni utilisé par les connexions à d'autres fins. Pour configurer un hôte, vous pouvez créer un PAT temporaire, puis, une fois l'hôte configuré, vous pouvez le supprimer.

- Vous pouvez choisir de configurer votre hôte à l'avance. Vous pouvez configurer un hôte avec ou sans VPC. Pour plus de détails sur la configuration du VPC et des informations supplémentaires sur la création d'un hôte, consultez la section Créer un hôte.

Rubriques

- Création d'une connexion à l' GitLab autogestion (console)
- Création d'un hôte et d'une connexion à une interface GitLab autogérée (CLI)

Création d'une connexion à l' GitLab autogestion (console)

Suivez ces étapes pour utiliser la CodePipeline console afin d'ajouter une action de connexion pour votre référentiel GitLab autogéré.

> **Note**
>
> GitLab les connexions autogérées fournissent uniquement l'accès aux référentiels détenus par le compte GitLab autogéré qui a été utilisé pour créer la connexion.

Avant de commencer :

Pour qu'une connexion hôte soit GitLab autogérée, vous devez avoir suivi les étapes de création d'une ressource hôte pour votre connexion. Consultez la section Gérer les hôtes pour les connexions.

Étape 1 : créer ou modifier votre pipeline

Pour créer ou modifier votre pipeline

1. Connectez-vous à la CodePipeline console.
2. Choisissez l'une des options suivantes.
   - Choisissez de créer un pipeline. Suivez les étapes décrites dans Créer un pipeline pour terminer le premier écran et choisissez Next. Sur la page Source, sous Fournisseur de source, sélectionnez GitLab Autogéré.
   - Choisissez de modifier un pipeline existant. Choisissez Modifier, puis sélectionnez Modifier l'étape. Choisissez d'ajouter ou de modifier votre action source. Sur la page Modifier l'action, sous Nom de l'action, entrez le nom de votre action. Dans Action provider, sélectionnez GitLab Autogéré.
3. Effectuez l'une des actions suivantes :
   - Sous Connexion, si vous n'avez pas encore créé de connexion avec votre fournisseur, choisissez Connect to GitLab self-managed. Passez à l'étape 2 : créer une connexion à l' GitLab autogestion.
   - Sous Connexion, si vous avez déjà créé une connexion avec votre fournisseur, choisissez la connexion, puis passez à l'étape 3 : Enregistrer votre action source GitLab autogérée.

Étape 2 : Création d'une connexion à l' GitLab autogestion

Une fois que vous avez choisi de créer la connexion, la page Connect to GitLab self-managed s'affiche.

Pour vous connecter à une solution GitLab autogérée

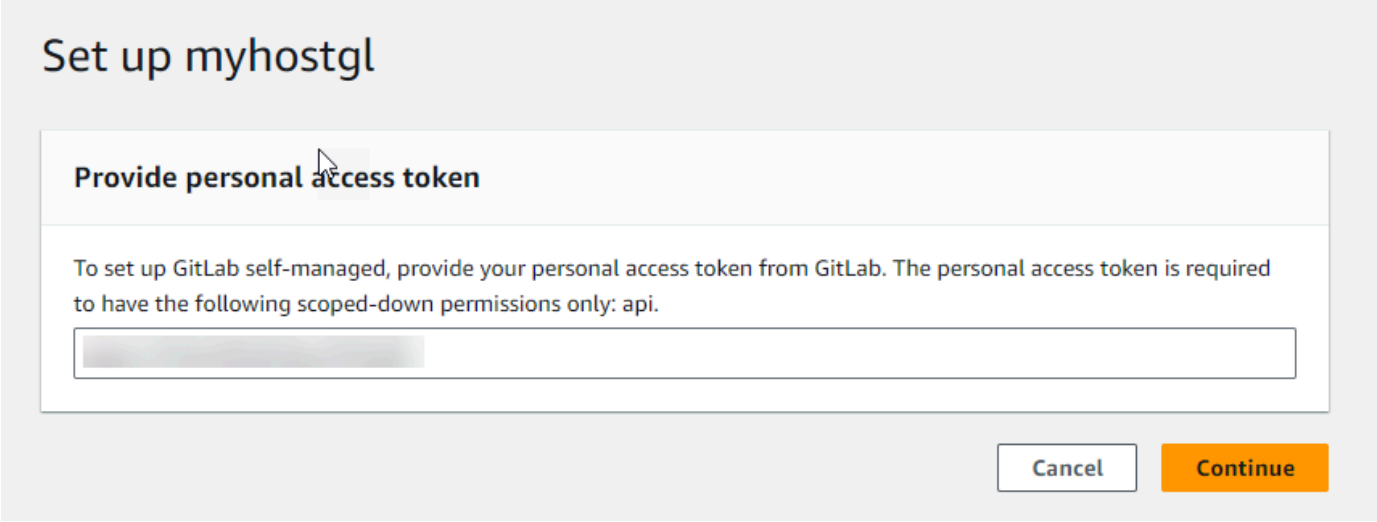
1. Dans Connection name (Nom de la connexion), saisissez le nom de votre connexion.
2. Dans URL, saisissez le point de terminaison de votre serveur.

> **Note**
>
> Si l'URL fournie a déjà été utilisée pour configurer un hôte pour une connexion, vous serez invité à choisir l'ARN de la ressource hôte créé précédemment pour ce point de terminaison.

3. Si vous avez lancé votre serveur dans un Amazon VPC et que vous souhaitez vous connecter à votre VPC, choisissez Utiliser un VPC et complétez les informations relatives au VPC.
4. Choisissez Connect to GitLab self-managed. La connexion créée s'affiche avec le statut En attente. Une ressource hôte est créée pour la connexion avec les informations de serveur que vous avez fournies. Pour le nom d'hôte, l'URL est utilisée.
5. Choisissez Update pending connection (Mettre à jour la connexion en attente).
6. Si une page s'ouvre avec un message de redirection confirmant que vous souhaitez continuer vers le fournisseur, choisissez Continuer. Entrez l'autorisation du fournisseur.
7. Une *`host_name`* page de configuration s'affiche. Dans Fournir un jeton d'accès personnel, indiquez le `api` champ d'application à votre GitLab PAT. Si le mode administrateur est activé sur votre GitLab instance, le PAT doit également inclure le `admin_mode` champ d'application. Pour plus d'informations, consultez la section Étendue des jetons d'accès sur le GitLab site Web.

   > **Note**
   >
   > Seul un administrateur peut créer et utiliser le PAT.

   Sélectionnez Continuer.



8. La page de connexion affiche la connexion créée avec un statut Disponible.

Étape 3 : Enregistrez votre action source GitLab autogérée

Suivez ces étapes dans l'assistant ou sur la page Modifier l'action pour enregistrer votre action source avec vos informations de connexion.

Pour terminer et enregistrer votre action source avec votre connexion

1. Dans Nom du référentiel, choisissez le nom de votre référentiel tiers.
2. Sous Déclencheurs du pipeline, vous pouvez ajouter des déclencheurs si votre action est une AWS CodeConnections action. Pour configurer la configuration des déclencheurs du pipeline et pour éventuellement filtrer à l'aide de déclencheurs, reportez-vous à la sectionAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ».
3. Dans Output artifact format (Format d'artefact de sortie), vous devez choisir le format de vos artefacts.
   - Pour stocker les artefacts de sortie de l'action GitLab autogérée à l'aide de la méthode par défaut, choisissez CodePipelinepar défaut. L'action accède aux fichiers depuis le référentiel et stocke les artefacts dans un fichier ZIP dans le magasin d'artefacts du pipeline.
   - Pour stocker un fichier JSON contenant une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git, choisissez Full clone (Clone complet). Cette option ne peut être utilisée que par des actions CodeBuild en aval.
4. Choisissez Suivant dans l'assistant ou Enregistrer sur la page d'action Modifier.

Création d'un hôte et d'une connexion à une interface GitLab autogérée (CLI)

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour créer une connexion.

Pour ce faire, utilisez la commande create-connection.

> **Important**
>
> Une connexion créée via le AWS CLI ou AWS CloudFormation est en `PENDING` état par défaut. Après avoir créé une connexion à l'aide de la CLI CloudFormation, utilisez la console pour modifier la connexion afin de définir son état`AVAILABLE`.

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour créer un hôte pour les connexions installées.

Vous utilisez un hôte pour représenter le point de terminaison de l'infrastructure sur laquelle votre fournisseur tiers est installé. Une fois que vous avez terminé la création de l'hôte avec la CLI, l'hôte est en attente. Vous configurez ou enregistrez ensuite l'hôte pour le faire passer au statut Disponible. Une fois l'hôte disponible, procédez comme suit pour créer une connexion.

Pour ce faire, utilisez la commande create-host.

> **Important**
>
> Un hôte créé via le AWS CLI est en `Pending` statut par défaut. Après avoir créé un hôte à l'aide de la CLI, utilisez la console ou l'interface de ligne de commande pour configurer l'hôte afin qu'il définisse son statut`Available`.

Pour créer un hôte

1. Ouvrez une invite de terminal (Linux, macOS ou Unix) ou de commande (Windows). Utilisez le AWS CLI pour exécuter la create-host commande, en spécifiant le `--name``--provider-type`, et `--provider-endpoint` pour votre connexion. Dans cet exemple, le nom du fournisseur tiers est `GitLabSelfManaged` et le point de terminaison est `my-instance.dev`.

   ```
   aws codestar-connections create-host --name MyHost --provider-type
    GitLabSelfManaged --provider-endpoint "https://my-instance.dev"
   ```

   En cas de succès, cette commande renvoie les informations Amazon Resource Name (ARN) hôte semblables à ce qui suit.

   ```
   {
       "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
   Host-28aef605"
   }
   ```

   Après cette étape, l'hôte présente l'état `PENDING`.
2. Utilisez la console pour terminer la configuration de l'hôte et passer l'hôte vers l'état `Available`.

Pour créer une connexion à l' GitLab autogéré

1. Ouvrez une invite de terminal (Linux, macOS ou Unix) ou de commande (Windows). Utilisez le AWS CLI pour exécuter la create-connection commande, en spécifiant le `--host-arn` et `--connection-name` pour votre connexion.

   ```
   aws codestar-connections create-connection --host-arn arn:aws:codestar-
   connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
    MyConnection
   ```

   En cas de succès, cette commande renvoie les informations ARN de connexion semblables à ce qui suit.

   ```
   {
       "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
   aEXAMPLE-8aad"
   }
   ```

2. Utilisez la console pour configurer la connexion en attente.
3. Le pipeline détecte par défaut les modifications lors de l'envoi du code vers le référentiel des sources de connexion. Pour configurer la configuration du déclencheur du pipeline pour le lancement manuel ou pour les balises Git, effectuez l'une des opérations suivantes :

   - Pour configurer la configuration du déclencheur du pipeline de manière à ce qu'elle commence par un déverrouillage manuel uniquement, ajoutez la ligne suivante à la configuration :

     ```
     "DetectChanges": "false",
     ```

   - Pour configurer la configuration des déclencheurs du pipeline afin de filtrer avec des déclencheurs, voir plus de détails dansAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ». Par exemple, ce qui suit ajoute au niveau du pipeline de la définition JSON du pipeline. Dans cet exemple, `release-v0` et `release-v1` sont les balises Git à inclure et `release-v2` les balises Git à exclure.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
     ```

```
                    {
                        "tags": {
                            "includes": [
                                "release-v0", "release-v1"
                            ],
                            "excludes": [
                                "release-v2"
                            ]
                        }
                    }
                ]
            }
        }
    ]
```

Utiliser une connexion partagée avec un autre compte

Vous pouvez créer et gérer une connexion partagée à l'aide de AWS RAM. Cela permet de partager des connexions Comptes AWS pour accéder à des référentiels tiers. Cela permet d'utiliser une connexion unique dans les CodePipeline pipelines entre les comptes tout en réduisant la nécessité pour les utilisateurs de gérer et d'administrer des connexions distinctes dans chaque compte.

Pour utiliser les connexions partagées dans CodePipeline, procédez comme suit.

- Créez une connexion à l'aide de la console Developer Tools sous Paramètres. Voir Créer une connexion.
- Configurez le partage des ressources à l'aide de AWS RAM. Voir Partager des connexions avec Comptes AWS.
- Lorsque vous utilisez l'assistant de création de pipeline ou la page d'action Modifier de la CodePipeline console pour choisir le fournisseur de connexion, par exemple l'option fournisseur Bitbucket, vous pouvez choisir la connexion qui a été partagée avec le compte cible.

Automatisez le démarrage des pipelines en utilisant des déclencheurs et des filtres

Les déclencheurs vous permettent de configurer votre pipeline pour qu'il démarre sur un type d'événement particulier ou sur un type d'événement filtré, par exemple lorsqu'une modification est détectée sur une branche ou une pull request en particulier. Les déclencheurs sont configurables pour les actions source avec des connexions qui utilisent l'`CodeStarSourceConnection`action dans CodePipeline GitHub, telles que Bitbucket et GitLab. Pour plus d'informations sur les actions source qui utilisent des connexions, consultezAjoutez des fournisseurs de sources tiers aux pipelines à l'aide de AWS CodeConnections.

Les actions source, telles que CodeCommit S3, utilisent la détection automatique des modifications pour démarrer les pipelines lorsqu'une modification est apportée. Pour de plus amples informations, veuillez consulter CodeCommit actions à la source et EventBridge.

Vous spécifiez les déclencheurs à l'aide de la console ou de la CLI.

Vous spécifiez les types de filtres comme suit :

- Pas de filtre

  Cette configuration de déclencheur démarre votre pipeline à chaque poussée vers la branche par défaut spécifiée dans le cadre de la configuration de l'action.
- Spécifier le filtre

  Vous ajoutez un filtre qui démarre votre pipeline sur un filtre spécifique, par exemple sur les noms de branches pour un envoi de code, et récupère le commit exact. Cela permet également de configurer le pipeline pour qu'il ne démarre pas automatiquement lors d'une modification.
  - Push
    - Les combinaisons de filtres valides sont les suivantes :
      - Balises Git

        Inclure ou exclure
      - branches

        Inclure ou exclure
      - branches + chemins de fichiers

Inclure ou exclure

- Pull request
  - Les combinaisons de filtres valides sont les suivantes :
    - branches

      Inclure ou exclure
    - branches + chemins de fichiers

      Inclure ou exclure
- Ne pas détecter les modifications

  Cela n'ajoute aucun déclencheur et le pipeline ne démarre pas automatiquement lors d'une modification.

Le tableau suivant fournit des options de filtre valides pour chaque type d'événement. Le tableau indique également quelles configurations de déclencheur sont définies par défaut sur true ou false pour la détection automatique des modifications dans la configuration de l'action.

Configuration du déclencheur	Type d'événement	Options de filtre	Détecter les modifications
Ajouter un déclencheur — aucun filtre	Aucune	none	true
Ajouter un déclencheur — filtre en cas de poussée de code	événement push	Balises Git, branches, chemins de fichiers	false
Ajouter un déclencheur — filtre pour les pull requests	pull requests	branches, chemins de fichiers	false
Pas de déclencheur — ne pas détecter	Aucune	none	false

> **Note**
>
> Ce type de déclencheur utilise la détection automatique des modifications (en tant que type de `Webhook` déclencheur). Les fournisseurs d'actions source qui utilisent ce type de déclencheur sont des connexions configurées pour le transfert de code (Bitbucket Cloud GitHub, GitHub Enterprise Server, GitLab .com et GitLab autogérées).

Pour les définitions des champs et des références supplémentaires sur les déclencheurs, voir

Pour obtenir la liste des définitions de champs dans la structure JSON, consultez`déclencheurs`.

Pour le filtrage, les modèles d'expressions régulières au format global sont pris en charge, comme indiqué dansUtilisation de modèles globulaires dans la syntaxe.

> **Note**
>
> Dans certains cas, pour les pipelines dont les déclencheurs sont filtrés sur les chemins de fichiers, le pipeline peut ne pas démarrer lorsqu'une branche avec un filtre de chemin de fichier est créée pour la première fois. Pour de plus amples informations, veuillez consulter Les pipelines avec des connexions qui utilisent le filtrage des déclencheurs par chemin de fichier peuvent ne pas démarrer lors de la création de la branche.

Considérations relatives aux filtres déclencheurs

Les considérations suivantes s'appliquent lors de l'utilisation de déclencheurs.

- Vous ne pouvez pas ajouter plus d'un déclencheur par action source.
- Vous pouvez ajouter plusieurs types de filtres à un déclencheur. Pour obtenir un exemple, consultez 4 : Un déclencheur avec deux types de filtres push avec des inclusions et des exclusions contradictoires.
- Dans le cas d'un déclencheur avec filtres de chemins de branches et de fichiers, lorsque vous appuyez sur la branche pour la première fois, le pipeline ne s'exécute pas car il n'est pas possible d'accéder à la liste des fichiers modifiés pour la branche nouvellement créée.
- La fusion d'une pull request peut déclencher deux exécutions de pipeline dans les cas où les configurations de déclenchement push (filtre de branches) et pull request (filtre de branches) se croisent.

- Pour un filtre qui déclenche votre pipeline lors d'événements de pull request, pour le type d'événement de pull request fermé, le fournisseur de référentiel tiers pour votre connexion peut avoir un statut distinct pour un événement de fusion. Par exemple, dans Bitbucket, l'événement Git associé à une fusion n'est pas un événement de fermeture d'une pull request. Cependant GitHub, la fusion d'une pull request est un événement de fermeture. Pour de plus amples informations, veuillez consulter Événements de pull request pour les déclencheurs par fournisseur.
- Lorsque plusieurs actions de source dans un pipeline font référence à différentes branches du même référentiel via une connexion, une seule branche déclenche le pipeline de manière fiable. L'abonnement au webhook de la connexion est enregistré pour la combinaison du pipeline et du référentiel, et non pour chaque branche. Pour contourner le problème, utilisez un pipeline distinct pour chaque branche.

Événements de pull request pour les déclencheurs par fournisseur

Le tableau suivant fournit un résumé des événements Git, tels que la fermeture d'une pull request, qui se traduisent par des types d'événements de pull request par fournisseur.

	Fournisseur de référentiel pour votre connexion			
Événement de relations publiques pour le déclencheur	Bitbucket	GitHub	GHES	GitLab
Ouvrir : cette option déclenche le pipeline lorsqu'une pull request est créée pour le chemin de la branche/du fichier.	La création d'une pull request entraîne un événement OpenGit.	La création d'une pull request entraîne un événement OpenGit.	La création d'une pull request entraîne un événement OpenGit.	La création d'une pull request entraîne un événement OpenGit.
Mise à jour - Cette option déclenche	La publication d'une mise à jour entraîne	La publication d'une mise à jour entraîne	La publication d'une mise à jour entraîne	La publication d'une mise à jour entraîne

	Fournisseur de référentiel pour votre connexion			
Événement de relations publiques pour le déclencheur	Bitbucket	GitHub	GHES	GitLab
le pipeline lorsqu'une révision de pull request est publiée pour le chemin de branche/de fichier.	un événement Updated Git.	un événement Updated Git.	un événement Updated Git.	un événement Updated Git.
Fermé : cette option déclenche le pipeline lorsqu'une pull request est fermée pour le branch/file chemin.	La fusion d'une pull request dans Bitbucket entraîne un événement Closed Git. Important : la fermeture manuelle d'une pull request dans Bitbucket sans fusion n'entraîne pas d'événement Closed Git.	La fusion ou la fermeture manuelle d'une pull request entraîne un événement Closed Git.	La fusion ou la fermeture manuelle d'une pull request entraîne un événement Closed Git.	La fusion ou la fermeture manuelle d'une pull request entraîne un événement Closed Git.

Exemples de filtres déclencheurs

Dans le cas d'une configuration Git avec des filtres pour les types d'événements push et pull request, les filtres spécifiés peuvent entrer en conflit les uns avec les autres. Vous trouverez ci-dessous des exemples de combinaisons de filtres valides pour les événements de requêtes push et pull. Un déclencheur peut contenir plusieurs types de filtres, tels que deux types de filtres push dans la

configuration du déclencheur, et les types de filtres push et pull request utiliseront une opération OR entre eux, ce qui signifie que toute correspondance démarrera le pipeline. De même, chaque type de filtre peut inclure plusieurs filtres tels que FilePaths et branches ; ces filtres utiliseront une opération AND, ce qui signifie que seule une correspondance complète démarrera le pipeline. Chaque type de filtre peut contenir des inclusions et des exclusions, les exclusions ayant priorité sur les inclusions. Si le chemin d'une branche ou d'un fichier correspond à un modèle d'exclusion, il ne déclenchera pas le pipeline, même s'il correspond également à un modèle d'inclusion. Lorsqu'un commit modifie plusieurs fichiers, chaque fichier est évalué indépendamment par rapport au filtre ; si un fichier modifié est accepté (correspond à une inclusion et ne correspond pas à une exclusion), le pipeline démarre. Les noms inclus dans l'inclusion/exclusion, tels que les noms de branches, utilisent une opération OR. La liste suivante résume les opérations pour chaque partie de l'objet de configuration Git.

Pour une liste des définitions de champs dans la structure JSON et une référence détaillée pour les inclusions et les exclusions, voir`déclencheurs`.

Example 1 : Un type de filtre avec des filtres pour les branches et les chemins de fichiers (opération AND)

Pour un seul type de filtre tel que la pull request, vous pouvez combiner des filtres, et ces filtres utiliseront une opération AND, ce qui signifie que seule une correspondance complète démarrera le pipeline. L'exemple suivant montre une configuration Git pour un type d'événement push avec deux filtres différents (`filePathsetbranches`). Dans l'exemple suivant, `filePaths` sera and'ed avec : `branches`

```
{
  "filePaths": {
    "includes": ["common/**/*.js"]
  },
  "branches": {
    "includes": ["feature/**"]
  }
}
```

Avec la configuration Git ci-dessus, cet exemple montre un événement qui lancera l'exécution du pipeline en cas de réussite de l'opération AND. En d'autres termes, le chemin du fichier `common/app.js` est inclus pour le filtre, qui démarre le pipeline en tant que ET même si la branche `refs/heads/feature/triggers` spécifiée n'a pas eu d'impact.

```
{
```

```
  "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "common/app.js"
      ]
      ...
    }
  ]
}
```

L'exemple suivant montre un événement pour un déclencheur avec la configuration ci-dessus qui ne démarrera pas l'exécution du pipeline car la branche est capable de filtrer, mais pas le chemin du fichier.

```
{
   "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "src/Main.java"
      ]
      ...
    }
  ]
}
```

Example 2 : Les exclusions ont priorité sur les inclusions

Au sein d'un même filtre, les exclusions ont priorité sur les inclusions. L'exemple suivant montre une configuration Git avec un seul filtre (`branches`) dans l'objet de configuration. Cela signifie que si une branche correspond à un modèle d'exclusion (`feature-branch`dans l'exemple), le pipeline ne sera pas déclenché, même s'il correspond également à un modèle d'inclusion. Si le modèle d'inclusion correspond et qu'aucun modèle d'exclusion ne correspond, par exemple pour la `main` branche, le pipeline sera déclenché.

Pour l'exemple JSON suivant :

- Le fait d'envoyer un commit à la `main` branche déclenchera le pipeline
- Le fait d'envoyer un commit à la `feature-branch` branche ne déclenchera pas le pipeline.

```
{
  "branches": {
      "Includes": [
          "main"
      ],
      "Excludes": [
          "feature-branch"
      ]
   }
```

Example 3 : Un déclencheur avec des types de filtres de demande push et pull (opération OR), des filtres pour les chemins de fichiers et les branches (opération AND) et includes/excludes (les exclusions ont priorité)

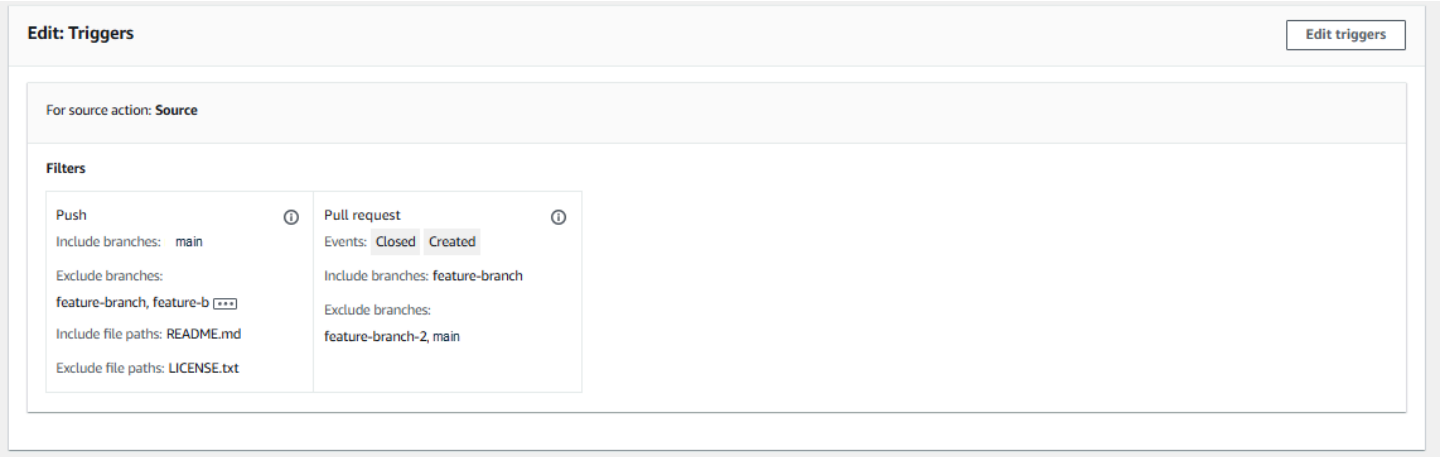
Les objets de configuration des déclencheurs, tels qu'un déclencheur contenant un type d'événement push et un type d'événement pull request, utilisent une opération OR entre les deux types d'événements. L'exemple suivant montre une configuration de déclencheur avec un type d'événement push avec la `main` branche incluse et un type d'événement pull request avec la même branche `main` exclue. En outre, le type d'événement push comporte un chemin de fichier `LICENSE.txt` exclu et un chemin de fichier `README.MD` inclus. Pour le second type d'événement, une pull request qui se trouve sur la branche `Closed` ou `Created` sur la `feature-branch` branche (incluse) démarre le pipeline, et le pipeline ne démarre pas lors de la création ou de la fermeture d'une pull request sur la `main` branche `feature-branch-2` ou (exclue). Dans chaque type d'événement, les exclusions ont priorité sur les inclusions. Par exemple, pour un événement de pull request sur la `feature-branch` branche (inclus pour la pull request), le type d'événement push exclut la `feature-branch` branche, de sorte que le push ne déclenchera pas le pipeline.

Pour l'exemple suivant,

- Le fait d'envoyer un commit à la `main` branche (inclus) pour le chemin du `README.MD` fichier (inclus) déclenchera le pipeline.
- Sur la `feature-branch` branche (exclue), l'envoi d'un commit ne déclenchera pas le pipeline.
- Sur la branche incluse, la modification du chemin du `README.MD` fichier (inclus) déclenche le pipeline.

- Sur la branche incluse, le fait de modifier uniquement le chemin du `LICENSE.TXT` fichier (exclu) ne déclenche pas le pipeline. Toutefois, si le même commit change également `README.MD` (inclus), le pipeline se déclenchera car chaque fichier est évalué indépendamment.
- Sur la `feature-branch` branche, la fermeture d'une pull request déclenchera le pipeline car il `feature-branch` est inclus pour le type d'événement pull request et le type d'événement CLOSED correspond.

L'image suivante montre la configuration.



Voici un exemple de JSON pour la configuration.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "branches": {
                                "includes": [
                                    "main"
                                ],
                                "excludes": [
                                    "feature-branch",
                                    "feature-branch-2"
                                ]
                            },
                            "filePaths": {
                                "includes": [
                                    "README.md"
```

```
                            ],
                            "excludes": [
                                "LICENSE.txt"
                            ]
                        }
                    }
                ],
                "pullRequest": [
                    {
                        "events": [
                            "CLOSED",
                            "OPEN"
                        ],
                        "branches": {
                            "includes": [
                                "feature-branch"
                            ],
                            "excludes": [
                                "feature-branch-2",
                                "main"
                            ]
                        }
                    }
                ]
            }
        }
    ]
},
```

Example 4 : Un déclencheur avec deux types de filtres push avec des inclusions et des exclusions contradictoires

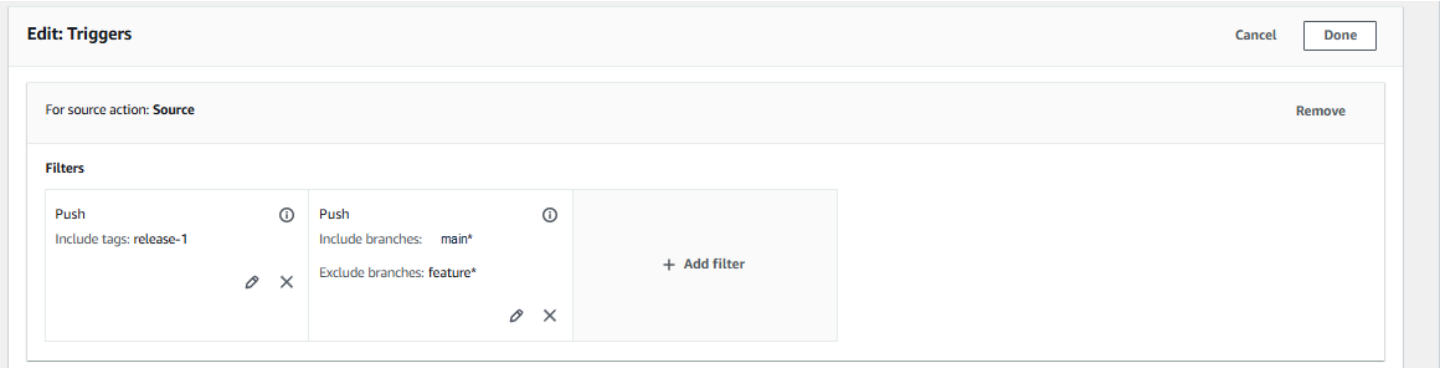
L'image suivante montre un type de filtre push spécifiant de filtrer sur la balise `release-1` (incluse). Un deuxième type de filtre push est ajouté, spécifiant de filtrer sur la branche `main` (inclus) et de ne pas démarrer pour un push vers les `feature*` branches (exclu).

Dans l'exemple suivant :

- Si vous appuyez sur un déclencheur à partir de la balise `release-1` (incluse pour le premier filtre push) sur la `feature-branch` branche (exclue comme `feature*` pour le second filtre push), le pipeline sera déclenché car les deux types de filtres push utilisent une opération OR entre eux, et le premier filtre push (tag`release-1`) correspond.

- Si vous appuyez sur un déclencheur depuis la `main` branche (inclus pour le deuxième filtre Push), le pipeline démarre.

L'exemple suivant de page d'édition montre les deux types de filtres Push et leur configuration pour les inclusions et les exclusions.



Voici un exemple de JSON pour la configuration.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "tags": {
                                "includes": [
                                    "release-1"
                                ]
                            }
                        },
                        {
                            "branches": {
                                "includes": [
                                    "main*"
                                ],
                                "excludes": [
                                    "feature*"
                                ]
                            }
                        }
                    ]
```

```
                }
            }
        ]
    },
```

Example 5 : Déclencheur configuré alors que la configuration d'action par défaut BranchName est utilisée pour un démarrage manuel

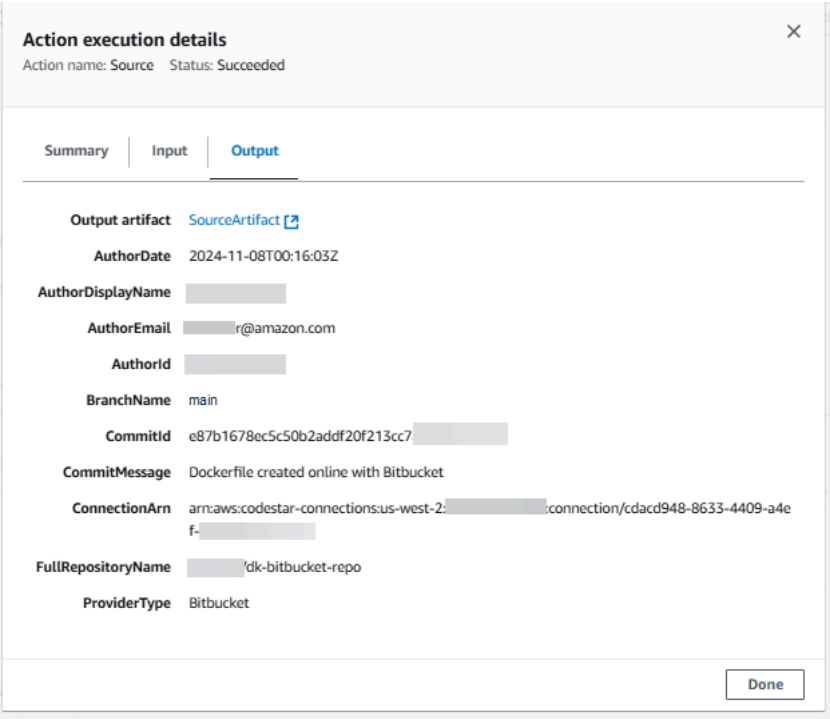
Le `BranchName` champ par défaut de configuration des actions définit une branche unique qui sera utilisée lorsque le pipeline sera démarré manuellement, tandis que les déclencheurs avec filtres peuvent être utilisés pour n'importe quelle branche ou branches que vous spécifiez.

Voici l'exemple de JSON pour la configuration de l'action affichant le `BranchName` champ.

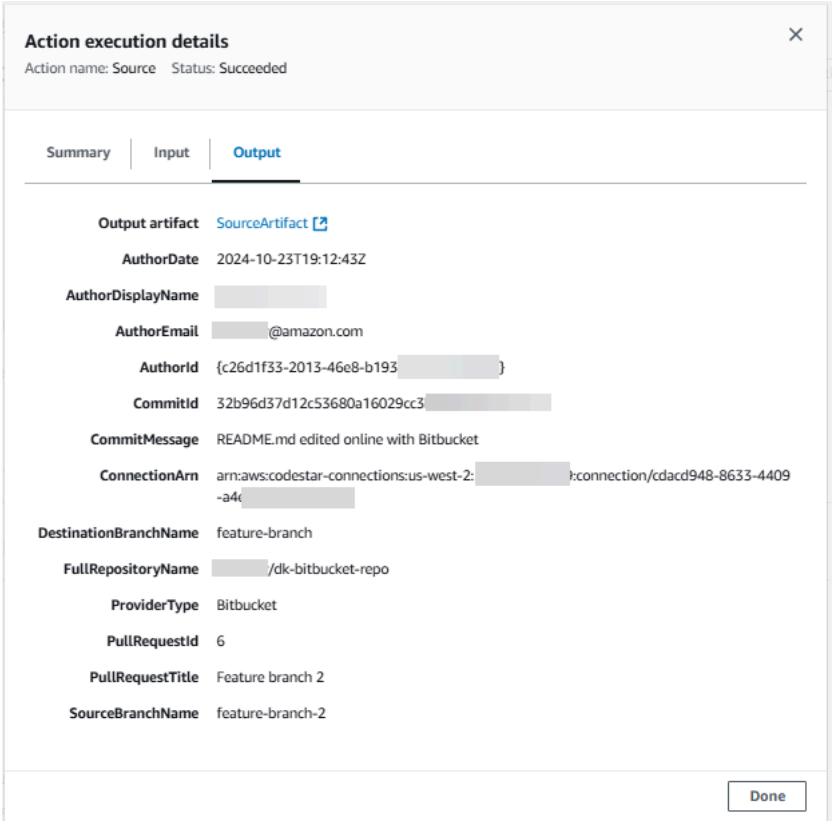
```
{
                "name": "Source",
                "actions": [
                    {
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "provider": "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"
                    }
```

```
        ],
```

L'exemple de sortie d'action suivant montre que la branche principale par défaut a été utilisée lorsque le pipeline a été démarré manuellement.



L'exemple de sortie d'action suivant montre la pull request et la branche qui ont été utilisées pour le déclencheur lors du filtrage par pull request.



Ajouter un déclencheur lors de l'envoi de code sans filtre

Les déclencheurs vous permettent de configurer votre pipeline pour qu'il démarre sur un type d'événement particulier, tel qu'un code push ou une pull request. Les déclencheurs sont configurables pour les actions source avec des connexions qui utilisent l' CodeStarSourceConnection action dans CodePipeline GitHub, telles que Bitbucket et GitLab.

Ajouter un déclencheur (console)

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.

   Les noms et le statut de tous les pipelines associés à votre AWS compte sont affichés.
2. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier. Sinon, suivez ces étapes dans l'assistant de création de pipeline.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Sur la page Modifier, choisissez l'action source que vous souhaitez modifier. Choisissez Modifier les déclencheurs. Choisissez d'ajouter un déclencheur.

5. Dans Type de déclencheur, choisissez Aucun filtre.

Ajouter un déclencheur avec des types d'événements de type « code push » ou « pull request »

Vous pouvez configurer des filtres pour les déclencheurs de pipeline afin que les exécutions de pipeline démarrent pour différents événements Git, tels que l'envoi de balises ou de branches, la modification de chemins de fichiers spécifiques, l'ouverture d'une demande d'extraction dans une branche spécifique, etc. Vous pouvez utiliser la AWS CodePipeline console ou les update-pipeline commandes create-pipeline et du AWS CLI pour configurer les filtres de déclenchement.

> **Note**
>
> Le `BranchName` champ de configuration des actions définit une seule branche, tandis que les déclencheurs dotés de filtres peuvent être utilisés pour n'importe quelle branche ou branches que vous spécifiez. Pour un pipeline où des déclencheurs sont utilisés pour filtrer les branches par requête push ou pull, le pipeline n'utilisera pas la branche de `BranchName` champ par défaut dans la configuration de l'action. Cependant, la branche dans le `BranchName` champ de la configuration de l'action est la branche par défaut lorsque le pipeline est démarré manuellement. Pour obtenir un exemple, consultez 5 : Déclencheur configuré alors que la configuration d'action par défaut BranchName est utilisée pour un démarrage manuel.

Vous pouvez spécifier des filtres pour les types de déclencheurs suivants :

- Push

  Un déclencheur permet de démarrer un pipeline lorsqu'une modification est envoyée à votre référentiel source. L'exécution utilisera le commit de la branche vers laquelle vous poussez (c'est-à-dire la branche de destination). Vous pouvez filtrer les déclencheurs push sur les branches, les chemins de fichiers ou les balises Git.

- Pull request

  Un déclencheur de pull request démarre un pipeline lorsqu'une pull request est ouverte, mise à jour ou fermée dans votre référentiel source. L'exécution utilisera le commit de la branche

source depuis laquelle vous extrayez (c'est-à-dire la branche source). Vous pouvez filtrer les déclencheurs de demandes d'extraction sur les branches et les chemins de fichiers.

Les types d'événements pris en charge pour les pull requests sont les suivants. Tous les autres événements de pull request sont ignorés.

- Ouvert
- Mis à jour
- Fermé (fusionné)

> **Note**
>
> Le comportement de certains événements de pull request peut varier d'un fournisseur à l'autre. Pour en savoir plus, consultez Événements de pull request pour les déclencheurs par fournisseur.

La définition du pipeline vous permet de combiner différents filtres au sein d'une même configuration de déclenchement par pression. Pour plus de détails sur la définition du pipeline, consultezAjouter des filtres pour les types d'événements de requêtes push et pull (CLI). Pour obtenir la liste des définitions de champs, voir les déclencheurs dans la référence sur la structure du pipeline de ce guide.

Rubriques

- Ajouter des filtres pour les types d'événements de type push et pull request (console)
- Ajouter des filtres pour les types d'événements de requêtes push et pull (CLI)
- Ajoutez des filtres pour les types d'événements de requêtes push et pull (CloudFormation modèles)

Ajouter des filtres pour les types d'événements de type push et pull request (console)

Vous pouvez utiliser la console pour ajouter des filtres pour les événements push et inclure ou exclure des branches ou des chemins de fichiers.

Ajouter des filtres (console)

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms et l'état de tous les pipelines associés à votre AWS compte sont affichés.

2. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier. Sinon, procédez comme suit dans l'assistant de création de pipeline.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Sur la page Modifier, choisissez l'action source que vous souhaitez modifier. Choisissez Modifier les déclencheurs. Choisissez Spécifier le filtre.
5. Dans Type d'événement, choisissez Push parmi les options suivantes.

   - Choisissez Push pour démarrer le pipeline lorsqu'une modification est envoyée à votre référentiel source. Cette option permet aux champs de spécifier des filtres pour les branches et les chemins de fichiers ou les balises Git.
   - Choisissez Pull request pour démarrer le pipeline lorsqu'une pull request est ouverte, mise à jour ou fermée dans votre référentiel source. Cette option permet aux champs de spécifier des filtres pour les branches de destination et les chemins de fichiers.

6. Sous Push, dans Type de filtre, choisissez l'une des options suivantes.

   - Choisissez Branche pour spécifier les branches de votre référentiel source que le déclencheur surveille afin de savoir quand démarrer l'exécution d'un flux de travail. Dans Include, entrez des modèles pour les noms de branches au format global que vous souhaitez spécifier pour la configuration du déclencheur afin de démarrer votre pipeline en cas de modification des branches spécifiées. Dans Exclure, entrez les modèles d'expressions régulières pour les noms de branches au format global que vous souhaitez spécifier pour que la configuration du déclencheur les ignore et ne démarre pas votre pipeline en cas de modification des branches spécifiées. Pour plus d'informations, consultez Utilisation de modèles globulaires dans la syntaxe.

     > **Note**
     >
     > Si les options d'inclusion et d'exclusion ont toutes deux le même modèle, la valeur par défaut est d'exclure le modèle.

Vous pouvez utiliser des modèles globulaires pour définir les noms de vos branches. Par exemple, utilisez `main*` pour faire correspondre toutes les branches commençant par`main`. Pour plus d'informations, consultez Utilisation de modèles globulaires dans la syntaxe.

Pour un déclencheur, spécifiez les branches vers lesquelles vous appuyez, c'est-à-dire les branches de destination. Pour un déclencheur de pull request, spécifiez les branches de destination vers lesquelles vous ouvrez une pull request.

- (Facultatif) Sous Chemins de fichiers, spécifiez les chemins de fichier pour votre déclencheur. Entrez les noms dans Inclure et Exclure, selon le cas.

  Vous pouvez utiliser des modèles globaux pour définir les noms de chemins de vos fichiers. Par exemple, utilisez pour `prod*` faire correspondre tous les chemins de fichiers commençant par`prod`. Pour plus d'informations, consultez Utilisation de modèles globulaires dans la syntaxe.

- Choisissez Tags pour configurer la configuration du déclencheur du pipeline afin de commencer par les balises Git. Dans Include, entrez des modèles pour les noms de balises au format global que vous souhaitez spécifier pour la configuration du déclencheur afin de démarrer votre pipeline lors de la publication de la ou des balises spécifiées. Dans Exclure, entrez les modèles d'expression régulière pour les noms de balises au format global que vous souhaitez spécifier pour que la configuration du déclencheur ignore et ne démarre pas votre pipeline lors de la publication de la ou des balises spécifiées. Si les options d'inclusion et d'exclusion ont toutes deux le même modèle de balise, la valeur par défaut est d'exclure le modèle de balise.

7. Sous Push, dans Type de filtre, choisissez l'une des options suivantes.

   - Choisissez Branche pour spécifier les branches de votre référentiel source que le déclencheur surveille afin de savoir quand démarrer l'exécution d'un flux de travail. Dans Include, entrez des modèles pour les noms de branches au format global que vous souhaitez spécifier pour la configuration du déclencheur afin de démarrer votre pipeline en cas de modification des branches spécifiées. Dans Exclure, entrez les modèles d'expressions régulières pour les noms de branches au format global que vous souhaitez spécifier pour que la configuration du déclencheur les ignore et ne démarre pas votre pipeline en cas de modification des branches spécifiées. Pour plus d'informations, consultez Utilisation de modèles globulaires dans la syntaxe.

> **Note**
>
> Si les options d'inclusion et d'exclusion ont toutes deux le même modèle, la valeur par défaut est d'exclure le modèle.

Vous pouvez utiliser des modèles globulaires pour définir les noms de vos branches. Par exemple, utilisez `main*` pour faire correspondre toutes les branches commençant par`main`. Pour plus d'informations, consultez Utilisation de modèles globulaires dans la syntaxe.

Pour un déclencheur, spécifiez les branches vers lesquelles vous appuyez, c'est-à-dire les branches de destination. Pour un déclencheur de pull request, spécifiez les branches de destination vers lesquelles vous ouvrez une pull request.

- (Facultatif) Sous Chemins de fichiers, spécifiez les chemins de fichier pour votre déclencheur. Entrez les noms dans Inclure et Exclure, selon le cas.

  Vous pouvez utiliser des modèles globaux pour définir les noms de chemins de vos fichiers. Par exemple, utilisez pour `prod*` faire correspondre tous les chemins de fichiers commençant par`prod`. Pour plus d'informations, consultez Utilisation de modèles globulaires dans la syntaxe.
- Choisissez Pull request pour configurer la configuration du déclencheur du pipeline afin de commencer par les événements de pull request que vous spécifiez.

Ajouter des filtres pour les types d'événements de requêtes push et pull (CLI)

Vous pouvez mettre à jour le JSON du pipeline pour ajouter des filtres pour les déclencheurs.

Pour créer ou mettre AWS CLI à jour votre pipeline, utilisez la `update-pipeline` commande `create-pipeline` ou.

L'exemple de structure JSON suivant fournit une référence pour les définitions de champs ci-dessous`create-pipeline`.

Pour obtenir la liste des définitions de champs, voir les déclencheurs dans la référence sur la structure du pipeline de ce guide.

```
{
    "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"
                        }
                    }
                ]
            }
        ]
```

```
    }
}
```

Ajoutez des filtres pour les types d'événements de requêtes push et pull (CloudFormation modèles)

Vous pouvez mettre à jour la ressource du pipeline CloudFormation pour ajouter un filtrage des déclencheurs.

L'extrait de modèle suivant fournit une référence YAML pour les définitions de champs de déclencheurs. Pour obtenir la liste des définitions de champs, reportez-vous à la section Déclencheurs dans la référence relative à la structure du pipeline de ce guide.

Pour un exemple de modèle complet pour la configuration d'une source de connexion et d'un filtre de déclenchement, voir Pipeline en deux étapes et configuration de déclencheur dans le Guide de CloudFormation l'utilisateur.

```
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
```

Ajouter un déclencheur pour désactiver la détection des modifications

Les déclencheurs vous permettent de configurer votre pipeline pour qu'il démarre sur un type d'événement particulier, tel qu'un push de code ou une pull request. Les déclencheurs sont configurables pour les actions source avec des connexions qui utilisent l'`CodeStarSourceConnection`action dans CodePipeline, par exemple GitHub, Bitbucket et GitLab.

Ajout d'un déclencheur pour désactiver la détection des modifications (console)

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms et l'état de tous les pipelines associés à votre AWS compte sont affichés.
2. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier. Sinon, procédez comme suit dans l'assistant de création de pipeline.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Sur la page Modifier, choisissez l'action source que vous souhaitez modifier. Choisissez Modifier les déclencheurs. Choisissez d'ajouter un déclencheur.
5. Dans Type de déclencheur, choisissez Ne pas détecter les modifications.

Démarrage et arrêt manuels des pipelines

Vous pouvez utiliser la AWS CodePipeline console ou le AWS CLI pour démarrer et arrêter les pipelines manuellement. Il existe des méthodes automatisées pour démarrer des pipelines avec détection des modifications basée sur les événements, comme détaillé dans, ou configuration des déclencheurs, comme détaillé dans.

Rubriques

- Démarrez un pipeline dans CodePipeline
- Arrêtez l'exécution d'un pipeline dans CodePipeline

Démarrez un pipeline dans CodePipeline

Chaque exécution de pipeline peut être démarrée en fonction d'un déclencheur différent. Chaque exécution de pipeline peut avoir un type de déclencheur différent, en fonction de la façon dont le pipeline est démarré. Le type de déclencheur pour chaque exécution est indiqué dans l'historique des exécutions d'un pipeline. Les types de déclencheurs peuvent dépendre du fournisseur d'actions source comme suit :

> **Note**
>
> Vous ne pouvez pas spécifier plus d'un déclencheur par action source.

- Création d'un pipeline : lorsqu'un pipeline est créé, son exécution démarre automatiquement. Il s'agit du type de `CreatePipeline` déclencheur indiqué dans l'historique des exécutions.
- Modifications apportées aux objets révisés : cette catégorie représente le type de `PutActionRevision` déclencheur dans l'historique des exécutions.
- Détection des modifications sur la branche et validation pour un push de code : cette catégorie représente le type de `CloudWatchEvent` déclencheur dans l'historique des exécutions. Lorsqu'une modification est détectée dans un commit source et une branche dans le référentiel source, votre pipeline démarre. Ce type de déclencheur utilise la détection automatique des modifications. Les fournisseurs d'actions source qui utilisent ce type de déclencheur sont S3 et CodeCommit. Ce type est également utilisé pour un calendrier qui démarre votre pipeline. Consultez Démarrez un pipeline selon un calendrier.

- Sondage des modifications de source : cette catégorie représente le type de `PollForSourceChanges` déclencheur dans l'historique des exécutions. Lorsqu'une modification est détectée dans un commit source et une branche dans le référentiel source par le biais d'un sondage, votre pipeline démarre. Ce type de déclencheur n'est pas recommandé et doit être migré pour utiliser la détection automatique des modifications. Les fournisseurs d'actions source qui utilisent ce type de déclencheur sont S3 et CodeCommit.
- Événements Webhook pour des sources tierces : cette catégorie représente le type de `Webhook` déclencheur dans l'historique des exécutions. Lorsqu'un changement est détecté par un événement webhook, votre pipeline démarre. Ce type de déclencheur utilise la détection automatique des modifications. Les fournisseurs d'actions source qui utilisent ce type de déclencheur sont des connexions configurées pour le transfert de code (Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab autogérées). GitLab.com
- Événements WebHookV2 pour des sources tierces : cette catégorie représente le type de `WebhookV2` déclencheur dans l'historique des exécutions. Ce type est destiné aux exécutions déclenchées en fonction des déclencheurs définis dans la définition du pipeline. Lorsqu'une version avec une balise Git spécifiée est détectée, votre pipeline démarre. Vous pouvez utiliser les balises Git pour marquer un commit avec un nom ou un autre identifiant qui aidera les autres utilisateurs du référentiel à comprendre son importance. Vous pouvez également utiliser des balises Git pour identifier un commit particulier dans l'historique d'un dépôt. Ce type de déclencheur désactive la détection automatique des modifications. Les fournisseurs d'actions source qui utilisent ce type de déclencheur sont des connexions configurées pour les balises Git (Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com).
- Démarrage manuel d'un pipeline : cette catégorie représente le type de `StartPipelineExecution` déclencheur dans l'historique des exécutions. Vous pouvez utiliser la console ou le AWS CLI pour démarrer un pipeline manuellement. Pour plus d'informations, consultez Lancement manuel d'un pipeline.
- RollbackStage: Cette catégorie représente le type de `RollbackStage` déclencheur dans l'historique des exécutions. Vous pouvez utiliser la console ou le AWS CLI pour revenir en arrière d'une étape manuellement ou automatiquement. Pour plus d'informations, consultez Configuration de la restauration des étapes.

Lorsque vous ajoutez une action source à votre pipeline qui utilise des types de déclencheurs de détection automatique des modifications, les actions fonctionnent avec des ressources supplémentaires. La création de chaque action source est détaillée dans des sections distinctes en raison de ces ressources supplémentaires pour la détection des modifications. Pour plus de détails

sur chaque fournisseur source et les méthodes de détection des modifications requises pour la détection automatique des modifications, consultezActions à la source et méthodes de détection des modifications.

Rubriques

- Lancement manuel d'un pipeline
- Démarrez un pipeline selon un calendrier
- Démarrer un pipeline avec un remplacement de révision de la source

Lancement manuel d'un pipeline

Par défaut, un pipeline se lance automatiquement lors de sa création et chaque fois qu'une modification est apportée à un référentiel source. Cependant, vous pouvez exécuter à nouveau la révision la plus récente par le biais du pipeline une deuxième fois. Vous pouvez utiliser la CodePipeline console ou la start-pipeline-execution commande AWS CLI and pour réexécuter manuellement la dernière révision dans votre pipeline.

Rubriques

- Démarrage manuel d'un pipeline (console)
- Démarrage manuel d'un pipeline (interface de ligne de commande)

Démarrage manuel d'un pipeline (console)

Pour lancer manuellement un pipeline et la plus récente révision dans un pipeline

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Dans Nom, choisissez le nom du pipeline que vous souhaitez lancer.
3. Sur la page des détails du pipeline, choisissez Release change. Si le pipeline est configuré pour transmettre des paramètres (variables du pipeline), le fait de sélectionner Release change ouvre la fenêtre Release change. Dans Variables de pipeline, dans le ou les champs des variables au niveau du pipeline, entrez la ou les valeurs que vous souhaitez transmettre pour cette exécution de pipeline. Pour de plus amples informations, veuillez consulter Référence des variables.

Cette opération exécute la révision la plus récente disponible dans chaque emplacement source spécifié d'une action source à travers le pipeline.

Démarrage manuel d'un pipeline (interface de ligne de commande)

Pour lancer manuellement un pipeline et la version plus récente d'un artefact dans un pipeline

1. Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et exécutez la start-pipeline-execution commande en spécifiant le nom du pipeline que vous souhaitez démarrer. AWS CLI Par exemple, pour lancer l'exécution de la dernière modification à travers un pipeline nommé *MyFirstPipeline* :

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

   Pour démarrer un pipeline dans lequel les variables sont configurées au niveau du pipeline, utilisez la start-pipeline-execution commande avec l'--variablesargument optionnel pour démarrer le pipeline et ajouter les variables qui seront utilisées lors de l'exécution. Par exemple, pour ajouter une variable `var1` dont la valeur est de1, utilisez la commande suivante :

   ```
   aws codepipeline start-pipeline-execution --name MyFirstPipeline --variables
    name=var1,value=1
   ```

2. Pour vérifier la réussite, affichez l'objet retourné. Cette commande renvoie un ID d'exécution similaire à ce qui suit :

   ```
   {
       "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
   }
   ```

   > **Note**
   >
   > Après avoir démarré le pipeline, vous pouvez suivre sa progression dans la CodePipeline console ou en exécutant la get-pipeline-state commande. Pour plus d'informations, consultez Afficher les pipelines (console) et Affichage des détails et de l'historique d'un pipeline (interface de ligne de commande).

Démarrez un pipeline selon un calendrier

Vous pouvez définir une règle EventBridge pour démarrer un pipeline selon un calendrier.

Créez une EventBridge règle qui planifie le démarrage de votre pipeline (console)

Pour créer une EventBridge règle avec un calendrier comme source d'événement

1. Ouvrez la EventBridge console Amazon à l'adresse https://console.aws.amazon.com/events/.
2. Dans le panneau de navigation, choisissez Rules.
3. Choisissez Créer une règle, puis sous Détails de la règle, choisissez Planifier.
4. Configurez le programme à l'aide d'un taux ou d'une expression fixe. Pour plus d'informations, consultez Expression de planification pour les règles.
5. Dans Targets, choisissez CodePipeline.
6. Entrez l'ARN du pipeline pour l'exécution du pipeline pour ce calendrier.

   > **Note**
   >
   > L'ARN du pipeline se trouve dans les paramètres de la console. Consultez Afficher l'ARN du pipeline et l'ARN du rôle de service (console).

7. Choisissez l'une des options suivantes pour créer ou spécifier un rôle de service IAM qui EventBridge autorise l'appel de la cible associée à votre EventBridge règle (dans ce cas, la cible est CodePipeline).

   - Choisissez Créer un nouveau rôle pour cette ressource spécifique afin de créer un rôle de service qui accorde EventBridge les autorisations nécessaires pour démarrer les exécutions de votre pipeline.
   - Choisissez Utiliser le rôle existant pour saisir un rôle de service qui accorde EventBridge les autorisations nécessaires pour démarrer les exécutions de votre pipeline.
8. Choisissez Configure details (Configurer les détails).
9. Sur la page Configurer les détails de la règle, entrez un nom et une description pour la règle, puis cochez État pour activer la règle.
10. Si la règle vous convient, choisissez Créer une règle.

Créez une EventBridge règle qui planifie le démarrage de votre pipeline (CLI)

Pour utiliser le AWS CLI pour créer une règle, appelez la put-rule commande en spécifiant :

- Un nom qui identifie de façon unique la règle que vous créez. Ce nom doit être unique pour tous les pipelines que vous créez et qui CodePipeline sont associés à votre AWS compte.
- L'expression de planification pour la règle.

Pour créer une EventBridge règle avec un calendrier comme source d'événement

1. Appelez la commande put-rule et incluez les paramètres `--name` et `--schedule-expression`.

   Exemples :

   L'exemple de commande suivant --schedule-expression permet de créer une règle appelée `MyRule2` qui filtre EventBridge selon un calendrier.

   ```
   aws events put-rule --schedule-expression 'cron(15 10 ? * 6L 2002-2005)' --name
    MyRule2
   ```

2. Pour ajouter CodePipeline en tant que cible, appelez la put-targets commande et incluez les paramètres suivants :

   - Le paramètre `--rule` s'utilise avec le la règle `rule_name` que vous avez créée à l'aide de la commande put-rule.
   - Le paramètre `--targets` s'utilise avec l'ID de liste `Id` de la cible figurant dans la liste des cibles et l'ARN du pipeline cible.

   L'exemple de commande suivant spécifie que pour la règle appelée `MyCodeCommitRepoRule`, l'`Id` cible est composé du numéro un, ce qui indique qu'il s'agit de la règle 1 dans une liste de cibles pour la règle. L'exemple de commande spécifie également un exemple d'ARN pour le pipeline. Le pipeline démarre lorsque des modifications sont effectuées dans le référentiel.

   ```
   aws events put-targets --rule MyCodeCommitRepoRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

3. Accordez les autorisations nécessaires EventBridge CodePipeline pour invoquer la règle. Pour plus d'informations, consultez la section Utilisation de politiques basées sur les ressources pour Amazon. EventBridge

a. Utilisez l'exemple suivant pour créer la stratégie d'approbation qui permet à EventBridge d'assumer le rôle de service. Nommez-la `trustpolicyforEB.json`.

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "events.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

b. Utilisez la commande suivante pour créer le rôle `Role-for-MyRule` et attachez la stratégie d'approbation.

   ```
   aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
    file://trustpolicyforEB.json
   ```

c. Créez le JSON de stratégie d'autorisations comme indiqué dans cet exemple pour le pipeline nommé `MyFirstPipeline`. Nommez la stratégie d'autorisations `permissionspolicyforEB.json`.

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "codepipeline:StartPipelineExecution"
               ],
               "Resource": [
   ```

```
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Utilisez la commande suivante pour attacher la nouvelle stratégie d'autorisations `CodePipeline-Permissions-Policy-for-EB` au rôle `Role-for-MyRule` que vous avez créé.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforCWE.json
```

Démarrer un pipeline avec un remplacement de révision de la source

Vous pouvez utiliser des remplacements pour démarrer un pipeline avec un ID de révision source spécifique que vous fournissez pour l'exécution du pipeline. Par exemple, si vous souhaitez démarrer un pipeline qui traitera un ID de validation spécifique à partir de votre CodeCommit source, vous pouvez ajouter l'ID de validation en tant que remplacement lorsque vous démarrez votre pipeline.

> **Note**
>
> Vous pouvez également créer un remplacement de source à l'aide d'une entrée de transformation d'entrée EventBridge pour utiliser l'`revisionValue`entrée de votre événement de pipeline, où le `revisionValue` est dérivé de la variable d'événement source pour votre clé d'objet, votre commit ou votre ID d'image. Pour plus d'informations, consultez l'étape facultative pour la saisie de la transformation d'entrée incluse dans les procédures sous Actions et ressources relatives aux sources Amazon ECR EventBridgeConnexion aux actions de source Amazon S3 avec une source activée pour les événements, ouCodeCommit actions à la source et EventBridge.

Il existe quatre types de révision des sources pour `revisionType` :

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

- `S3_OBJECT_KEY`

> **Note**
>
> Pour les révisions de source `COMMIT_ID` et les `IMAGE_DIGEST` types de révisions, l'ID de révision source s'applique à tout le contenu du référentiel, dans toutes les branches.

> **Note**
>
> Pour les `S3_OBJECT_VERSION_ID` et `S3_OBJECT_KEY` types de révisions de source, l'un ou l'autre des types peut être utilisé indépendamment, ou ils peuvent être utilisés ensemble pour remplacer la source par un identifiant de version spécifique ObjectKey . Pour`S3_OBJECT_KEY`, le paramètre de configuration `AllowOverrideForS3ObjectKey` doit être défini sur`true`. Pour plus d'informations sur les paramètres de configuration des sources S3, consultezParamètres de configuration.

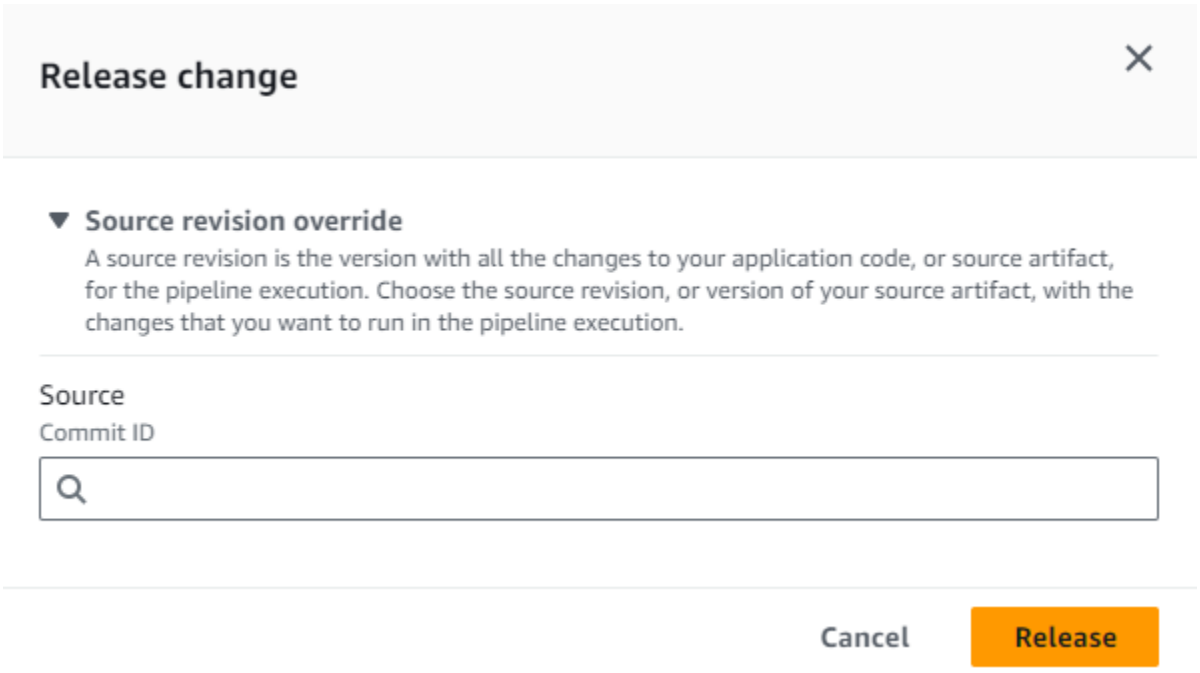
Rubriques

- Démarrer un pipeline avec un remplacement de révision de la source (console)
- Démarrez un pipeline avec un remplacement de révision de la source (CLI)

Démarrer un pipeline avec un remplacement de révision de la source (console)

Pour lancer manuellement un pipeline et la plus récente révision dans un pipeline

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Dans Nom, choisissez le nom du pipeline que vous souhaitez lancer.
3. Sur la page des détails du pipeline, choisissez Release change. Le choix de modification de version ouvre la fenêtre de modification de version. Pour remplacer la révision de la source, choisissez la flèche pour développer le champ. Dans Source, entrez l'ID de révision de la source. Par exemple, si votre pipeline possède une CodeCommit source, choisissez l'ID de validation dans le champ que vous souhaitez utiliser.



Démarrez un pipeline avec un remplacement de révision de la source (CLI)

Pour démarrer manuellement un pipeline et exécuter l'ID de révision source spécifié pour un artefact via un pipeline

1. Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et exécutez la start-pipeline-execution commande en spécifiant le nom du pipeline que vous souhaitez démarrer. AWS CLI Vous utilisez également --source-revisions cet argument pour fournir l'ID de révision source. La révision source est composée de ActionName, RevisionType et RevisionValue. Les valeurs RevisionType valides sont. `COMMIT_ID | IMAGE_DIGEST | S3_OBJECT_VERSION_ID | S3_OBJECT_KEY`

   Dans l'exemple suivant, pour commencer à exécuter la modification spécifiée via un pipeline nommécodecommit-pipeline, la commande suivante spécifie le nom de l'action source Source, un type de `COMMIT_ID` révision et un ID de validation de`78a25c18755ccac3f2a9eec099dEXAMPLE`.

   ```
   aws codepipeline start-pipeline-execution --name codecommit-pipeline --source-
   revisions
    actionName=Source,revisionType=COMMIT_ID,revisionValue=78a25c18755ccac3f2a9eec099dEXAMPLE
    --region us-west-1
   ```

2. Pour vérifier la réussite, affichez l'objet retourné. Cette commande renvoie un ID d'exécution similaire à ce qui suit :

```
{
    "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
}
```

> **Note**
>
> Après avoir démarré le pipeline, vous pouvez suivre sa progression dans la CodePipeline console ou en exécutant la get-pipeline-state commande. Pour plus d'informations, consultez Afficher les pipelines (console) et Affichage des détails et de l'historique d'un pipeline (interface de ligne de commande).

Arrêtez l'exécution d'un pipeline dans CodePipeline

Lorsque l'exécution d'un pipeline commence à travers un pipeline, elle procède étape par étape et verrouille la phase pendant que toutes les exécutions d'action de la phase sont en cours. Ces actions en cours doivent être gérées de manière à ce que, lorsque l'exécution du pipeline est arrêtée, elles soient autorisées à se terminer ou à être abandonnées.

Il existe deux manières d'arrêter l'exécution d'un pipeline :

- Arrêter et attendre : AWS CodePipeline attend pour arrêter l'exécution jusqu'à ce que toutes les actions en cours soient terminées (c'est-à-dire que les actions aient le `Failed` statut `Succeeded` ou). Cette option conserve les actions en cours. L'exécution est à l'état `Stopping` jusqu'à ce que les actions en cours soient terminées. Ensuite, l'exécution prend l'état `Stopped`. La phase se déverrouille une fois les actions terminées.

  Si vous choisissez d'arrêter et d'attendre, et que vous changez d'avis alors que votre exécution est toujours à l'état `Stopping`, vous pouvez choisir d'abandonner.
- Arrêter et abandonner : AWS CodePipeline arrête l'exécution sans attendre la fin des actions en cours. L'exécution passe brièvement à l'état `Stopping`, le temps que les actions en cours soient abandonnées. Une fois l'exécution arrêtée, l'exécution de l'action est à l'état `Abandoned` tandis que l'exécution du pipeline est à l'état `Stopped`. La phase se déverrouille.

Pour une exécution de pipeline à l'état `Stopped`, les actions de la phase où l'exécution s'est arrêtée peuvent faire l'objet d'une nouvelle tentative.

> **Warning**
>
> Cette option peut entraîner des tâches défaillantes ou hors séquence.

Rubriques

- Arrêt de l'exécution d'un pipeline (console)
- Arrêter une exécution entrante (console)
- Arrêt de l'exécution d'un pipeline (interface de ligne de commande)
- Arrêter une exécution entrante (CLI)

Arrêt de l'exécution d'un pipeline (console)

Vous pouvez utiliser la console pour arrêter l'exécution d'un pipeline. Choisissez une exécution, puis choisissez la méthode pour arrêter l'exécution du pipeline.

> **Note**
>
> Vous pouvez également arrêter l'exécution d'un pipeline qui est une exécution entrante. Pour en savoir plus sur l'arrêt d'une exécution entrante, consultezArrêter une exécution entrante (console).

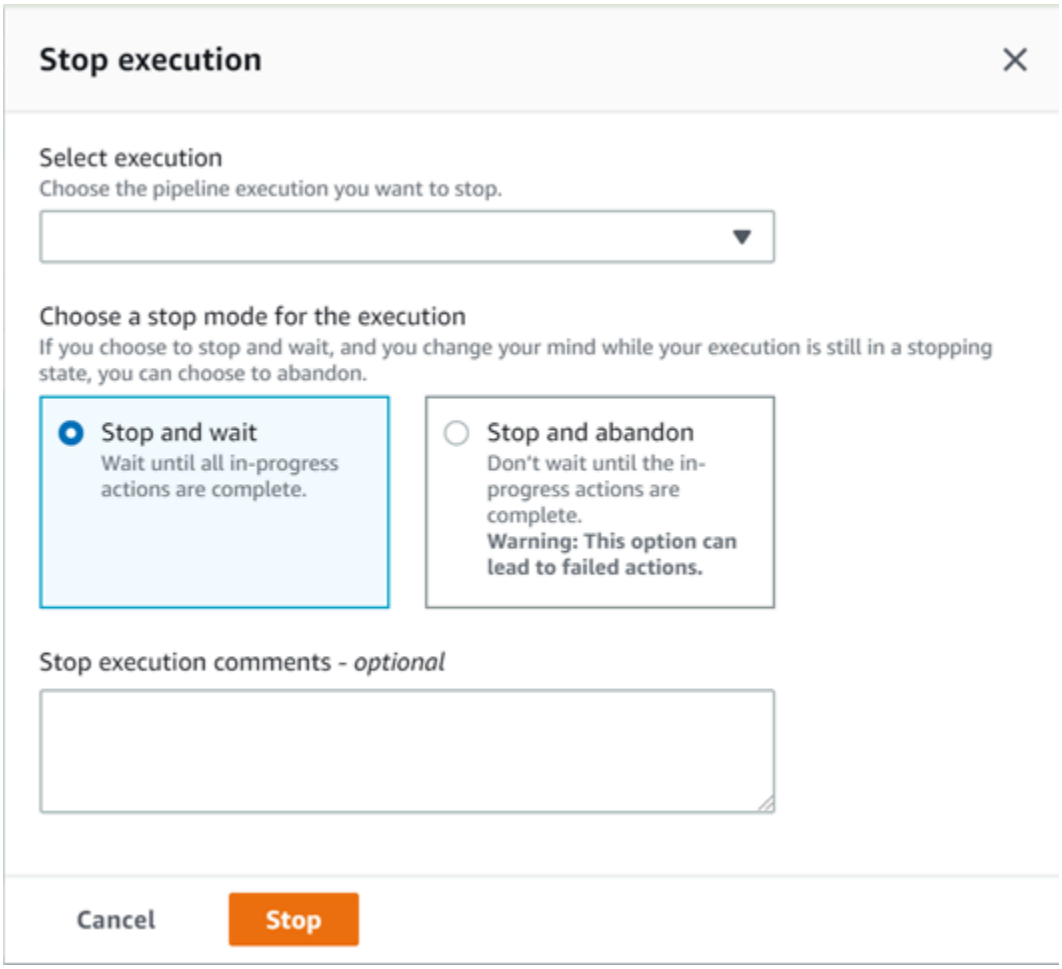
1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Effectuez l'une des actions suivantes :

   > **Note**
   >
   > Avant d'arrêter une exécution, nous vous recommandons de désactiver la transition au début de la phase. De cette façon, lorsque la phase se déverrouille en raison de l'arrêt de l'exécution, elle n'accepte aucune exécution de pipeline ultérieure.

- Dans Name (Nom), choisissez le nom du pipeline avec l'exécution que vous souhaitez arrêter. Sur la page des détails du pipeline, choisissez Stop execution (Arrêter l'exécution).
- Choisissez Afficher l'historique. Sur la page de l'historique, choisissez Stop execution (Arrêter l'exécution).

3. Sur la page Stop execution (Arrêter l'exécution) sous Select execution (Sélectionner l'exécution), choisissez l'exécution que vous souhaitez arrêter.

> **Note**
>
> L'exécution est affichée seulement si elle est toujours en cours. Les exécutions déjà terminées ne s'affichent pas.



4. Sous Select an action to apply to execution (Sélectionner une action à appliquer à l'exécution), choisissez l'une des options suivantes :

   - Pour vous assurer que l'exécution ne s'arrête pas tant que toutes les actions en cours ne sont pas terminées, choisissez Stop and wait (Arrêter et attendre).

     > **Note**
     >
     > Vous ne pouvez pas choisir d'arrêter et d'attendre si l'exécution est déjà à l'état Stopping, mais vous pouvez choisir d'arrêter et d'abandonner.

   - Pour arrêter sans attendre que les actions en cours soient terminées, choisissez Stop and abandon (Arrêter et abandonner).

     > **Warning**
     >
     > Cette option peut entraîner des tâches défaillantes ou hors séquence.

5. (Facultatif) Entrez des commentaires. Ces commentaires, ainsi que l'état de l'exécution, sont affichés sur la page d'historique de l'exécution.
6. Choisissez Arrêter.

   > **Important**
   >
   > Cette action ne peut pas être annulée.

7. Affichez l'état de l'exécution dans la visualisation du pipeline comme suit :

   - Si vous choisissez d'arrêter et d'attendre, l'exécution sélectionnée se poursuit jusqu'à ce que les actions en cours soient terminées.
     - Le message de bannière indiquant que l'action a abouti s'affiche en haut de la console.
     - Dans la phase actuelle, les actions en cours se poursuivent avec un état `InProgress`. Pendant que les actions sont en cours, l'exécution du pipeline est à l'état `Stopping`.

       Une fois les actions terminées (c'est-à-dire lorsque l'action a échoué ou abouti), l'exécution du pipeline passe à l'état `Stopped` et l'action passe à l'état `Succeeded` ou `Failed`. Vous pouvez également afficher l'état de l'action sur la page des détails de l'exécution. Vous

pouvez afficher l'état de l'exécution sur la page de l'historique de l'exécution ou sur la page des détails de l'exécution.

- L'exécution du pipeline passe brièvement à un `Stopping` état, puis il passe à l'état `Stopped`. Vous pouvez afficher l'état de l'exécution sur la page de l'historique de l'exécution ou sur la page des détails de l'exécution.

- Si vous choisissez d'arrêter et d'abandonner, l'exécution n'attend pas que les actions en cours se terminent.
  - Le message de bannière indiquant que l'action a abouti s'affiche en haut de la console.
  - Dans la phase actuelle, les actions en cours passent à l'état `Abandoned`. Vous pouvez également afficher l'état de l'action sur la page des détails de l'exécution.
  - L'exécution du pipeline passe brièvement à un `Stopping` état, puis il passe à l'état `Stopped`. Vous pouvez afficher l'état de l'exécution sur la page de l'historique de l'exécution ou sur la page des détails de l'exécution.

Vous pouvez afficher l'état de l'exécution du pipeline dans la vue de l'historique de l'exécution et la vue de l'historique détaillé.

Arrêter une exécution entrante (console)

Vous pouvez utiliser la console pour arrêter une exécution entrante. Une exécution entrante est une exécution de pipeline qui attend d'entrer dans une phase où la transition a été désactivée. Lorsque la transition est activée, une exécution entrante `InProgress` continue d'entrer dans la phase. Une exécution entrante qui `Stopped` n'entre pas en phase.

> **Note**
>
> Une fois qu'une exécution entrante a été arrêtée, elle ne peut pas être réessayée.

Si aucune exécution entrante n'est visible, cela signifie qu'aucune exécution n'est en attente lors d'une transition de phase désactivée.

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte seront affichés.

2. Choisissez le nom du pipeline pour lequel vous souhaitez arrêter l'exécution entrante. Effectuez l'une des actions suivantes :

   - Dans la vue Pipeline, choisissez l'ID d'exécution entrante, puis choisissez d'arrêter l'exécution.
   - Choisissez le pipeline et choisissez Afficher l'historique. Dans l'historique des exécutions, choisissez l'ID d'exécution entrante, puis choisissez d'arrêter l'exécution.

3. Dans le modal Arrêter l'exécution, suivez les étapes de la section ci-dessus pour sélectionner l'ID d'exécution et spécifier la méthode d'arrêt.

   Utilisez la get-pipeline-state commande pour afficher l'état de l'exécution entrante.

Arrêt de l'exécution d'un pipeline (interface de ligne de commande)

Pour arrêter manuellement un pipeline, utilisez la stop-pipeline-execution commande avec les paramètres suivants : AWS CLI

- ID d'exécution (obligatoire)
- Commentaires (facultatif)
- Nom du pipeline (obligatoire)
- Balise d'abandon (facultatif, la valeur par défaut est false)

Format de la commande :

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --pipeline-
execution-id Execution_ID [--abandon | --no-abandon] [--reason STOP_EXECUTION_REASON]
```

1. Ouvrez une invite de terminal (Linux, macOS ou Unix) ou de commande (Windows).
2. Pour arrêter l'exécution d'un pipeline, choisissez l'une des options suivantes :

   - Pour vous assurer que l'exécution ne s'arrête pas tant que toutes les actions en cours ne sont pas terminées, choisissez d'arrêter et d'attendre. Pour ce faire, incluez le paramètre no-abandon. Par défaut et si vous ne spécifiez pas ce paramètre, la commande s'arrête et attend. Utilisez le AWS CLI pour exécuter la stop-pipeline-execution commande, en spécifiant le nom du pipeline et l'ID d'exécution. Par exemple, pour arrêter un pipeline nommé *MyFirstPipeline* avec l'option d'arrêt et d'attente spécifiée :

```
aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
pipeline-execution-id d-EXAMPLE --no-abandon
```

Par exemple, pour arrêter un pipeline nommé *MyFirstPipeline* avec l'utilisation de l'option par défaut d'arrêt et d'attente et l'inclusion de commentaires :

```
aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
pipeline-execution-id d-EXAMPLE --reason "Stopping execution after the build
 action is done"
```

> **Note**
>
> Vous ne pouvez pas choisir d'arrêter et d'attendre si l'exécution est déjà à l'état Stopping. Vous pouvez choisir d'arrêter et d'abandonner une exécution déjà à l'état Stopping.

- Pour arrêter sans attendre que les actions en cours soient terminées, choisissez d'arrêter et d'abandonner. Incluez le paramètre abandon. Utilisez le AWS CLI pour exécuter la stop-pipeline-execution commande, en spécifiant le nom du pipeline et l'ID d'exécution.

  Par exemple, pour arrêter un pipeline nommé *MyFirstPipeline* en spécifiant l'option d'abandon et en choisissant d'inclure des commentaires :

  ```
  aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
  pipeline-execution-id d-EXAMPLE --abandon --reason "Stopping execution for a bug
   fix"
  ```

Arrêter une exécution entrante (CLI)

Vous pouvez utiliser la CLI pour arrêter une exécution entrante. Une exécution entrante est une exécution de pipeline qui attend d'entrer dans une phase où la transition a été désactivée. Lorsque la transition est activée, une exécution entrante `InProgress` continue d'entrer dans la phase. Une exécution entrante qui `Stopped` n'entre pas en phase.

> **Note**
>
> Une fois qu'une exécution entrante a été arrêtée, elle ne peut pas être réessayée.

Si aucune exécution entrante n'est visible, cela signifie qu'aucune exécution n'est en attente lors d'une transition de phase désactivée.

Pour arrêter manuellement une exécution entrante, utilisez la stop-pipeline-execution commande avec les paramètres suivants : AWS CLI

- ID d'exécution entrante (obligatoire)
- Commentaires (facultatif)
- Nom du pipeline (obligatoire)
- Balise d'abandon (facultatif, la valeur par défaut est false)

Format de la commande :

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --
pipeline-execution-id Inbound_Execution_ID [--abandon | --no-abandon] [--
reason STOP_EXECUTION_REASON]
```

Suivez les étapes de la procédure ci-dessus pour entrer la commande et spécifier la méthode d'arrêt.

Utilisez la get-pipeline-state commande pour afficher l'état de l'exécution entrante.

Afficher l'historique et définir le mode d'exécution des pipelines

Pour analyser la progression du pipeline, vous pouvez consulter l'historique des exécutions du pipeline et des actions. Pour configurer la méthode de publication pour le traitement des exécutions de pipeline, définissez le mode d'exécution.

Rubriques

- Afficher les exécutions dans CodePipeline
- Définir ou modifier le mode d'exécution du pipeline

Afficher les exécutions dans CodePipeline

Vous pouvez utiliser la AWS CodePipeline console ou le AWS CLI pour afficher l'état d'exécution, consulter l'historique des exécutions et réessayer les étapes ou les actions qui ont échoué.

Rubriques

- Affichage de l'historique d'exécution du pipeline (console)
- Affichage du statut d'exécution (console)
- Afficher une exécution entrante (console)
- Affichage des révisions de la source d'exécution du pipeline (console)
- Affichage des exécutions des actions (console)
- Affichage des artefacts d'action et des informations de stockage des artefacts (console)
- Affichage des détails et de l'historique d'un pipeline (interface de ligne de commande)

Affichage de l'historique d'exécution du pipeline (console)

Vous pouvez utiliser la CodePipeline console pour afficher la liste de tous les pipelines de votre compte. Vous pouvez également afficher les informations pour chaque pipeline, comme la dernière fois où des actions ont été exécutées dans le pipeline, si une transition entre les étapes est activée ou désactivée, si des actions ont échoué, etc. Vous pouvez également afficher une page d'historique présentant les détails pour toutes les exécutions de pipeline pour lesquelles l'historique est enregistré.

> **Note**
>
> Lorsque vous passez d'un mode d'exécution à un autre, la vue et l'historique du pipeline peuvent changer. Pour de plus amples informations, veuillez consulter Définir ou modifier le mode d'exécution du pipeline.

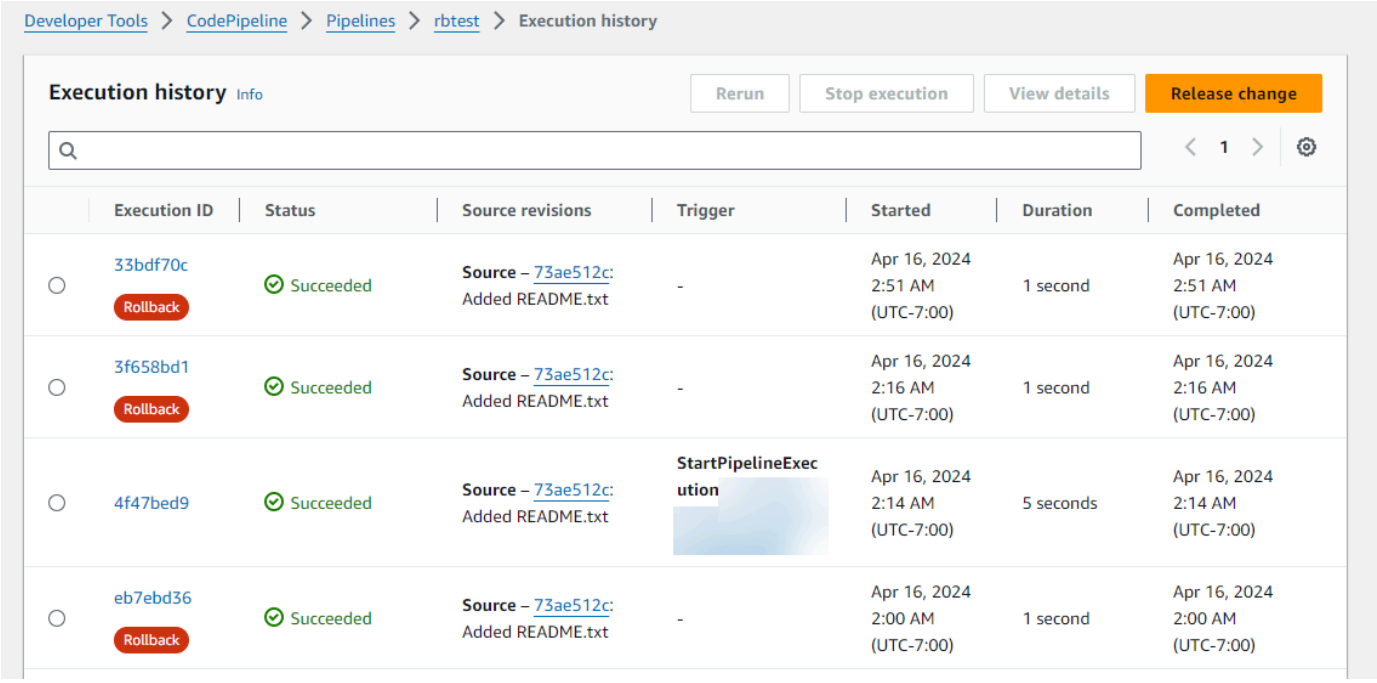
L'historique d'exécution est conservé pendant 12 mois maximum.

Vous pouvez utiliser la console pour afficher l'historique des exécutions d'un pipeline, y compris le statut, la source et la chronologie détaillée de chaque exécution.

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés, ainsi que leur statut.
2. Dans Name, choisissez le nom du pipeline.
3. Choisissez Afficher l'historique.

> **Note**
>
> Pour un pipeline en mode d'exécution PARALLÈLE, la vue principale du pipeline n'affiche pas la structure du pipeline ni les exécutions en cours. Pour un pipeline en mode d'exécution PARALLÈLE, vous accédez à la structure du pipeline en choisissant l'ID de l'exécution que vous souhaitez afficher sur la page d'historique des exécutions. Choisissez Historique dans la barre de navigation de gauche, choisissez l'ID d'exécution pour l'exécution parallèle, puis visualisez le pipeline dans l'onglet Visualisation.



4. Affichez le statut, les révisions source, les détails de modifications et les déclencheurs liés à chaque exécution pour votre pipeline. Les exécutions de pipeline qui ont été annulées afficheront le type d'exécution Rollback sur l'écran des détails de la console. Pour l'échec de l'exécution qui a déclenché la restauration automatique, l'ID d'exécution échouée est affiché.
5. Choisissez une exécution. La vue détaillée affiche les détails de l'exécution, l'onglet Chronologie, l'onglet Visualisation et l'onglet Variables. Les valeurs des variables au niveau du pipeline sont résolues au moment de l'exécution du pipeline et peuvent être consultées dans l'historique des exécutions pour chaque exécution.

> **Note**
>
> Les variables de sortie des actions du pipeline peuvent être consultées dans l'onglet Variables de sortie sous l'historique de chaque exécution d'action.

Affichage du statut d'exécution (console)

Vous pouvez afficher le statut du pipeline dans Statut, sur la page de l'historique d'exécution. Choisissez un lien d'ID d'exécution, puis affichez le statut de l'action.

Les états suivants sont valides pour les pipelines, les étapes et les actions :

> **Note**
>
> Les états de pipeline suivants s'appliquent également à une exécution de pipeline qui est une exécution entrante. Pour consulter une exécution entrante et son statut, consultezAfficher une exécution entrante (console).

Pipeline-level états

État du pipeline	Description
InProgress	L'exécution du pipeline est en cours d'exécution.
Arrêt en cours	L'exécution du pipeline s'arrête en raison d'une demande d'arrêt et d'attente ou d'arrêt et d'abandon de l'exécution du pipeline.
Arrêté(e)	Le processus d'arrêt aboutit et l'exécution du pipeline est arrêtée.
Réussi(e)	L'exécution de pipeline s'est terminée avec succès.
Remplacé	Tandis que l'exécution de ce pipeline était en attente de l'achèvement de l'étape suivante, une autre exécution de pipeline plus récente a avancé et a poursuivi via ce pipeline.
Échec	L'exécution de pipeline ne s'est pas terminée avec succès.

Stage-level états

État de l'étape	Description
InProgress	L'étape est en cours d'exécution.
Arrêt en cours	L'exécution de la phase s'arrête en raison d'une demande d'arrêt et d'attente ou d'arrêt et d'abandon de l'exécution du pipeline.
Arrêté(e)	Le processus d'arrêt aboutit et l'exécution de la phase est arrêtée.
Réussi(e)	L'étape s'est terminée avec succès.
Échec	L'étape ne s'est pas terminée avec succès.

Action-level états

État de l'action	Description
InProgress	L'action est en cours d'exécution.
Abandonné	L'action est abandonnée en raison d'une demande d'arrêt et d'abandon de l'exécution du pipeline.
Réussi(e)	L'action s'est terminée avec succès.
Échec	Pour les action d'approbation, l'état FAILED signifie que l'action a été rejetée par le réviseur ou a échoué en raison d'une configuration d'action incorrecte.

Afficher une exécution entrante (console)

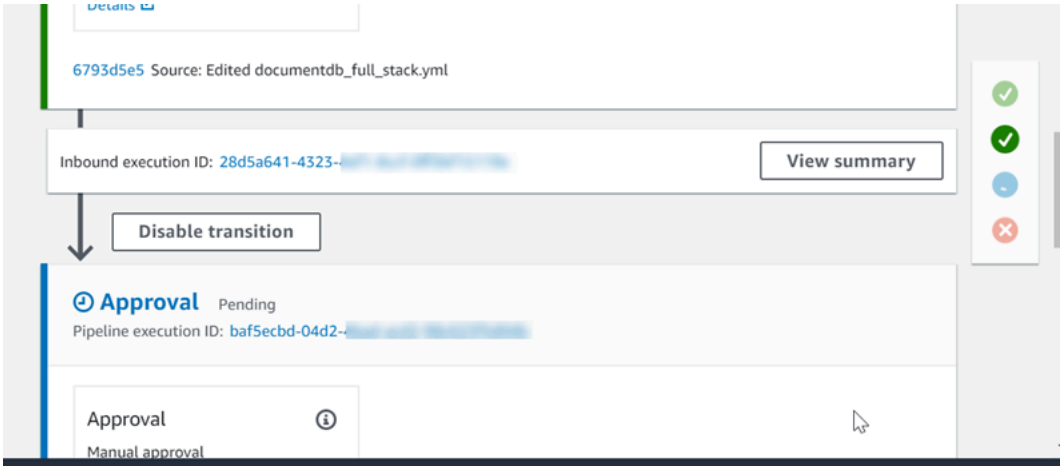
Vous pouvez utiliser la console pour consulter l'état et les détails d'une exécution entrante. Lorsque la transition est activée ou que l'étape devient disponible, une exécution entrante se `InProgress` poursuit et entre dans la phase. Une exécution entrante avec un `Stopped` statut n'entre pas dans la phase. L'état d'exécution d'un flux entrant change `Failed` si le pipeline est modifié. Lorsque vous modifiez un pipeline, toutes les exécutions en cours ne se poursuivent pas et le statut d'exécution passe à`Failed`.

Si aucune exécution entrante n'est visible, cela signifie qu'aucune exécution n'est en attente lors d'une transition de phase désactivée.

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte seront affichés.
2. Choisissez le nom du pipeline dont vous souhaitez afficher l'exécution entrante. Procédez de l'une des manières suivantes :

   - Choisissez View (Afficher). Dans le diagramme du pipeline, dans le champ ID d'exécution entrante situé devant votre transition désactivée, vous pouvez consulter l'ID d'exécution entrante.



Choisissez Afficher le résumé pour afficher les détails de l'exécution, tels que l'ID d'exécution, le déclencheur source et le nom de l'étape suivante.

- Choisissez le pipeline, puis cliquez sur Afficher l'historique.

Affichage des révisions de la source d'exécution du pipeline (console)

Vous pouvez afficher les détails sur les artefacts sources (artefact de sortie créé à la première étape d'un pipeline) utilisés lors de l'exécution d'un pipeline. Ces détails incluent les identifiants comme les identifiants de validation, les commentaires et, si vous utilisez l'interface de ligne de commande, les numéros de version des actions de génération du pipeline. Pour certains types de révision, vous pouvez afficher et ouvrir l'URL de la validation. Les révisions des sources se composent des éléments suivants :

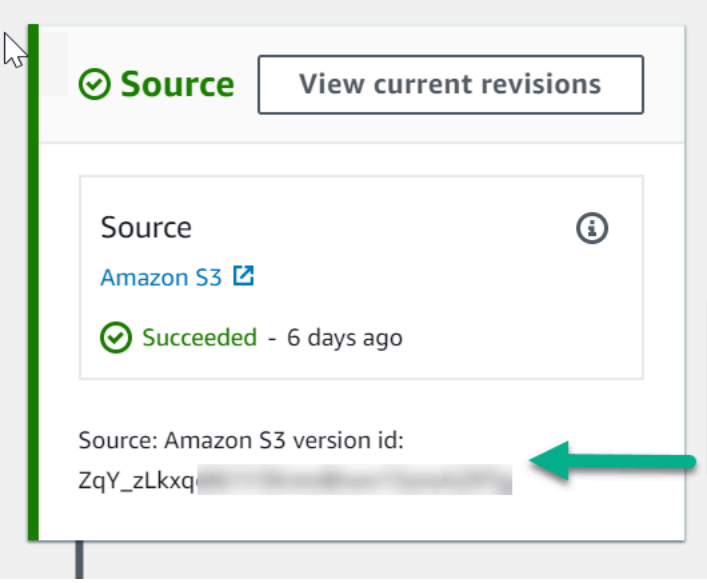
- Résumé : informations récapitulatives sur la révision la plus récente de l'artefact. Pour GitHub et les CodeCommit référentiels, le message de validation. Pour les compartiments ou les actions Amazon S3, le contenu fourni par l'utilisateur d'une clé récapitulative de révision des artefacts du codepipeline indiquée dans les métadonnées de l'objet.
- revisionUrl : URL de révision pour la révision de l'artefact (par exemple, URL du référentiel externe).
- revisionId : ID de révision pour la révision de l'artefact. Par exemple, pour un changement de source dans un GitHub référentiel CodeCommit or, il s'agit de l'ID de validation. Pour les artefacts stockés dans GitHub nos CodeCommit référentiels, l'ID de validation est lié à une page de détails de validation.

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

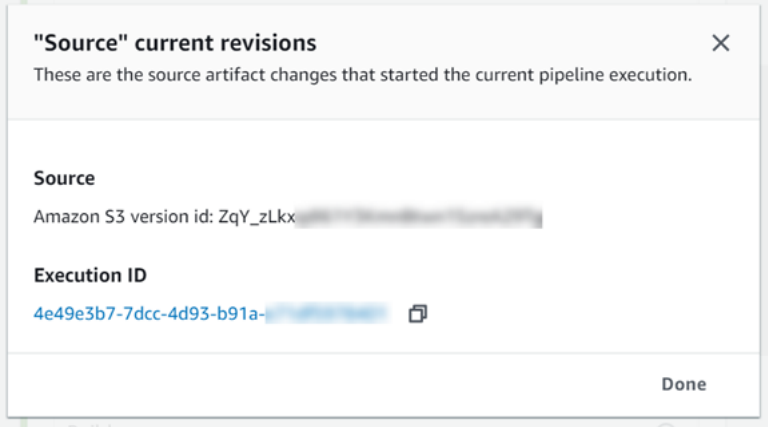
   Les noms de tous les pipelines qui vous sont associés Compte AWS seront affichés.

2. Choisissez le nom du pipeline pour lequel vous souhaitez afficher les informations de révision source. Effectuez l'une des actions suivantes :

   - Choisissez Afficher l'historique. Dans Source des révisions, modifiez la source de chaque exécution répertoriée.
   - Recherchez une action pour laquelle vous souhaitez afficher les informations de révision source, puis localisez les informations relatives à la révision au bas de l'étape :



   Choisissez View current revisions (Afficher les révisions actuelles) pour afficher les informations source. À l'exception des artefacts stockés dans des compartiments Amazon S3, les identifiants tels que les identifiants de validation figurant dans cette vue détaillée des informations sont liés aux pages d'informations sources pour les artefacts.



Affichage des exécutions des actions (console)

Vous pouvez afficher les détails des actions d'un pipeline, comme les ID des exécutions des actions, les artefacts d'entrée, ceux de sortie et le statut. Vous pouvez consulter les détails des actions en choisissant un pipeline dans la console, puis en sélectionnant un ID d'exécution.

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés.
2. Choisissez le nom du pipeline pour lequel vous souhaitez afficher les détails de l'action, puis sélectionnez Afficher l'historique.
3. Dans ID d'exécution, choisissez l'ID d'exécution dont vous souhaitez afficher les détails d'exécution des actions.
4. Vous pouvez consulter les informations suivantes dans l'onglet Chronologie :

   a. Dans Nom de l'action, choisissez le lien pour ouvrir une page de détails de l'action où vous pouvez afficher le statut, le nom de l'étape, le nom de l'action, les données de configuration et les informations sur les artefacts.

   b. Dans le champ Fournisseur, choisissez le lien permettant d'afficher les détails sur le fournisseur de l'action. Par exemple, dans l'exemple de pipeline précédent, si vous choisissez entre CodeDeploy les étapes de préparation ou de production, la page de CodeDeploy console de l' CodeDeploy application configurée pour cette étape s'affiche.

Affichage des artefacts d'action et des informations de stockage des artefacts (console)

Vous pouvez afficher les détails sur les artefacts d'entrée et de sortie pour une action. Vous pouvez également choisir un lien qui vous conduit aux d'informations d'artefact pour cette action. Etant donné que le magasin d'artefacts utilise la gestion des versions, chaque exécution d'action possède un emplacement d'artefacts d'entrée et de sortie unique.

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés.
2. Choisissez le nom du pipeline pour lequel vous souhaitez afficher les détails de l'action, puis sélectionnez Afficher l'historique.
3. Dans ID d'exécution, choisissez l'ID d'exécution dont vous souhaitez afficher les détails des actions.
4. Dans l'onglet Chronologie, dans Nom de l'action, choisissez le lien permettant d'ouvrir une page sur les détails de l'action.
5. Sur la page de détails, dans l'onglet Exécution, visualisez l'état et le calendrier d'exécution de l'action.
6. Dans l'onglet Configuration, visualisez la configuration des ressources pour l'action (par exemple, le CodeBuild nom du projet de génération).
7. Dans l'onglet Artefacts, visualisez les détails de l'artefact dans Type d'artefact et Fournisseur d'artefacts. Choisissez le lien sous Nom de l'artefact pour afficher les artefacts dans le magasin d'artefacts.
8. Dans l'onglet Variables de sortie, visualisez les variables résolues à partir des actions du pipeline pour l'exécution de l'action.

Affichage des détails et de l'historique d'un pipeline (interface de ligne de commande)

Vous pouvez exécuter les commandes suivantes pour afficher les détails sur vos pipelines et exécutions de pipeline :

- list-pipelinescommande pour afficher un résumé de tous les pipelines associés à votre Compte AWS.
- Commande get-pipeline pour vérifier les détails d'un seul pipeline.
- list-pipeline-executions pour afficher des résumés des exécutions les plus récentes d'un pipeline.
- get-pipeline-executionpour afficher des informations sur une exécution d'un pipeline, ainsi que des informations sur les artéfacts, l'ID d'exécution du pipeline, et le nom, la version et l'état du pipeline
- get-pipeline-state commande pour afficher le pipeline, l'étape et le statut de l'action.
- list-action-executions pour afficher les détails de l'exécution de l'action d'un pipeline.

Rubriques

- Afficher l'historique des exécutions avec list-pipeline-executions (CLI)
- Afficher l'état du pipeline avec get-pipeline-state (CLI)
- Afficher l'état d'exécution des appels entrants avec get-pipeline-state (CLI)
- Afficher l'état et les révisions des sources avec get-pipeline-execution (CLI)
- Afficher les exécutions d'actions avec list-action-execution (CLI)

Afficher l'historique des exécutions avec list-pipeline-executions (CLI)

Vous pouvez afficher l'historique d'exécution du pipeline.

- Pour afficher les détails sur les dernières exécutions d'un pipeline, exécutez la commande list-pipeline-executions, en spécifiant le nom unique du pipeline. Par exemple, pour afficher des informations sur l'état actuel d'un pipeline nommé *MyFirstPipeline*, saisissez ce qui suit :

  ```
  aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline
  ```

  Cette commande renvoie des informations récapitulatives sur toutes les exécutions de pipeline pour lesquelles l'historique a été enregistré. Le récapitulatif comprend les heures de démarrage et de fin, la durée et l'état.

  Les exécutions de pipeline qui ont été annulées indiqueront le type d'exécution`Rollback`. Pour l'échec de l'exécution qui a déclenché la restauration automatique, l'ID d'exécution échouée est affiché.

L'exemple suivant montre les données renvoyées pour un pipeline nommé *MyFirstPipeline* pour lequel trois exécutions ont eu lieu :

```
{
    "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"
```

```
        }
```

Pour plus de détails sur l'exécution d'un pipeline, exécutez get-pipeline-execution, en spécifiant l'ID unique de l'exécution du pipeline. Par exemple, pour afficher plus de détails sur la première exécution de l'exemple précédent, saisissez les éléments suivants :

```
aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
execution-id 7cf7f7cb-3137-539g-j458-d7eu3EXAMPLE
```

Cette commande renvoie des informations récapitulatives sur l'exécution d'un pipeline, ainsi que des informations sur les artefacts, l'ID d'exécution du pipeline, et le nom, la version et l'état du pipeline.

L'exemple suivant montre les données renvoyées pour un pipeline nommé *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"
            }
        ]
    }
}
```

Afficher l'état du pipeline avec get-pipeline-state (CLI)

Vous pouvez utiliser l'interface de ligne de commande pour afficher l'état du pipeline, de l'étape et de l'action.

- Pour afficher les détails relatifs à l'état actuel d'un pipeline, exécutez la commande get-pipeline-state, en spécifiant le nom unique du pipeline. Par exemple, pour afficher des informations sur l'état actuel d'un pipeline nommé *MyFirstPipeline*, saisissez ce qui suit :

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

Cette commande affiche l'état actuel de toutes les étapes du pipeline et le statut des actions au cours de ces étapes.

L'exemple suivant montre les données qui s'affichent pour un pipeline de trois étapes nommé *MyFirstPipeline*, dans lequel les deux premières étapes et actions ont réussi, la troisième est en échec et la transition entre les deuxième et troisième étapes est désactivée :

```
{
    "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"
        }
    ]
}
```

Afficher l'état d'exécution des appels entrants avec get-pipeline-state (CLI)

Vous pouvez utiliser l'interface de ligne de commande pour afficher l'état de l'exécution entrante. Lorsque la transition est activée ou que l'étape devient disponible, une exécution entrante se `InProgress` poursuit et entre dans la phase. Une exécution entrante avec un `Stopped` statut n'entre pas dans la phase. L'état d'exécution d'un flux entrant change `Failed` si le pipeline est modifié. Lorsque vous modifiez un pipeline, toutes les exécutions en cours ne se poursuivent pas et le statut d'exécution passe à`Failed`.

- Pour afficher les détails relatifs à l'état actuel d'un pipeline, exécutez la commande get-pipeline-state, en spécifiant le nom unique du pipeline. Par exemple, pour afficher des informations sur l'état actuel d'un pipeline nommé *`MyFirstPipeline`*, saisissez ce qui suit :

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

  Cette commande affiche l'état actuel de toutes les étapes du pipeline et le statut des actions au cours de ces étapes. La sortie indique également l'ID d'exécution du pipeline à chaque étape et indique s'il existe un ID d'exécution entrante pour une étape avec une transition désactivée.

  L'exemple suivant montre les données renvoyées pour un pipeline en deux étapes nommé*`MyFirstPipeline`*, où la première étape indique une transition activée et une exécution de pipeline réussie, et la seconde étape, nomméeBeta, montre une transition désactivée et un ID d'exécution entrante. L'exécution entrante peut avoir un `FAILED` état `InProgressStopped`, ou.

  ```
  {
      "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
}
```

Afficher l'état et les révisions des sources avec get-pipeline-execution (CLI)

Vous pouvez afficher les détails sur les artefacts sources (artefacts de sortie créés à la première étape d'un pipeline) utilisés lors de l'exécution d'un pipeline. Ces détails incluent les identifiants comme les identifiants de validation, les commentaires, l'ancienneté des artéfacts et leurs mises à jour et, si vous utilisez l'interface de ligne de commande, les numéros de version des actions de génération. Pour certains types de révision, vous pouvez afficher et ouvrir l'URL de la validation de la version de l'artéfact. Les révisions des sources se composent des éléments suivants :

- Résumé : informations récapitulatives sur la révision la plus récente de l'artefact. Pour GitHub et les AWS CodeCommit référentiels, le message de validation. Pour les compartiments ou les actions Amazon S3, le contenu fourni par l'utilisateur d'une clé récapitulative de révision des artefacts du codepipeline indiquée dans les métadonnées de l'objet.
- revisionUrl : ID de validation pour la révision de l'artefact. Pour les artefacts stockés dans GitHub nos AWS CodeCommit référentiels, l'ID de validation est lié à une page de détails de validation.

Vous pouvez exécuter la commande get-pipeline-execution pour afficher des informations sur les plus récentes révisions source qui ont été incluses dans une exécution du pipeline. Une fois que vous avez exécuté la commande get-pipeline-state pour obtenir des détails sur toutes les étapes d'un pipeline, vous trouvez l'ID d'exécution qui s'applique à une étape particulière dont vous souhaitez connaître les détails d'une révision source. Utilisez ensuite l'ID d'exécution dans la commande get-pipeline-execution. (Étant donné que les étapes d'un pipeline peuvent avoir été correctement exécutées lors d'exécutions précédentes du pipeline, elles peuvent avoir des ID d'exécution différents.)

En d'autres termes, si vous souhaitez afficher les informations sur les artefacts qui sont actuellement à l'étape Intermédiaire, exécutez la commande get-pipeline-state, trouvez l'ID d'exécution actuel de l'étape Intermédiaire et exécutez la commande get-pipeline-execution à l'aide de cet ID d'exécution.

Pour afficher l'état et les révisions des sources dans un pipeline

1. Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et utilisez le AWS CLI pour exécuter la get-pipeline-state commande. Pour un pipeline nommé *MyFirstPipeline*, saisissez :

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

   Cette commande affiche l'état le plus récent d'un pipeline, y compris l'identifiant d'exécution du pipeline le plus récent pour chaque étape.

2. Pour afficher les détails relatifs à une exécution du pipeline, exécutez la commande get-pipeline-execution, indiquez le nom unique et l'ID d'exécution du pipeline de l'exécution dont vous souhaitez afficher les informations relatives à l'artefact. Par exemple, pour afficher les informations relatives à l'exécution d'un pipeline nommé *MyFirstPipeline* avec l'ID d'exécution 3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE, saisissez ce qui suit :

   ```
   aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
   execution-id 3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE
   ```

   Cette commande affiche des informations sur chaque révision source faisant partie de l'exécution du pipeline, ainsi que des informations d'identification sur le pipeline. Seules les informations sur les étapes du pipeline incluses dans cette exécution s'affichent. D'autres étapes du pipeline peuvent ne pas avoir été incluses dans cette exécution de ce pipeline.

   L'exemple suivant montre les données renvoyées pour une partie du pipeline nommée*MyFirstPipeline*, où un artefact nommé « MyApp » est stocké dans un GitHub référentiel :

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"
         }
    }
}
```

Afficher les exécutions d'actions avec list-action-execution (CLI)

Vous pouvez afficher les détails de l'exécution des actions d'un pipeline, comme les ID des exécutions des actions, les artefacts d'entrée, ceux de sortie, le résultat de l'exécution et le statut. Fournissez le filtre de l'ID d'exécution pour renvoyer une liste d'actions dans une exécution de pipeline :

> **Note**
>
> L'historique détaillé des exécutions est disponible pour les exécutions effectuées le 21 février 2019 ou après cette date.

- Pour afficher les exécutions d'action d'un pipeline, procédez de l'une des façons suivantes :
  - Pour afficher les détails de toutes les exécutions d'actions dans un pipeline, exécutez la commande list-action-executions, en indiquant le nom unique du pipeline. Par exemple, pour afficher les exécutions d'actions dans un pipeline nommé *MyFirstPipeline*, saisissez ce qui suit :

    ```
    aws codepipeline list-action-executions --pipeline-name MyFirstPipeline
    ```

    L'exemple suivant illustre une partie des exemples de sortie pour cette commande :

```
{
    "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"
                }
            }
        },
```

```
. . .
```

- Pour afficher toutes les exécutions d'actions dans une exécution de pipeline, exécutez la commande list-action-executions, en indiquant le nom unique du pipeline et l'ID d'exécution. Par exemple, pour afficher les exécutions d'actions pour un*Execution_ID*, entrez les informations suivantes :

```
aws codepipeline list-action-executions --pipeline-name MyFirstPipeline --filter
 pipelineExecutionId=Execution_ID
```

- L'exemple suivant illustre une partie des exemples de sortie pour cette commande :

```
{
    "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"
                    }
                ]
            },
```

```
. . .
```

Définir ou modifier le mode d'exécution du pipeline

Vous pouvez définir le mode d'exécution de votre pipeline afin de spécifier la manière dont les exécutions multiples sont gérées.

Pour plus d'informations sur les modes d'exécution du pipeline, consultezFonctionnement des exécutions de pipeline.

> **Important**
>
> Pour les pipelines en mode PARALLÈLE, lorsque vous modifiez le mode d'exécution du pipeline sur QUEUED ou SUPERSEDED, l'état du pipeline n'affiche pas l'état mis à jour en tant que PARALLÈLE. Pour de plus amples informations, veuillez consulter Les pipelines passés du mode PARALLÈLE afficheront un mode d'exécution précédent.

> **Important**
>
> Pour les pipelines en mode PARALLÈLE, lorsque vous modifiez le mode d'exécution du pipeline sur QUEUED ou SUPERSEDED, la définition du pipeline pour chaque mode ne sera pas mise à jour. Pour de plus amples informations, veuillez consulter Les pipelines en mode PARALLÈLE ont une définition de pipeline obsolète s'ils sont modifiés lors du passage en mode QUEUED ou SUPERSEDED.

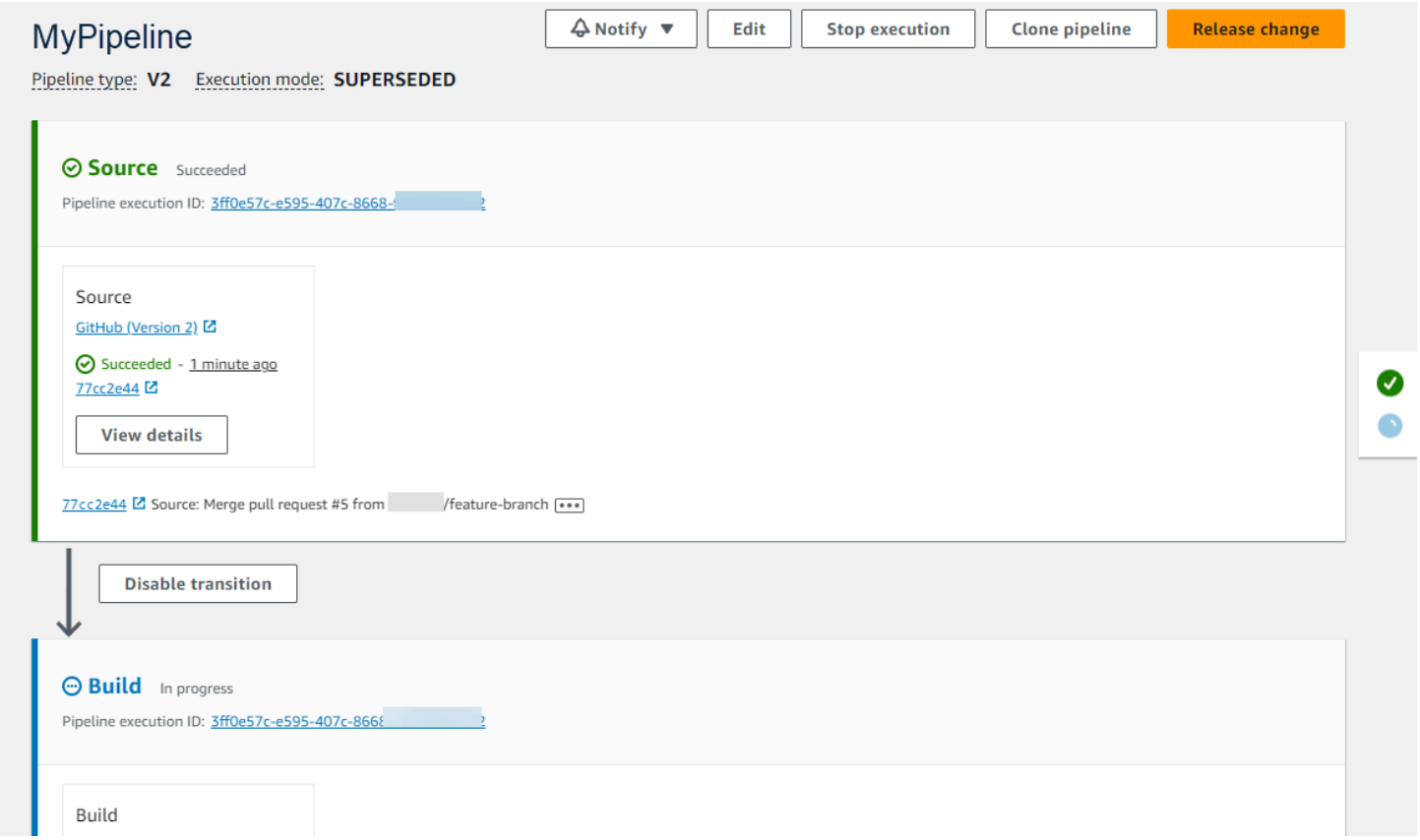
> **Important**
>
> Pour les pipelines en mode PARALLÈLE, la restauration des étapes n'est pas disponible. De même, les conditions de défaillance avec un type de résultat d'annulation ne peuvent pas être ajoutées à un pipeline en mode PARALLÈLE.

Considérations relatives à l'affichage des modes d'exécution

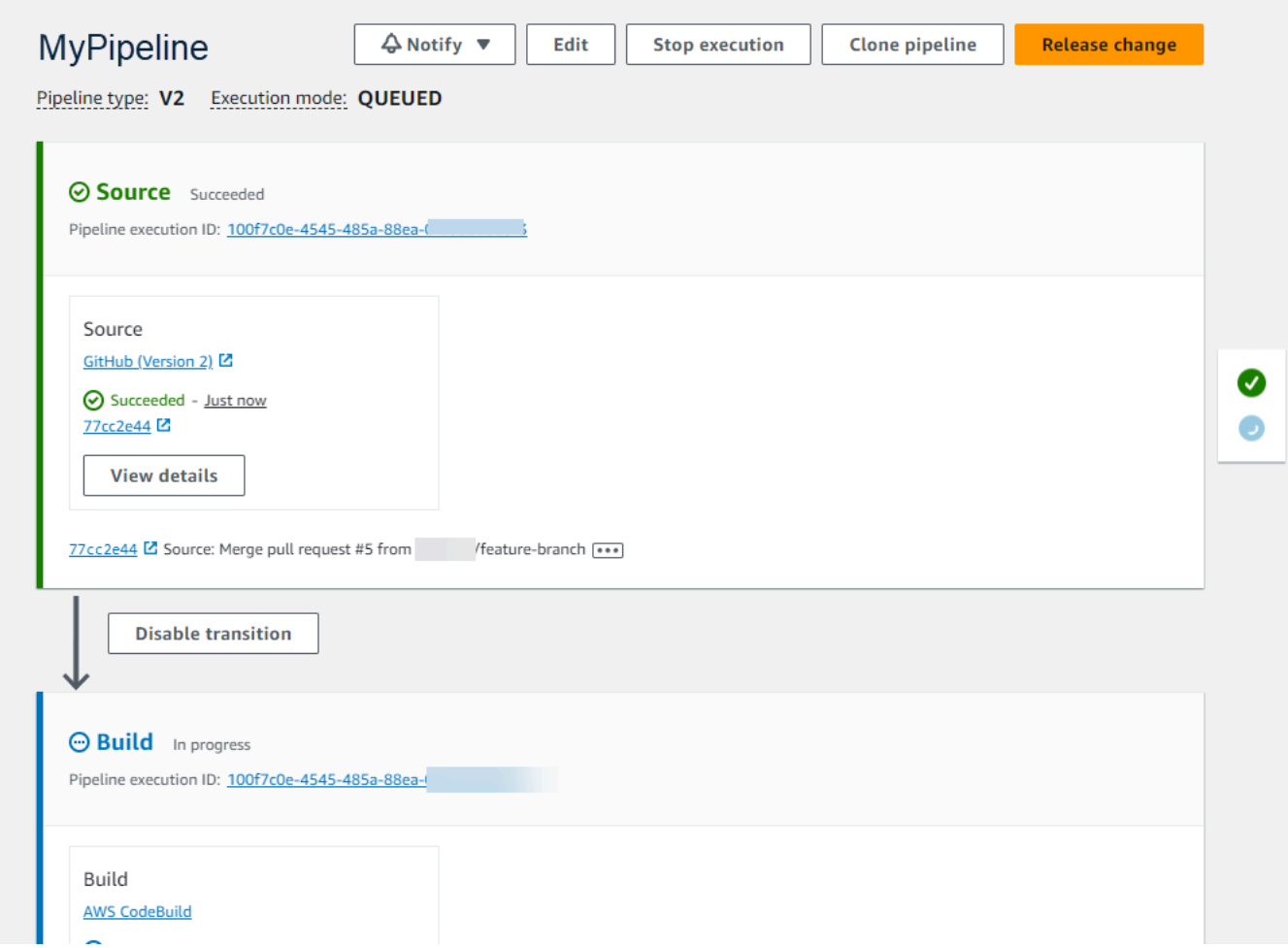
L'affichage des pipelines dans des modes d'exécution spécifiques doit être pris en compte.

Pour les modes SUPERSEDED et QUEUED, utilisez la vue du pipeline pour voir les exécutions en cours, puis cliquez sur l'ID d'exécution pour afficher les détails et l'historique. Pour le mode PARALLÈLE, cliquez sur l'ID d'exécution pour afficher l'exécution en cours dans l'onglet Visualisation.

Ce qui suit montre l'affichage du mode SUPERSEDED dans. CodePipeline

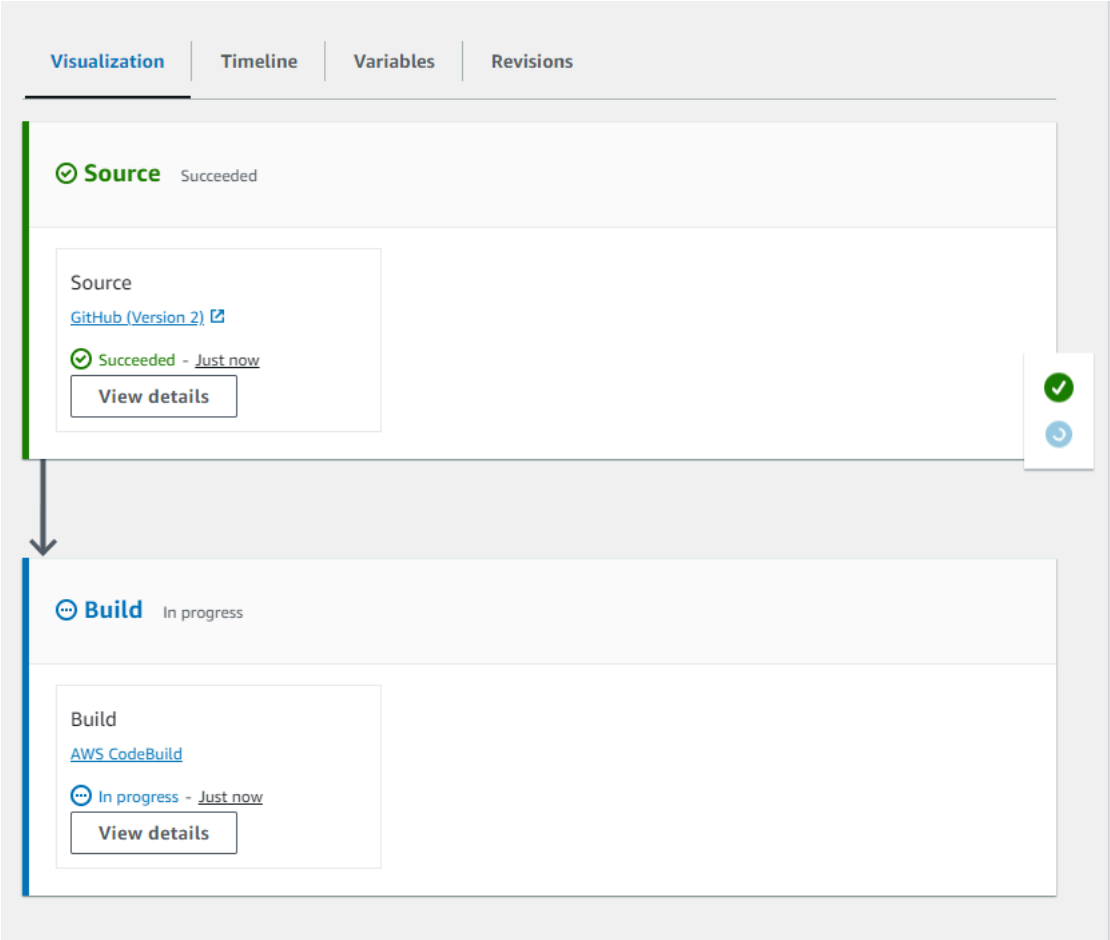


Ce qui suit montre l'affichage du mode QUEUED dans. CodePipeline



Ce qui suit montre l'affichage du mode PARALLÈLE dans CodePipeline.

> **Important**
>
> Pour les pipelines en mode PARALLÈLE, la restauration des étapes n'est pas disponible. De même, les conditions de défaillance avec un type de résultat d'annulation ne peuvent pas être ajoutées à un pipeline en mode PARALLÈLE.



Considérations relatives au passage d'un mode d'exécution à un autre

Les considérations suivantes concernent les pipelines lors de la modification du mode du pipeline. Lorsque vous passez d'un mode d'exécution à un autre en mode Édition, puis que vous enregistrez la modification, certaines vues ou certains états peuvent être ajustés.

Par exemple, lorsque vous passez du mode PARALLÈLE à un mode QUEUED ou SUPERSEDED, l'exécution démarrée en mode PARALLÈLE continuera de s'exécuter. Vous pouvez les consulter sur la page de l'historique des exécutions. La vue du pipeline indiquera l'exécution qui s'est exécutée en mode QUEUED ou SUPERSEDED plus tôt ou un état vide dans le cas contraire.

Autre exemple : lorsque vous passez du mode QUEUED ou SUPERSEDED au mode PARALLÈLE, vous ne verrez plus la page du pipeline. view/state Pour visualiser une exécution en mode PARALLÈLE, utilisez l'onglet Visualisation sur la page des détails de l'exécution. Les exécutions démarrées en mode SUPERSEDED ou QUEUED seront annulées.

Le tableau suivant fournit plus de détails.

Changement de mode	Détails de l'exécution active et en attente	Informations sur l'état du pipeline
REMPLACÉ par REMPLACÉ/ REMPLACÉ par EN FILE D'ATTENTE	• Les exécutions actives sont annulées une fois les actions en cours terminées. • Les exécutions en attente sont annulées.	L'état du pipeline, tel que annulé, est conservé entre la version du premier mode et celle du second mode.
DE LA FILE D'ATTENTE À LA FILE D'ATTENTE/DE LA FILE D'ATTENTE À REMPLACÉ	• Les exécutions actives sont annulées une fois les actions en cours terminées. • Les exécutions en attente sont annulées.	L'état du pipeline, tel que annulé, est conservé entre la version du premier mode et celle du second mode.
PARALLÈLE à PARALLÈLE	Toutes les exécutions sont autorisées à s'exécuter indépendamment des mises à jour des définitions de pipeline.	Vide. Le mode parallèle n'a pas d'état de pipeline.
REMPLACÉ en PARALLÈLE /EN FILE D'ATTENTE en PARALLÈLE	• Les exécutions actives sont annulées une fois les actions en cours terminées. • Les exécutions en attente sont annulées.	Vide. Le mode parallèle n'a pas d'état de pipeline.

Définir ou modifier le mode d'exécution du pipeline (console)

Vous pouvez utiliser la console pour définir le mode d'exécution du pipeline.

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms et l'état de tous les pipelines associés à votre AWS compte sont affichés.

2. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier.

3. Sur la page des détails du pipeline, choisissez Edit.
4. Sur la page Modifier, choisissez Modifier : propriétés du pipeline.
5. Choisissez le mode pour votre pipeline.
   - Remplacé
   - En file d'attente (pipeline de type V2 requis)
   - Parallèle (pipeline de type V2 requis)
6. Sur la page Modifier, choisissez OK.

Définissez le mode d'exécution du pipeline (CLI)

AWS CLI Pour définir le mode d'exécution du pipeline, utilisez la `update-pipeline` commande `create-pipeline` ou.

1. Ouvrez une session de terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et exécutez la get-pipeline commande pour copier la structure du pipeline dans un fichier JSON. Par exemple, pour un pipeline nommé **`MyFirstPipeline`**, saisissez la commande suivante :

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Cette commande ne renvoie rien, mais le fichier que vous avez créé doit apparaître dans le répertoire où vous avez exécuté la commande.
2. Ouvrez le fichier JSON dans n'importe quel éditeur de texte brut et modifiez la structure du fichier pour refléter le mode d'exécution du pipeline que vous souhaitez définir, tel que QUEUED.

   ```
   "executionMode": "QUEUED"
   ```

   L'exemple suivant montre comment définir le mode d'exécution sur QUEUED dans un exemple de pipeline en deux étapes.

   ```
   {
       "pipeline": {
           "name": "MyPipeline",
           "roleArn": "arn:aws:iam::111122223333:role/service-role/
   AWSCodePipelineServiceRole-us-east-1-dkpippe",
           "artifactStore": {
               "type": "S3",
   ```

```
            "location": "bucket"
        },
        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "provider": "CodeCommit",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "BranchName": "main",
                            "OutputArtifactFormat": "CODE_ZIP",
                            "PollForSourceChanges": "true",
                            "RepositoryName": "MyDemoRepo"
                        },
                        "outputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "inputArtifacts": [],
                        "region": "us-east-1",
                        "namespace": "SourceVariables"
                    }
                ]
            },
            {
                "name": "Build",
                "actions": [
                    {
                        "name": "Build",
                        "actionTypeId": {
                            "category": "Build",
                            "owner": "AWS",
                            "provider": "CodeBuild",
                            "version": "1"
                        },
                        "runOrder": 1,
```

```
                        "configuration": {
                            "ProjectName": "MyBuildProject"
                        },
                        "outputArtifacts": [
                            {
                                "name": "BuildArtifact"
                            }
                        ],
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "region": "us-east-1",
                        "namespace": "BuildVariables"
                    }
                ]
            }
        ],
        "version": 1,
        "executionMode": "QUEUED"
    }
}
```

3. Si vous utilisez la structure de pipeline extraite à l'aide de la commande get-pipeline, vous devez modifier la structure dans le fichier JSON. Vous devez supprimer les lignes `metadata` du fichier afin que la commande update-pipeline puisse l'utiliser. Supprimez la section de la structure de pipeline dans le fichier JSON (les lignes `"metadata": { }` et les champs `"created"`, `"pipelineARN"` et `"updated"`).

   Par exemple, supprimez les lignes suivantes de la structure :

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

   Enregistrez le fichier.

4. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline :

> **Important**
>
> N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

Cette commande affiche toute la structure du pipeline mise à jour.

> **Note**
>
> La commande update-pipeline interrompt le pipeline. Si une révision est exécutée dans le pipeline lorsque vous exécutez la commande update-pipeline celle-ci est interrompue. Vous devez lancer manuellement le pipeline pour exécuter cette révision dans le pipeline mis à jour.

Annuler ou réessayer les étapes

Vous pouvez utiliser la AWS CodePipeline console ou le AWS CLI pour revenir en arrière ou réessayer manuellement une étape ou les actions d'une étape. Pour configurer les conditions des étapes dans lesquelles vous souhaitez utiliser Rollback ou Retry comme résultat configuré, voir. Configuration des conditions pour une étape

Rubriques

- Configuration d'une nouvelle tentative d'étape pour une étape ayant échoué ou des actions ayant échoué
- Configuration de la restauration des étapes

Configuration d'une nouvelle tentative d'étape pour une étape ayant échoué ou des actions ayant échoué

Vous pouvez réessayer une étape qui a échoué sans avoir à réexécuter un pipeline depuis le début. Pour ce faire, vous pouvez essayer de nouveau les actions qui ont échoué dans une étape ou en réessayant toutes les actions de l'étape en commençant par la première action de la phase. Lorsque vous réessayez les actions qui ont échoué au cours d'une étape, toutes les actions encore en cours continuent de fonctionner et les actions qui ont échoué sont à nouveau déclenchées. Lorsque vous réessayez une étape qui a échoué à partir de la première action de la phase, aucune action ne peut être en cours pour cette étape. Avant qu'une étape puisse être réessayée, toutes les actions doivent avoir échoué ou certaines actions ont échoué et d'autres ont réussi.

> **Important**
>
> Réessayer une étape ayant échoué recommence toutes les actions de la phase depuis la première action de la phase, et réessayer les actions qui ont échoué permet de réessayer toutes les actions qui ont échoué de la phase. Cela remplace les artefacts de sortie des actions précédemment réussies lors de la même exécution.
> Bien que les artefacts puissent être remplacés, l'historique des exécutions des actions précédemment réussies est toujours conservé.

Si vous utilisez la console pour visualiser un pipeline, un bouton de phase Réessayer ou un bouton Réessayer les actions ayant échoué apparaît sur l'étape qui peut être réessayée.

Si vous utilisez l' AWS interface de ligne de commande, vous pouvez utiliser la get-pipeline-state commande pour déterminer si des actions ont échoué.

> **Note**
>
> Dans les cas suivants, il se peut que vous ne puissiez pas réessayer une étape :
>
> - Toutes les actions de l'étape ont réussi et la phase n'est donc pas en état d'échec.
> - La structure globale du pipeline a changé après la défaillance de l'étape.
> - Une autre tentative est déjà en cours dans l'étape.

Rubriques

- Considérations relatives à une nouvelle tentative
- Réessayer manuellement une étape ayant échoué
- Configurer une étape pour une nouvelle tentative automatique en cas d'échec

Considérations relatives à une nouvelle tentative

Les considérations relatives à une nouvelle tentative d'étape sont les suivantes :

- Vous ne pouvez configurer une nouvelle tentative automatique en cas d'échec de l'étape que pour une seule nouvelle tentative.
- Vous pouvez configurer une nouvelle tentative automatique en cas d'échec de l'étape pour toutes les actions, y compris les `Source` actions.

Réessayer manuellement une étape ayant échoué

Vous pouvez réessayer manuellement une étape ayant échoué à l'aide de la console ou de l'interface de ligne de commande.

Vous pouvez également configurer une étape pour une nouvelle tentative automatique en cas d'échec, comme indiqué dansConfigurer une étape pour une nouvelle tentative automatique en cas d'échec.

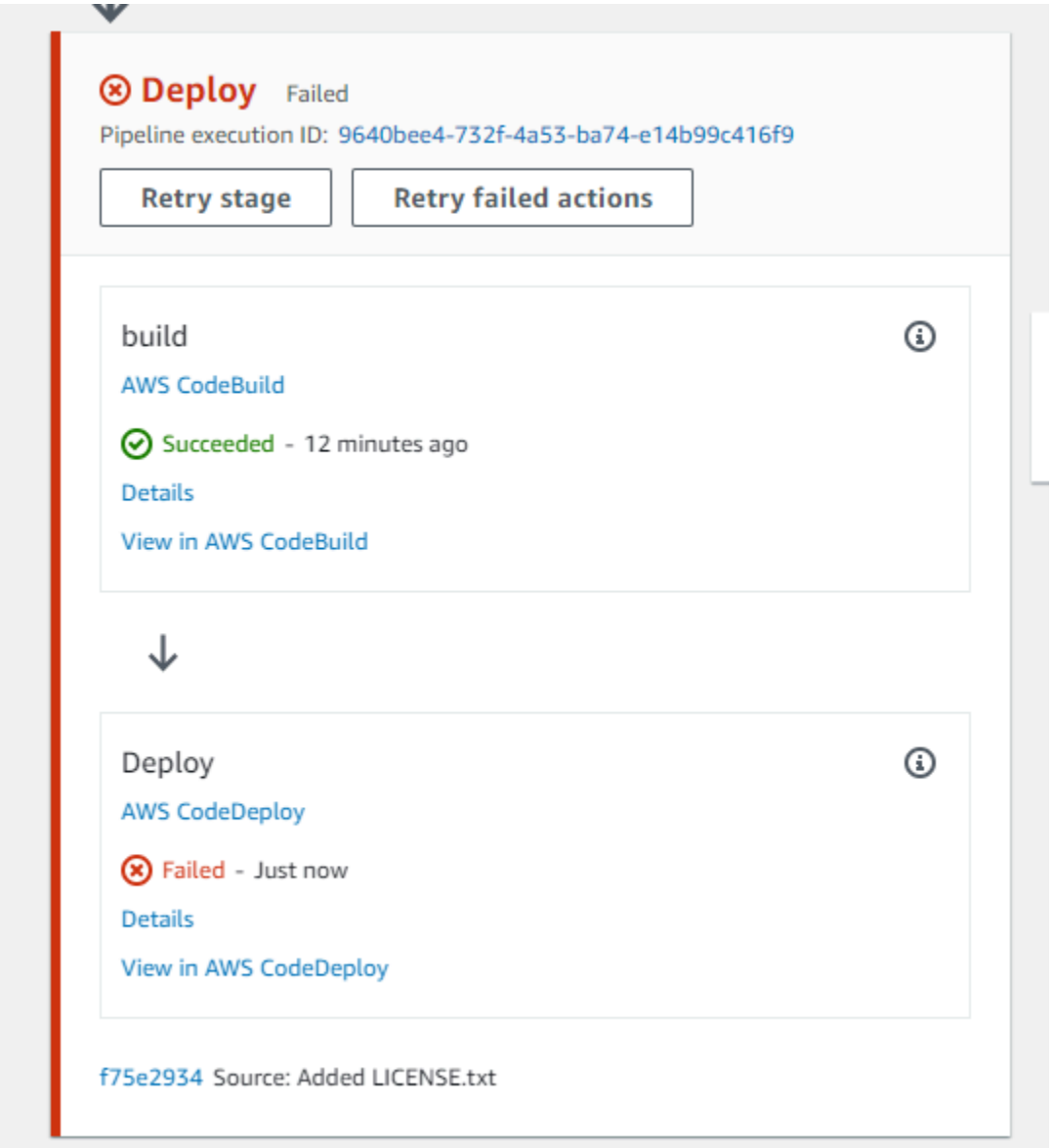
Réessayer manuellement une étape ayant échoué (console)

Pour réessayer une étape ayant échoué ou des actions ayant échoué dans une étape, console

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés.
2. Dans Name, choisissez le nom du pipeline.
3. Localisez l'étape sur laquelle l'action a échoué, puis choisissez l'une des options suivantes :

   - Pour réessayer toutes les actions de la phase, choisissez Réessayer la phase.
   - Pour ne réessayer que les actions ayant échoué au cours de l'étape, choisissez Réessayer les actions ayant échoué.



Si toutes les actions retentées dans l'étape aboutissent avec succès, le pipeline continue de s'exécuter.

Réessayer manuellement une étape ayant échoué (CLI)

Pour réessayer une étape ayant échoué ou des actions ayant échoué dans une étape - CLI

Pour AWS CLI réessayer toutes les actions ou toutes les actions ayant échoué, exécutez la retry-stage-execution commande avec les paramètres suivants :

```
--pipeline-name <value>
--stage-name <value>
--pipeline-execution-id <value>
--retry-mode ALL_ACTIONS/FAILED_ACTIONS
```

> **Note**
>
> Les valeurs que vous pouvez utiliser pour `retry-mode` sont `FAILED_ACTIONS` et`ALL_ACTIONS`.

1. Sur un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows), exécutez la retry-stage-execution commande, comme indiqué dans l'exemple suivant pour un pipeline nommé`MyPipeline`.

   ```
   aws codepipeline retry-stage-execution --pipeline-name MyPipeline --stage-name
    Deploy --pipeline-execution-id b59babff-5f34-EXAMPLE --retry-mode FAILED_ACTIONS
   ```

   La sortie renvoie l'ID d'exécution :

   ```
   {
       "pipelineExecutionId": "b59babff-5f34-EXAMPLE"
   }
   ```

2. Vous pouvez également exécuter la commande avec un fichier d'entrée JSON. Vous devez d'abord créer un fichier JSON identifiant le pipeline, l'étape contenant les actions ayant échoué et la dernière exécution du pipeline dans cette étape. Exécutez la commande retry-stage-execution avec le paramètre `--cli-input-json`. Pour récupérer les informations dont vous avez besoin pour le fichier JSON, le plus simple est d'utiliser la commande get-pipeline-state.

   a. Sur un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows), exécutez la get-pipeline-state commande sur un pipeline. Par exemple, pour un pipeline nommé MyFirstPipeline, vous devez taper quelque chose de similaire à ce qui suit :

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

La réponse à la commande inclut des informations sur l'état du pipeline pour chaque étape. Dans l'exemple suivant, la réponse indique qu'une ou plusieurs actions a/ont échoué lors de l'étape Intermédiaire :

```
{
    "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. Dans un éditeur de texte brut, créez un fichier où vous enregistrerez ce qui suit, au format JSON :

- Le nom du pipeline qui contient les actions ayant échoué
- Le nom de l'étape qui contient les actions ayant échoué
- L'identifiant de la dernière exécution du pipeline dans l'étape
- Le mode de nouvelle tentative.

Dans l' MyFirstPipeline exemple précédent, votre fichier ressemblerait à ceci :

```
{
```

```
    "pipelineName": "MyFirstPipeline",
    "stageName": "Staging",
    "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
    "retryMode": "FAILED_ACTIONS"
}
```

c. Enregistrez le fichier avec un nom tel que **`retry-failed-actions.json`**.

d. Utilisez le fichier que vous avez créé lorsque vous avez exécuté la commande retry-stage-execution. Par exemple :

> **Important**
>
> N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

```
aws codepipeline retry-stage-execution --cli-input-json file://retry-failed-actions.json
```

e. Pour afficher les résultats de la nouvelle tentative, ouvrez la CodePipeline console et choisissez le pipeline contenant les actions qui ont échoué, ou utilisez à nouveau la get-pipeline-state commande. Pour de plus amples informations, veuillez consulter Afficher les pipelines et les détails dans CodePipeline.

Configurer une étape pour une nouvelle tentative automatique en cas d'échec

Vous pouvez configurer une étape pour une nouvelle tentative automatique en cas d'échec. L'étape effectuera une nouvelle tentative et affichera l'état de la nouvelle tentative pour l'étape ayant échoué sur la page Afficher le pipeline.

Vous pouvez configurer le mode de nouvelle tentative en spécifiant que l'étape doit automatiquement réessayer soit toutes les actions de la phase ayant échoué, soit uniquement les actions ayant échoué de la phase.

Configurer une étape pour une nouvelle tentative automatique en cas d'échec (console)

Vous pouvez utiliser la console pour configurer une étape pour une nouvelle tentative automatique.

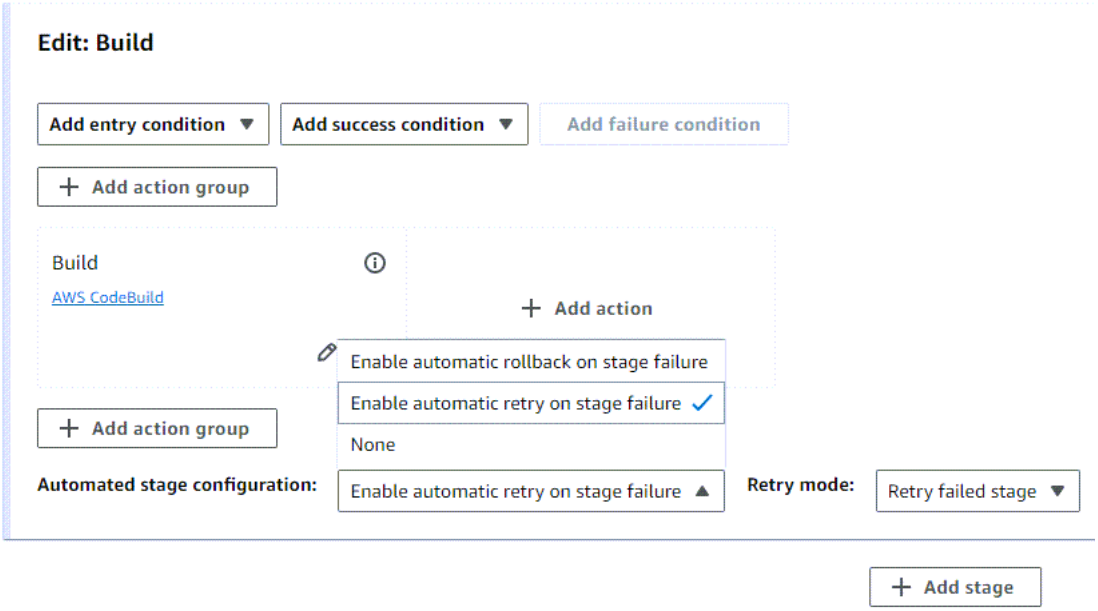
Configurer une étape pour une nouvelle tentative automatique (console)

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

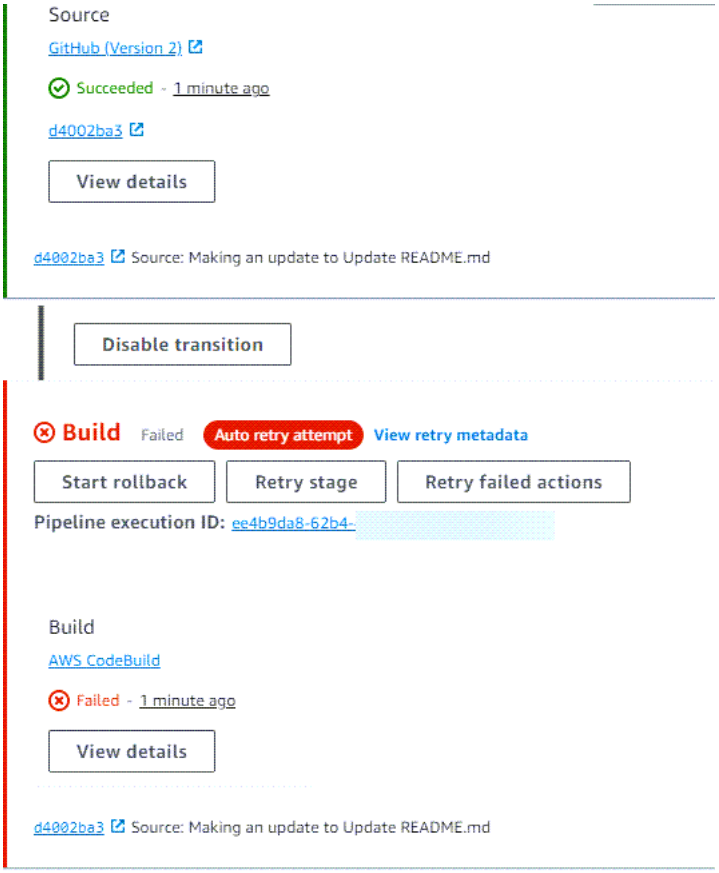
   Les noms et l'état de tous les pipelines associés à votre AWS compte sont affichés.
2. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Sur la page Modifier, pour l'action que vous souhaitez modifier, choisissez Modifier l'étape.
5. Choisissez Configuration automatique de l'étape :, puis choisissez Activer la nouvelle tentative automatique en cas d'échec de la phase. Enregistrez les modifications apportées à votre pipeline.
6. Dans Configuration automatique de l'étape :, choisissez l'un des modes de nouvelle tentative suivants :

   - Pour spécifier que le mode réessaiera toutes les actions de la phase, choisissez Retry failed stage.
   - Pour spécifier que le mode ne réessaiera que les actions ayant échoué au cours de l'étape, choisissez Réessayer les actions ayant échoué.

   Enregistrez les modifications apportées à votre pipeline.



7. Une fois votre pipeline exécuté, si l'étape échoue, une nouvelle tentative automatique sera effectuée. Les exemples suivants montrent une étape de construction qui a été réessayée automatiquement.



8. Pour afficher les détails de la nouvelle tentative, choisissez. La fenêtre s'affiche.



Configurer une étape pour une nouvelle tentative automatique (CLI)

Pour configurer une étape AWS CLI afin de réessayer automatiquement en cas d'échec, utilisez les commandes pour créer ou mettre à jour un pipeline comme indiqué dans Création d'un pipeline, d'étapes et d'actions etModifier un pipeline dans CodePipeline.

- Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et exécutez la AWS CLI `update-pipeline` commande en spécifiant la condition de défaillance dans la structure du pipeline. L'exemple suivant configure une nouvelle tentative automatique pour une étape nommée : `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",
                    },
```

```
            }
```

Configurez une étape pour une nouvelle tentative automatique (CloudFormation)

Pour configurer une étape pour une nouvelle tentative automatique en cas d'échec, utilisez le paramètre de cycle de vie de la `OnFailure` phase. CloudFormation Utilisez le `RetryConfiguration` paramètre pour configurer le mode de nouvelle tentative.

```
OnFailure:
     Result: RETRY
     RetryConfiguration:
         RetryMode: ALL_ACTIONS
```

- Mettez à jour le modèle comme indiqué dans l'extrait suivant. L'exemple suivant configure une nouvelle tentative automatique pour une étape nommée : `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'
```

Pour plus d'informations sur la configuration de la nouvelle tentative d'étape en cas d'échec, consultez OnFailure le Guide `StageDeclaration` de l'CloudFormation utilisateur ci-dessous.

Configuration de la restauration des étapes

Vous pouvez revenir d'une étape à une exécution qui s'est déroulée avec succès à cette étape. Vous pouvez préconfigurer un stage pour l'annuler en cas d'échec, ou vous pouvez annuler manuellement un stage. L'opération annulée entraînera une nouvelle exécution. L'exécution du pipeline cible choisie pour la restauration est utilisée pour récupérer les révisions et les variables de la source.

Le type d'exécution, standard ou rétroactif, s'affiche dans l'historique du pipeline, l'état du pipeline et les détails de son exécution.

Rubriques

- Considérations relatives aux annulations
- Faire reculer une étape manuellement
- Configurer une étape pour la restauration automatique
- Afficher l'état de restauration dans la liste des exécutions
- Afficher les détails de l'état de restauration

Considérations relatives aux annulations

Les considérations relatives à la restauration des étapes sont les suivantes :

- Vous ne pouvez pas annuler un stage source.
- Le pipeline ne peut revenir à une exécution précédente que si l'exécution précédente a été démarrée dans la version actuelle de la structure du pipeline.
- Vous ne pouvez pas revenir à un ID d'exécution cible qui est un type d'exécution d'annulation.
- CodePipeline utilisera les variables et les artefacts de l'exécution vers laquelle il revient.

Faire reculer une étape manuellement

Vous pouvez annuler manuellement une étape à l'aide de la console ou de l'interface de ligne de commande. Le pipeline ne peut revenir à une exécution précédente que si l'exécution précédente a été démarrée dans la version actuelle de la structure du pipeline.

Vous pouvez également configurer une étape pour qu'elle revienne automatiquement en cas d'échec, comme indiqué dansConfigurer une étape pour la restauration automatique.

Faire reculer une étape manuellement (console)

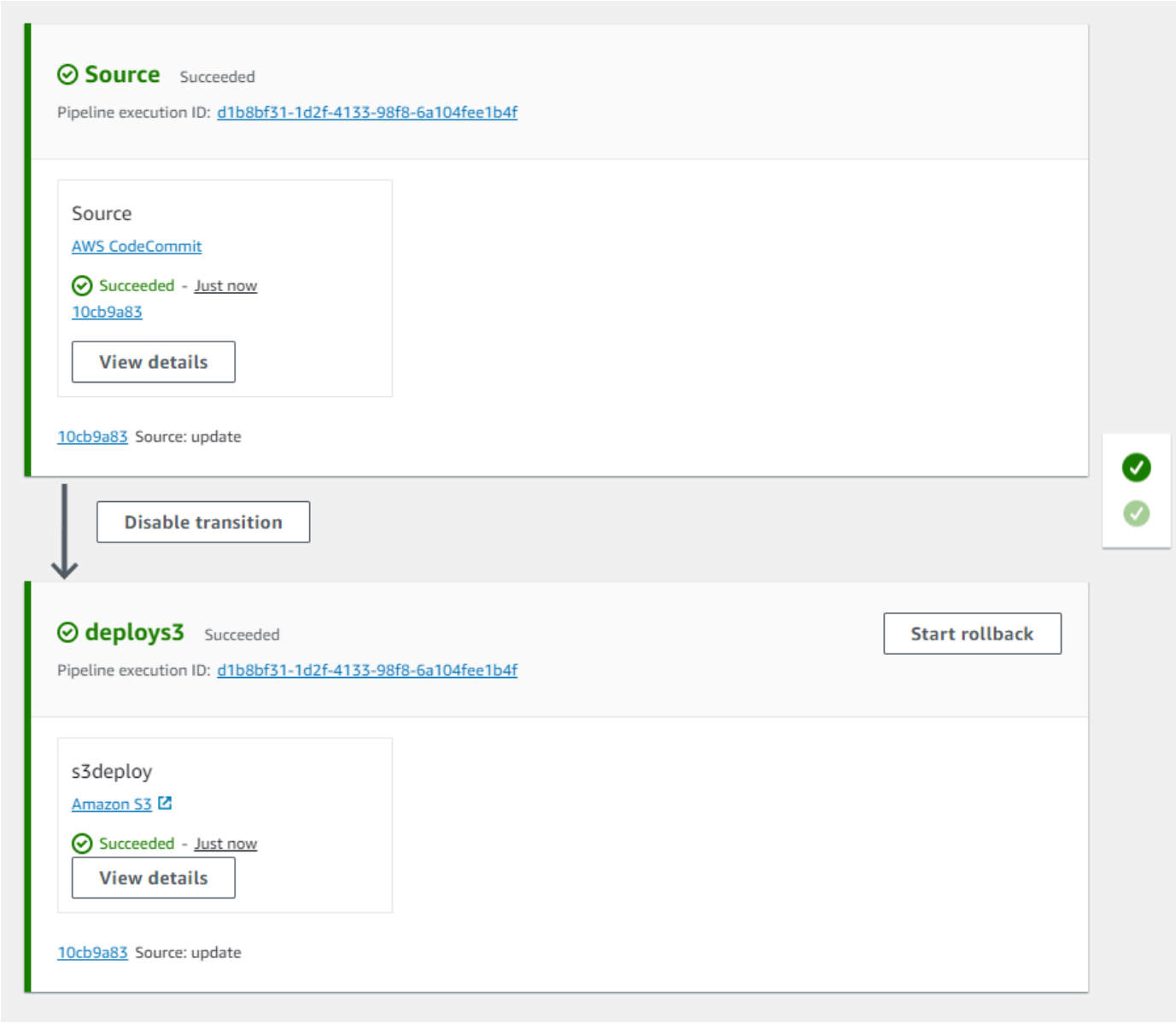
Vous pouvez utiliser la console pour ramener manuellement une étape à l'exécution d'un pipeline cible. Lorsqu'une étape est annulée, une étiquette de restauration s'affiche sur la visualisation du pipeline dans la console.

Faire reculer une étape manuellement (console)

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

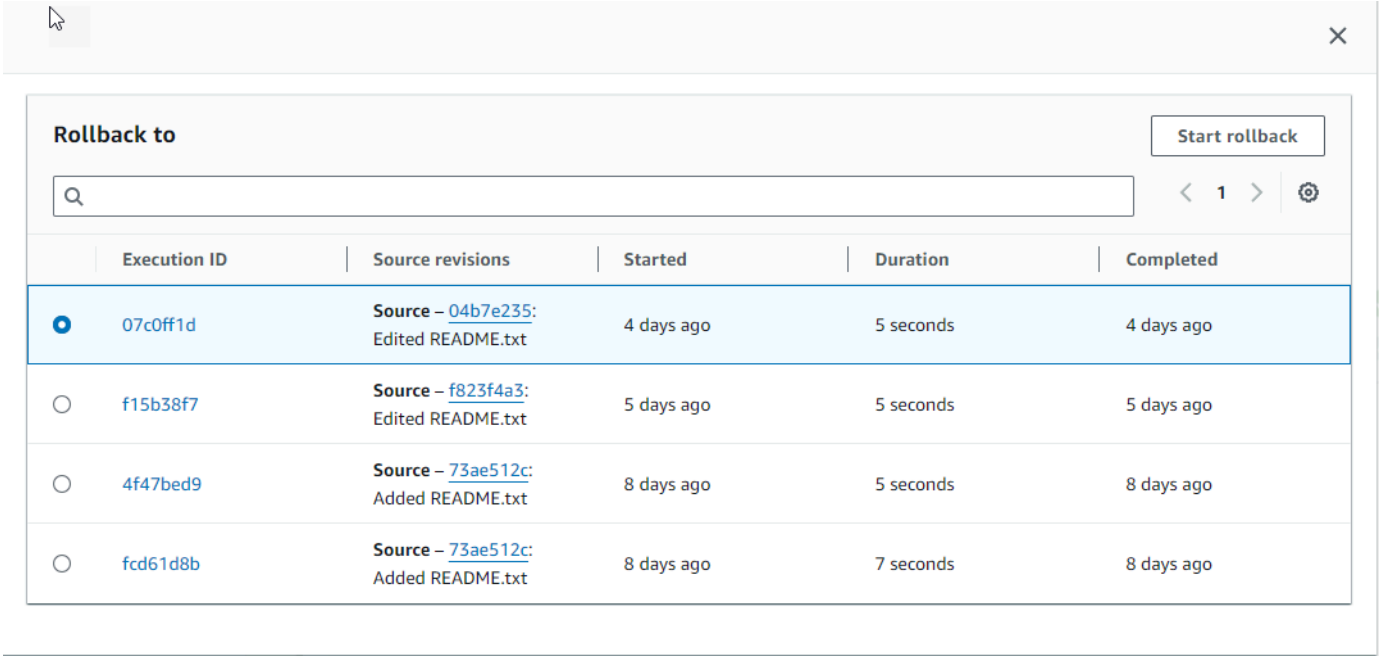
   Les noms et l'état de tous les pipelines associés à votre AWS compte sont affichés.

2. Dans Nom, choisissez le nom du pipeline contenant l'étape à annuler.

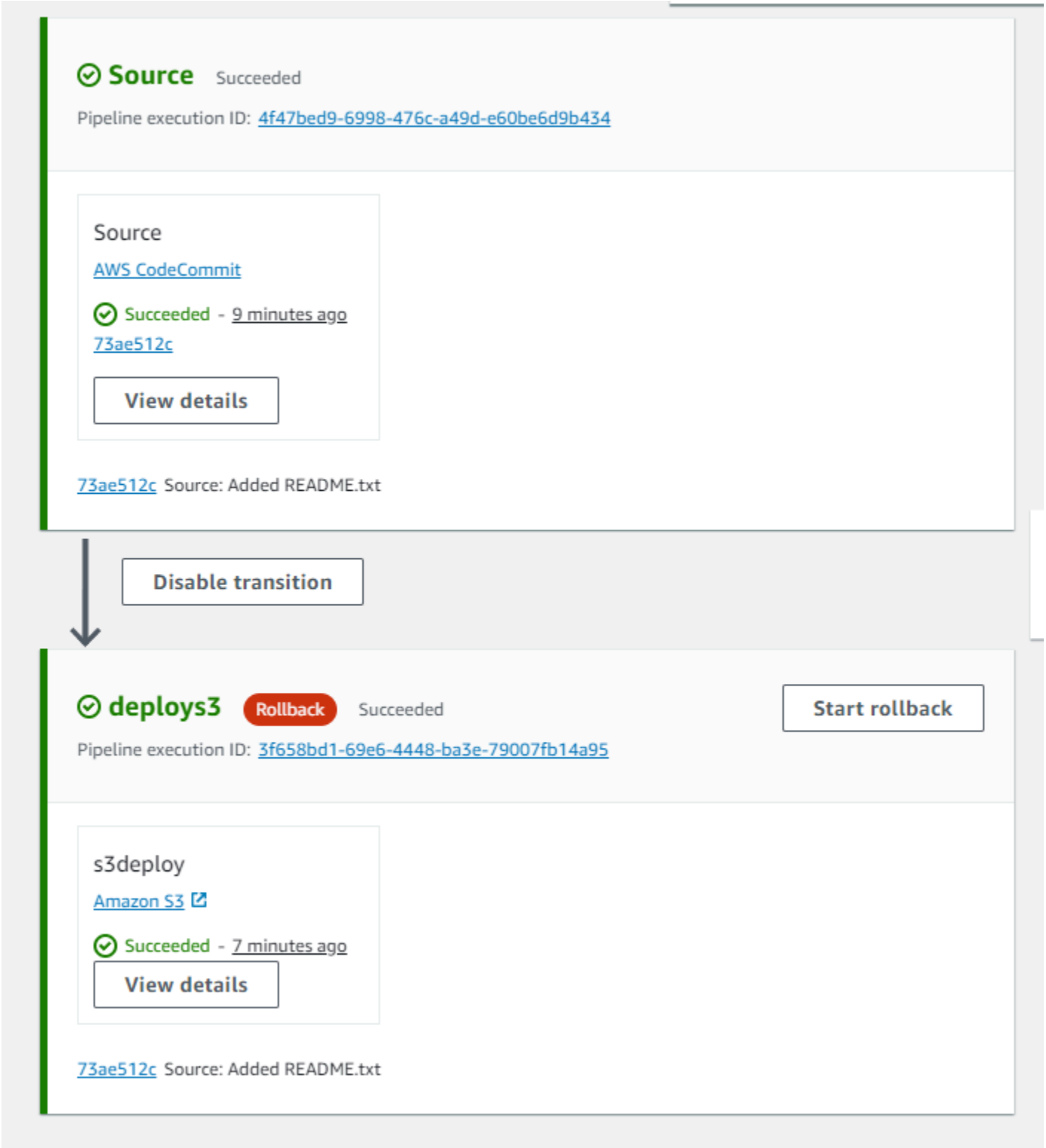


3. Sur la scène, choisissez Démarrer la restauration. La page Revenir à la page s'affiche.
4. Choisissez l'exécution cible vers laquelle vous souhaitez revenir en arrière.

> **Note**
>
> La liste des exécutions de pipeline cibles disponibles représentera toutes les exécutions dans la version actuelle du pipeline à compter du 1er février 2024.



Le schéma suivant montre un exemple de stage annulé avec le nouvel ID d'exécution.



Faire reculer une étape manuellement (CLI)

Pour utiliser le AWS CLI pour annuler manuellement une étape, utilisez la `rollback-stage` commande.

Vous pouvez également annuler une étape manuellement, comme indiqué dansFaire reculer une étape manuellement.

> **Note**
>
> La liste des exécutions de pipeline cibles disponibles représentera toutes les exécutions dans la version actuelle du pipeline à compter du 1er février 2024.

Pour annuler une étape manuellement (CLI)

1. La commande CLI pour la restauration manuelle nécessitera l'ID d'exécution d'une exécution de pipeline précédemment réussie au cours de l'étape. Pour obtenir l'ID d'exécution du pipeline cible que vous allez spécifier, utilisez la commande list-pipeline-executions avec un filtre qui renverra les exécutions réussies au cours de l'étape. Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et utilisez le AWS CLI pour exécuter la `list-pipeline-executions` commande, en spécifiant le nom du pipeline et le filtre pour les exécutions réussies au cours de la phase. Dans cet exemple, la sortie répertorie les exécutions de pipeline pour le pipeline nommé MyFirstPipeline et les exécutions réussies dans l'étape nomméedeploys3.

   ```
   aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline --filter
    succeededInStage={stageName=deploys3}
   ```

   Dans la sortie, copiez l'ID d'exécution de la précédente exécution réussie que vous souhaitez spécifier pour l'annulation. Vous l'utiliserez à l'étape suivante comme identifiant d'exécution cible.

2. Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et utilisez le AWS CLI pour exécuter la `rollback-stage` commande, en spécifiant le nom du pipeline, le nom de la phase et l'exécution cible vers laquelle vous souhaitez revenir. Par exemple, pour annuler une étape nommée Deploy pour un pipeline nommé *`MyFirstPipeline`* :

   ```
   aws codepipeline rollback-stage --pipeline-name MyFirstPipeline --stage-name Deploy
    --target-pipeline-execution-id bc022580-4193-491b-8923-9728dEXAMPLE
   ```

   La sortie renvoie l'ID d'exécution de la nouvelle exécution annulée. Il s'agit d'un identifiant distinct qui utilise les révisions source et les paramètres de l'exécution cible spécifiée.

Configurer une étape pour la restauration automatique

Vous pouvez configurer les étapes d'un pipeline pour qu'elles soient annulées automatiquement en cas de panne. Lorsque la phase échoue, elle revient à la dernière exécution réussie. Le pipeline ne peut revenir à une exécution précédente que si l'exécution précédente a été démarrée dans la version actuelle de la structure du pipeline. Étant donné que la configuration de restauration automatique fait partie de la définition du pipeline, votre étape de pipeline ne sera automatiquement restaurée qu'une fois l'exécution réussie du pipeline dans la phase de pipeline.

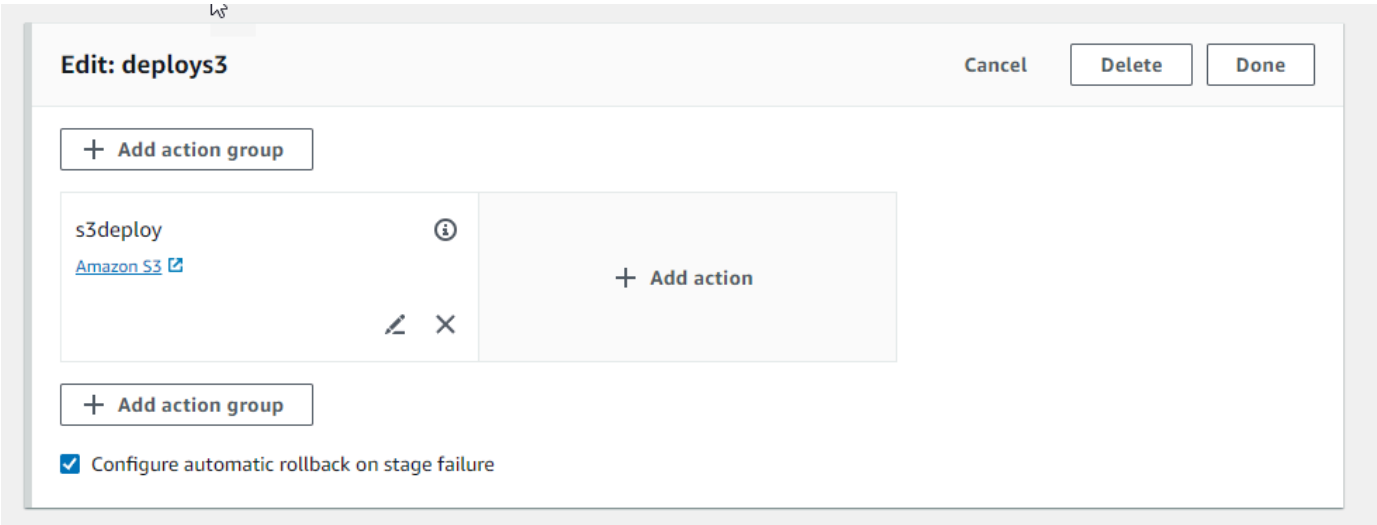
Configurer une étape pour la restauration automatique (console)

Vous pouvez ramener une étape à une exécution réussie précédente spécifiée. Pour plus d'informations, consultez RollbackStage le Guide des CodePipeline API.

Configurer une étape pour la restauration automatique (console)

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms et l'état de tous les pipelines associés à votre AWS compte sont affichés.
2. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Sur la page Modifier, pour l'action que vous souhaitez modifier, choisissez Modifier l'étape.
5. Choisissez Configuration automatique de l'étape :, puis choisissez Configurer la restauration automatique en cas de défaillance de la phase. Enregistrez les modifications apportées à votre pipeline.



Configurer une étape pour la restauration automatique (CLI)

AWS CLI Pour configurer une étape ayant échoué afin de revenir automatiquement à la dernière exécution réussie, utilisez les commandes pour créer ou mettre à jour un pipeline comme indiqué dans Création d'un pipeline, d'étapes et d'actions etModifier un pipeline dans CodePipeline.

- Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et utilisez le AWS CLI pour exécuter la `update-pipeline` commande, en spécifiant la condition de défaillance dans la structure du pipeline. L'exemple suivant configure la restauration automatique pour une étape nommée : `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"
                }
            }
```

Pour plus d'informations sur la configuration des conditions de défaillance pour la restauration des étapes, consultez FailureConditions la référence de l'CodePipeline API.

Configurez une étape pour la restauration automatique (CloudFormation)

À utiliser CloudFormation pour configurer une étape pour qu'elle revienne automatiquement en cas d'échec, utilisez le `OnFailure` paramètre. En cas d'échec, la phase revient automatiquement à la dernière exécution réussie.

```
OnFailure:
     Result: ROLLBACK
```

- Mettez à jour le modèle comme indiqué dans l'extrait suivant. L'exemple suivant configure la restauration automatique pour une étape nommée : `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'
```

Pour plus d'informations sur la configuration des conditions de défaillance pour la restauration des étapes, consultez OnFailure le Guide `StageDeclaration` de l'CloudFormation utilisateur ci-dessous.

Afficher l'état de restauration dans la liste des exécutions

Vous pouvez consulter l'état et l'ID d'exécution cible d'une exécution d'annulation.

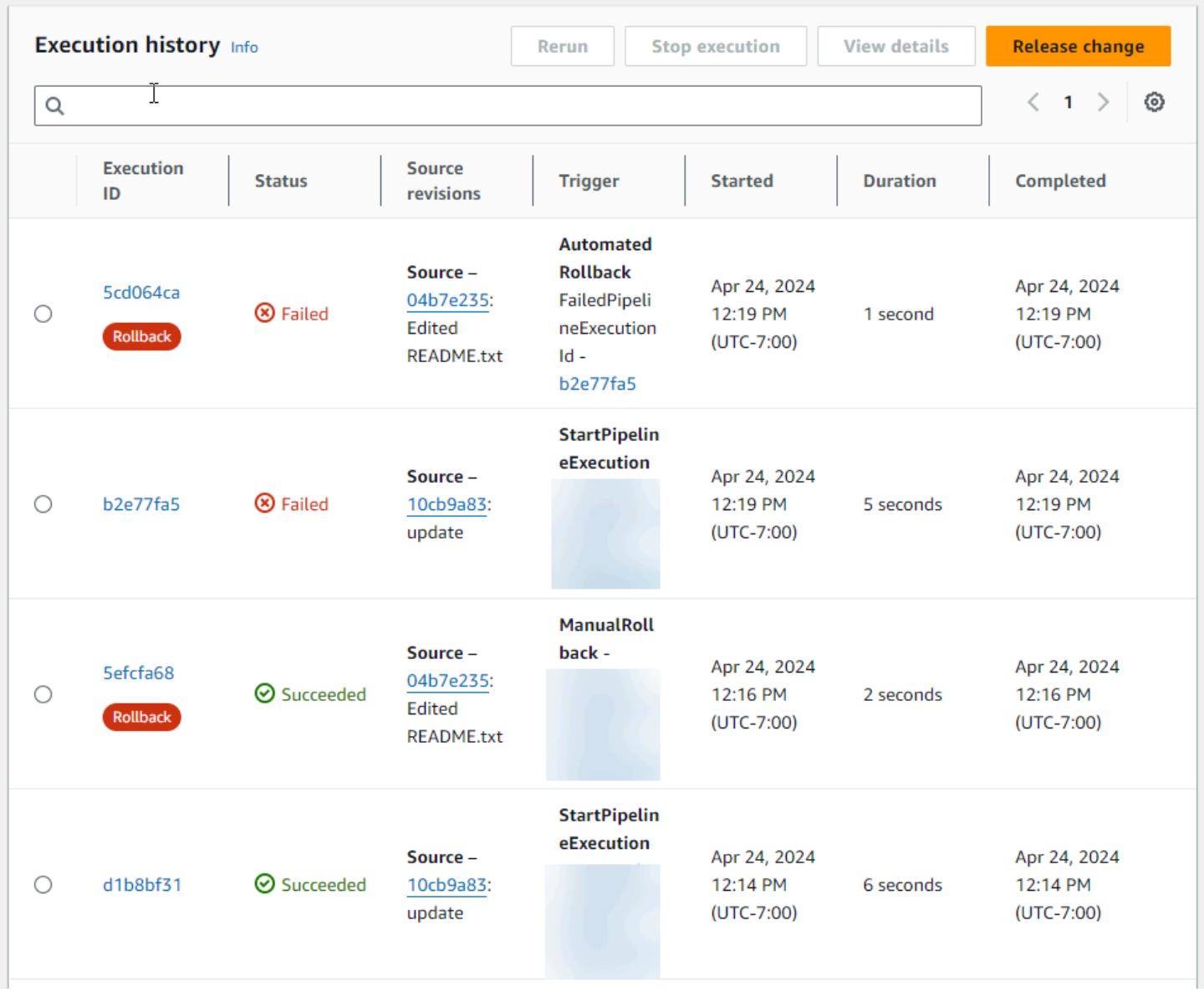
Afficher l'état de restauration dans la liste des exécutions (console)

Vous pouvez utiliser la console pour afficher l'état et l'ID d'exécution cible d'une exécution annulée dans la liste des exécutions.

Afficher l'état d'exécution de la restauration dans la liste des exécutions (console)

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms et l'état de tous les pipelines qui vous Compte AWS sont associés sont affichés.
2. Dans Nom, choisissez le nom du pipeline que vous souhaitez afficher.
3. Choisissez History (Historique). La liste des exécutions affiche le libellé Rollback.



Choisissez l'ID d'exécution dont vous souhaitez afficher les détails.

Afficher l'état de restauration avec list-pipeline-execution (CLI)

Vous pouvez utiliser l'interface de ligne de commande pour afficher l'état et l'ID d'exécution cible d'une exécution d'annulation.

- Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et exécutez la list-pipeline-executions commande : AWS CLI

```
aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline
```

Cette commande renvoie la liste de toutes les exécutions terminées associées au pipeline.

L'exemple suivant montre les données renvoyées pour un pipeline nommé *MyFirstPipeline* là où une exécution de restauration a démarré le 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"
```

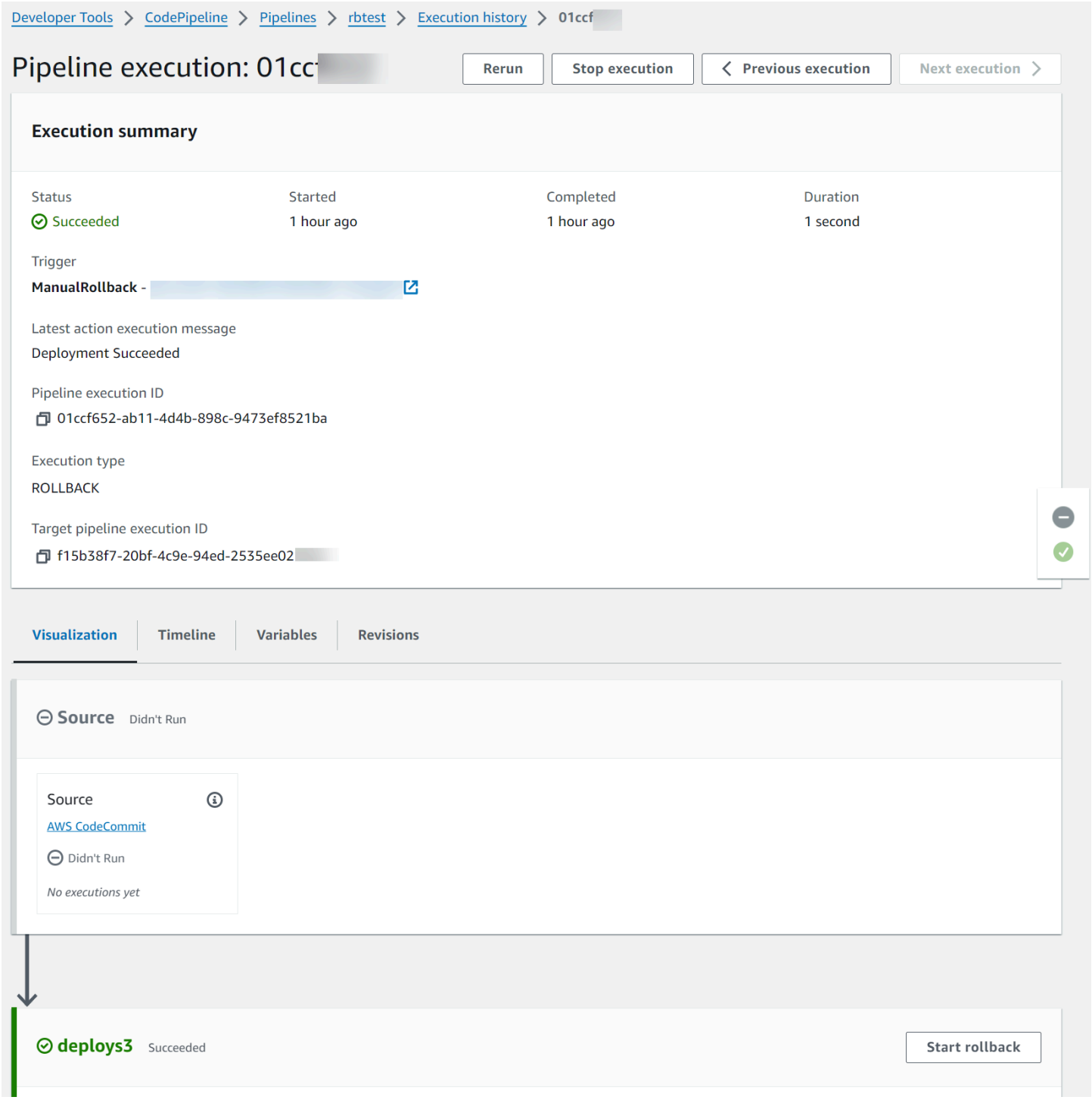
```
                }
            ],
            "trigger": {
                "triggerType": "StartPipelineExecution",
                "triggerDetail": "arn:aws:sts::<account_ID>:assumed-role/<role>"
            },
            "executionMode": "SUPERSEDED"
        },
        {
            "pipelineExecutionId": "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"
            }
         },
```

Afficher les détails de l'état de restauration

Vous pouvez consulter l'état et l'ID d'exécution cible d'une exécution d'annulation.

Afficher l'état de restauration sur la page détaillée (console)

Vous pouvez utiliser la console pour afficher l'état et l'ID d'exécution du pipeline cible pour une exécution d'annulation.



Afficher les détails de la restauration avec get-pipeline-execution (CLI)

Les exécutions de pipeline qui ont été annulées apparaîtront dans la sortie pour obtenir l'exécution du pipeline.

- Pour afficher les détails relatifs à un pipeline, exécutez la commande get-pipeline-execution, en spécifiant le nom unique du pipeline. Par exemple, pour afficher les détails relatifs à un pipeline nommé *MyFirstPipeline*, saisissez ce qui suit :

  ```
  aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
  execution-id 3f658bd1-69e6-4448-ba3e-79007EXAMPLE
  ```

  Cette commande affiche la structure du pipeline.

  L'exemple suivant montre les données renvoyées pour une partie d'un pipeline nomméeMyFirstPipeline, où l'ID d'exécution de la restauration et les métadonnées sont affichés.

  ```
  {
      "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"
          }
  ```

```
    }
}
```

Afficher l'état de restauration avec get-pipeline-state (CLI)

Les exécutions de pipeline qui ont été annulées apparaîtront dans la sortie pour obtenir l'état du pipeline.

- Pour afficher les détails relatifs à un pipeline, exécutez la commande get-pipeline-state, en spécifiant le nom unique du pipeline. Par exemple, pour afficher les détails de l'état d'un pipeline nommé*MyFirstPipeline*, entrez ce qui suit :

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

  L'exemple suivant montre les données renvoyées avec le type d'exécution rollback.

  ```
  {
      "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"
}
```

Configuration des conditions pour une étape

Vous pouvez spécifier une condition pour une étape, par exemple en vérifiant la présence d'une variable spécifique dans l'exécution du pipeline, puis activer un résultat pour la condition, par exemple en sautant l'étape ou en l'échouant. Un pipeline peut être configuré pour vérifier les conditions de l'étape pendant l'exécution, en spécifiant les vérifications pour une étape, puis en spécifiant comment l'étape doit se poursuivre lorsque certaines conditions sont remplies. Les conditions contiennent une ou plusieurs règles disponibles dans une liste de règles dans CodePipeline. Si toutes les règles d'une condition réussissent, la condition est remplie. Vous pouvez configurer les conditions de sorte que lorsque les critères ne sont pas remplis, le résultat spécifié soit activé.

Chaque condition possède un ensemble de règles qui est un ensemble ordonné de règles évaluées ensemble. Par conséquent, si une règle échoue dans la condition, la condition échoue. Vous pouvez modifier les conditions des règles lors de l'exécution du pipeline.

Les conditions sont utilisées pour des types d'expressions spécifiques et chacune possède des options spécifiques pour les résultats disponibles comme suit :

- Entrée - Les conditions pour effectuer des contrôles qui, si elles sont remplies, permettent d'accéder à une étape. Les règles sont activées avec les options de résultat suivantes : Echec ou Ignorer
- En cas d'échec - Les conditions pour effectuer des vérifications pour l'étape en cas d'échec. Les règles sont activées avec l'option de résultat suivante : Rollback
- En cas de succès - Les conditions pour effectuer des vérifications pour l'étape où elle réussit. Les règles sont activées avec les options de résultat suivantes : Rollback ou Fail

Les conditions sont soutenues par un ensemble de règles pour chaque type de condition.

Pour chaque type de condition, des actions spécifiques sont définies par la condition. L'action résulte de la réussite ou de l'échec de la vérification de l'état. Par exemple, la condition d'entrée (condition d'entrée) rencontre une alarme (règle), puis la vérification est réussie et le résultat (action) est que l'entrée de l'étape est bloquée.

Vous pouvez également utiliser la AWS CodePipeline console ou le AWS CLI pour revenir manuellement en arrière ou réessayer une étape ou les actions d'une étape. Consultez Configuration des conditions pour une étape.

Rubriques

- Cas d'utilisation pour les conditions scéniques
- Considérations relatives aux résultats configurés pour les conditions de l'étape
- Considérations relatives aux règles configurées pour les conditions de l'étape
- Création de conditions d'entrée
- Création de conditions en cas de défaillance
- Créer des conditions de réussite
- Supprimer les conditions de l'étape
- Conditions de phase dérogatoires

Cas d'utilisation pour les conditions scéniques

Les conditions de phase présentent de nombreux cas d'utilisation pour configurer la sécurité des rejets et des changements dans les pipelines. Voici des exemples de cas d'utilisation des conditions de stade.

- Utilisez une condition d'entrée pour définir une condition qui vérifiera l'état de l' CloudWatchalarme, qui bloquera ensuite toute modification si l'environnement de production n'est pas en bon état.
- Utilisez une condition d'entrée avec un temps d'attente de 60 pour définir une condition à évaluer lorsque toutes les actions d'une étape sont terminées avec succès, puis annulez les modifications si une CloudWatch alarme passe en état d'ALARME dans les 60 minutes.
- Utilisez une condition En cas de réussite pour définir une condition afin que, lorsque l'étape se termine avec succès, la règle vérifie si l'heure actuelle se situe dans la fenêtre de déploiement, puis déploie si la règle réussit.

Considérations relatives aux résultats configurés pour les conditions de l'étape

Les considérations relatives aux conditions de stage sont les suivantes :

- Vous ne pouvez pas utiliser une nouvelle tentative d'étape automatique avec les conditions OnFailure.
- Lors de la configuration d'une condition avec un résultat de restauration, le stage ne peut revenir à une exécution précédente que si la version actuelle de la structure de pipeline est disponible.

- Lorsque vous configurez une condition avec un résultat d'annulation, vous ne pouvez pas revenir à un ID d'exécution cible qui est un type d'exécution d'annulation.
- Pour les conditions d'entrée qui utilisent le résultat Skip pour sauter l'étape si la condition échoue, seules les `VariableCheck` règles `LambdaInvoke` et sont prises en charge.
- Vous ne pouvez pas réessayer manuellement une étape sur une étape en statut Ignorée.
- Vous ne pouvez pas effectuer de restauration manuelle vers une étape dont le statut est ignoré.
- Vous ne pouvez pas modifier une condition si elle est configurée avec un résultat Ignorer.
- À l'exception des résultats Ignorer, vous pouvez modifier une condition d'étape lors du démarrage de l'exécution d'un pipeline. Pour une condition d'étape dans laquelle une dérogation est activée, l'exécution se déroulera comme indiqué dans le tableau suivant.

Type	Résultat configuré en cas de défaillance de la condition	Statut de phase	Ignorer le comportement
Entry	Fail	In-progress	L'étape se poursuit.
Entry	Ignorer	Ignoré	Non applicable.
OnFailure	Restauration	Échec	L'étape est un échec.
OnSuccess	Restauration	Réussi(e)	L'étape se poursuit.
OnSuccess	Fail	Échec	L'étape se poursuit.

Considérations relatives aux règles configurées pour les conditions de l'étape

Les considérations relatives aux règles disponibles pour les conditions d'étape sont les suivantes :

- Pour la `LambdaInvoke` règle, vous devez d'abord configurer la fonction Lambda à utiliser dans la règle. Préparez l'ARN de la fonction Lambda à fournir lorsque vous configurez la règle.
- Pour la `CloudWatchAlarm` règle, vous devez d'abord configurer l'événement CloudWatch Events à utiliser dans la règle. Préparez l'ARN de l'événement à fournir lorsque vous configurez la règle.

Création de conditions d'entrée

Vous pouvez configurer les conditions d'entrée pour une étape à l'aide de la console ou de l'interface de ligne de commande. Vous allez configurer les règles et les résultats correspondants pour chaque condition. Pour obtenir un résultat d'annulation, le pipeline ne peut revenir à une exécution précédente que si l'exécution précédente a été démarrée dans la version actuelle de la structure du pipeline.

Les étapes fournissent un exemple de condition d'entrée qui utilise une règle de surveillance.

Pour plus d'informations, consultez la section Condition et RuleExecution dans le Guide de l'CodePipeline API. RuleTypeId

Création de conditions d'entrée - exemple de CloudWatchAlarm règle (console)

Vous pouvez configurer les conditions d'entrée pour une étape, ainsi que les règles et les résultats que vous souhaitez que la phase exécute lorsque les conditions sont remplies.

Configurer une condition d'entrée (console)

1. Renseignez tous les prérequis, tels que la création de la ressource et de l'ARN pour une règle dans laquelle une ressource est fournie, telle que le AWS CloudWatchAlarm.
2. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms et l'état de tous les pipelines qui vous Compte AWS sont associés sont affichés.
3. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier.
4. Sur la page des détails du pipeline, choisissez Edit.
5. Sur la page Modifier, pour l'action que vous souhaitez modifier, choisissez Modifier l'étape.
6. Choisissez Ajouter une condition d'entrée. La carte de condition avant l'entrée dans l'étape s'affiche avec l'option Échec disponible pour cette condition.
7. Choisissez Ajouter une règle, puis procédez comme suit.

   a. Dans Nom de la règle, entrez le nom de votre règle. Pour cet exemple, entrez `MyAlarmRule`.

b. Dans Fournisseur de règles, choisissez le fournisseur de règles préconfiguré à ajouter à votre condition. Pour cet exemple, choisissez AWS CloudWatchAlarm, puis procédez comme suit.

c. Dans Région, choisissez la région correspondant à votre état ou laissez la valeur par défaut.

d. Dans Nom de l'alarme, choisissez la CloudWatch ressource à utiliser pour la règle. Vous devez déjà avoir créé la ressource dans votre compte.

e. (Facultatif) Dans Temps d'attente, entrez le temps CodePipeline d'attente si l'alarme est en état ALARME lors de sa première évaluation. Si l'état de l'alarme est OK lorsque la règle est vérifiée pour la première fois, la règle aboutira immédiatement.

f. (Facultatif) Entrez tout état d'alarme spécifique à surveiller et entrez le rôle ARN le cas échéant.

g. Lorsque vous avez terminé de modifier la scène, choisissez OK. Sur la page de modification du pipeline, choisissez Enregistrer.

8. Après la course, visualisez le résultat.

Création de conditions de saisie avec Ignorer le résultat et la **`VariableCheck`** règle (console)

Vous pouvez configurer les conditions d'entrée pour une étape de sorte que si la condition d'entrée n'est pas remplie, l'étape soit ignorée. Si la condition échoue, le résultat s'active et l'étape est ignorée. Lorsqu'une étape est ignorée, le statut de l'étape est Ignorée et le statut de l'action est Non exécutée. Pour les considérations relatives aux conditions de stade avec les résultats Skip, voirConsidérations relatives aux résultats configurés pour les conditions de l'étape.

Dans l'exemple suivant, la règle de vérification des variables détermine que la valeur ne correspond pas et l'étape de génération est ignorée.

Configurez une condition d'entrée avec Ignorer résultat (console)

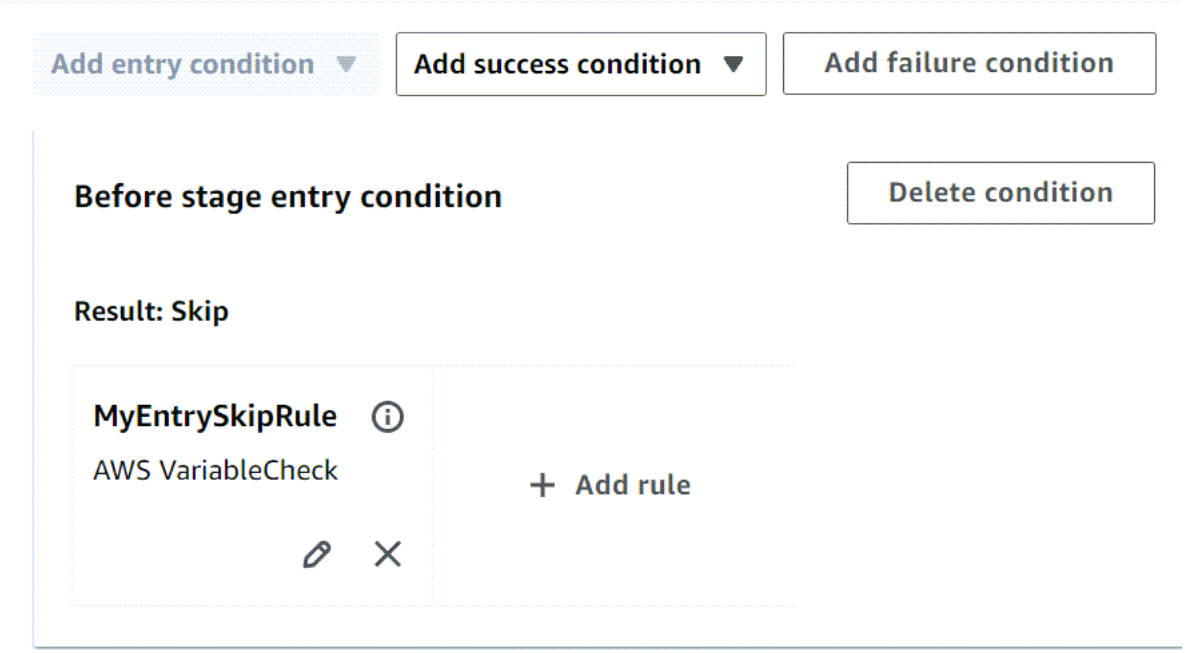
1. Renseignez tous les prérequis, tels que la création de la ressource et de l'ARN pour une règle dans laquelle une ressource est fournie, telle que le AWS CloudWatchAlarm.
2. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms et l'état de tous les pipelines qui vous Compte AWS sont associés sont affichés.

3. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier.
4. Sur la page des détails du pipeline, choisissez Edit.
5. Sur la page Modifier, pour l'action que vous souhaitez modifier, choisissez Modifier l'étape.
6. Choisissez Ajouter une condition d'entrée, puis choisissez Ignorer comme résultat.
7. Choisissez Ajouter une règle, puis procédez comme suit.

   a. Dans Nom de la règle, entrez le nom de votre règle. Pour cet exemple, entrez `MyAlarmRule`.

   b. Dans Fournisseur de règles, choisissez le fournisseur de règles préconfiguré à ajouter à votre condition. Pour cet exemple, choisissez VariableCheck, puis effectuez les étapes suivantes.

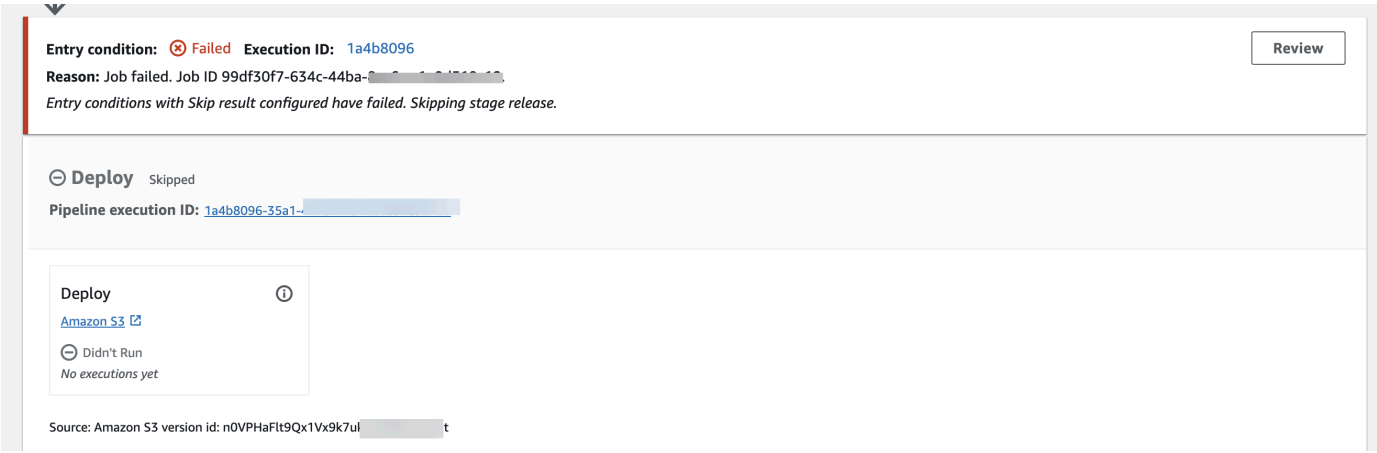


   c. Dans Région, choisissez la région correspondant à votre état ou conservez la valeur par défaut.

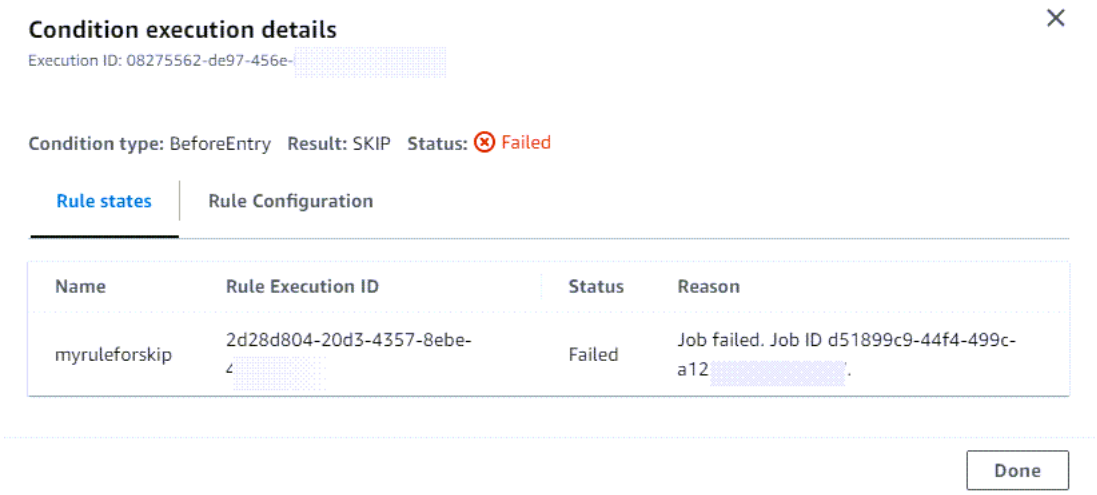
   d. Dans Variable, choisissez la variable à comparer, par exemple `#{SourceVariables.FullRepositoryName}` pour un pipeline qui possède une action source GitHub (via GitHub App). Entrez le nom du référentiel et choisissez l'opérateur, tel que Equals.

   e. Lorsque vous avez terminé de modifier la scène, choisissez OK. Sur la page de modification du pipeline, choisissez Enregistrer.

8. Après la course, visualisez le résultat.



9. Pour consulter les détails, choisissez Réviser. Le détail de l'exemple suivant montre que le résultat configuré pour la condition est Skip, qui ne peut pas être remplacé. Le statut de la règle est Échec car la condition n'est pas remplie.



Création de conditions d'entrée (CLI)

AWS CLI Pour configurer une condition d'entrée, utilisez les commandes pour créer ou mettre à jour un pipeline comme indiqué dans Création d'un pipeline, d'étapes et d'actions etModifier un pipeline dans CodePipeline.

Configuration de la condition et de la règle ou des règles (CLI)

- Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et exécutez la AWS CLI `update-pipeline` commande en spécifiant la condition de défaillance dans la structure du pipeline. L'exemple suivant configure une condition d'entrée pour une étape nommée `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"
                }
            ]
        }
    ]
  }
}
```

Pour plus d'informations sur la configuration des conditions de réussite pour la restauration des étapes, consultez SuccessConditions la référence de l'CodePipeline API.

Création de conditions d'entrée (CFN)

CloudFormation Pour configurer une condition d'entrée, utilisez le `beforeEntry` paramètre. À l'entrée, la scène exécutera la règle et exécutera le résultat.

```
beforeEntry:
     Result: FAIL
```

- Mettez à jour le modèle comme indiqué dans l'extrait suivant. L'exemple suivant configure une condition d'entrée avec une règle nommée `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
```

Pour plus d'informations sur la configuration des conditions BeforeEntry, consultez AWS::CodePipeline : :Pipeline `StageDeclaration` dans le BeforeEntryConditions Guide de l'CloudFormation utilisateur.

Création de conditions en cas de défaillance

Vous pouvez configurer les conditions On Failure pour une étape à l'aide de la console ou de l'interface de ligne de commande. Vous allez configurer les règles et les résultats correspondants pour chaque condition. Pour obtenir un résultat d'annulation, le pipeline ne peut revenir à une exécution précédente que si l'exécution précédente a été démarrée dans la version actuelle de la structure du pipeline.

Création de conditions en cas de défaillance (console)

Vous pouvez configurer les conditions En cas d'échec pour une étape, ainsi que les règles et les résultats que vous souhaitez que la phase exécute lorsque les conditions sont remplies.

Configurer une condition en cas de défaillance (console)

1. Renseignez tous les prérequis, tels que la création de la ressource et de l'ARN pour une règle dans laquelle une ressource est fournie, telle que la LambdaInvoke règle.
2. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

Les noms et l'état de tous les pipelines qui vous Compte AWS sont associés sont affichés.

3. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier.
4. Sur la page des détails du pipeline, choisissez Edit.
5. Sur la page Modifier, pour l'action que vous souhaitez modifier, choisissez Modifier l'étape.
6. Choisissez Ajouter une condition de défaillance. La carte de condition de défaillance s'affiche avec l'option Rollback disponible pour cette condition.
7. Choisissez Ajouter une règle, puis procédez comme suit.

   a. Dans Nom de la règle, entrez le nom de votre règle. Pour cet exemple, entrez `MyLambdaRule`.
   b. Dans Fournisseur de règles, choisissez le fournisseur de règles préconfiguré à ajouter à votre condition. Pour cet exemple, choisissez AWS LambdaInvoke, puis procédez comme suit.
   c. Dans Région, choisissez la région correspondant à votre état ou conservez la valeur par défaut.
   d. Dans Artefacts d'entrée, choisissez l'artefact source.
   e. Dans Nom de la fonction, choisissez la ressource Lambda à utiliser pour la règle. Vous devez déjà avoir créé la ressource dans votre compte.
   f. (Facultatif) Dans Paramètres utilisateur, entrez toutes les paires qui représentent des paramètres pour une configuration supplémentaire.
   g. (Facultatif) Dans Role Arn, entrez le rôle ARN s'il est configuré.
   h. (Facultatif) Dans Délai d'attente en minutes, entrez la durée en minutes pendant laquelle la règle doit attendre avant l'expiration du délai.
   i. Lorsque vous avez terminé de modifier la scène, choisissez OK. Sur la page de modification du pipeline, choisissez Enregistrer.

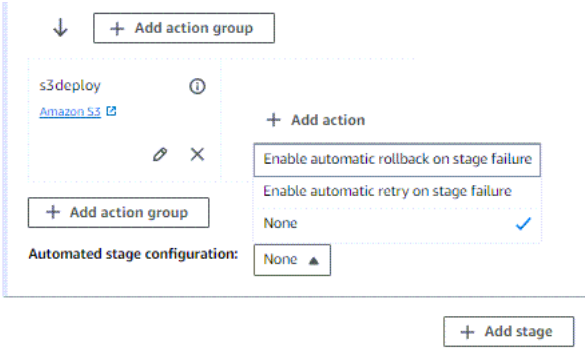
Exemple de création de conditions OnFailure avec un résultat de nouvelle tentative (console)

Vous pouvez configurer les conditions OnFailure pour une étape de sorte que si la condition d'entrée n'est pas remplie, la phase soit réessayée. Dans le cadre de ce résultat, vous configurez le mode de nouvelle tentative, en spécifiant s'il faut réessayer les actions qui ont échoué ou réessayer l'étape qui a échoué.

Configurer une condition OnFailure avec un résultat de nouvelle tentative (console)

1. Renseignez tous les prérequis, tels que la création de la ressource et de l'ARN pour une règle dans laquelle une ressource est fournie, telle que le AWS CloudWatchAlarm.
2. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms et l'état de tous les pipelines qui vous Compte AWS sont associés sont affichés.
3. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier.
4. Sur la page des détails du pipeline, choisissez Edit.
5. Sur la page Modifier, pour l'action que vous souhaitez modifier, choisissez Modifier l'étape.
6. Au bas de l'étape, sous Configuration automatique de la phase :, choisissez Activer la nouvelle tentative automatique en cas d'échec de la phase. En mode Réessayer, choisissez l'étape Réessayer avec échec ou Réessayer les actions qui ont échoué.



7. Choisissez d'ajouter une condition OnFailure, puis choisissez Ajouter une règle et entrez une règle pour la condition.

   a. Dans Nom de la règle, entrez le nom de votre règle. Pour cet exemple, entrez `MyAlarmRule`.

   b. Dans Fournisseur de règles, choisissez le fournisseur de règles préconfiguré à ajouter à votre condition. Pour cet exemple, choisissez CloudWatchAlarm, puis effectuez les étapes suivantes.

   c. Dans Région, choisissez la région correspondant à votre état ou conservez la valeur par défaut.

   d. Dans Nom de l'alarme, choisissez la ressource configurée pour l'alerte.

   e. Lorsque vous avez terminé de modifier la scène, choisissez OK. Sur la page de modification du pipeline, choisissez Enregistrer.

8. Après la course, visualisez le résultat.

Création de conditions en cas de défaillance (CLI)

AWS CLI Pour configurer une condition En cas de défaillance, utilisez les commandes pour créer ou mettre à jour un pipeline comme indiqué dans Création d'un pipeline, d'étapes et d'actions etModifier un pipeline dans CodePipeline.

Configuration de la condition et de la règle ou des règles (CLI)

- Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et exécutez la AWS CLI `update-pipeline` commande en spécifiant la condition de défaillance dans la structure du pipeline. L'exemple suivant configure une condition On Failure pour une étape nommée `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"
                    }
                ]
            }
        ]
    }
}
```

Pour plus d'informations sur la configuration des conditions de défaillance, consultez FailureConditions la référence de l'CodePipeline API.

Création de conditions en cas de défaillance (CFN)

CloudFormation Pour configurer une condition En cas de défaillance, utilisez le `OnFailure` paramètre. En cas d'échec, la scène exécutera la règle et exécutera le résultat.

```
OnFailure:
     Result: ROLLBACK
```

- Mettez à jour le modèle comme indiqué dans l'extrait suivant. L'exemple suivant configure une OnFailure condition avec une règle nommée `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
```

Pour plus d'informations sur la configuration des conditions de défaillance, consultez OnFailure `StageDeclaration` le Guide de l'CloudFormation utilisateur ci-dessous.

Créer des conditions de réussite

Vous pouvez configurer les conditions On Success pour une étape à l'aide de la console ou de l'interface de ligne de commande. Vous allez configurer les règles et les résultats correspondants pour chaque condition. Pour obtenir un résultat d'annulation, le pipeline ne peut revenir à une exécution précédente que si l'exécution précédente a été démarrée dans la version actuelle de la structure du pipeline.

Les étapes fournissent un exemple de condition En cas de réussite qui utilise une règle de fenêtre de déploiement.

Pour plus d'informations, consultez la section Condition et RuleExecution dans le Guide de l'CodePipeline API. RuleTypeId

Création selon les conditions de réussite (console)

Vous pouvez configurer les conditions On Success pour une étape, ainsi que les règles et les résultats que vous souhaitez que la phase exécute lorsque les conditions sont remplies.

Configurer une condition En cas de réussite (console)

1. Renseignez tous les prérequis, tels que la création de la ressource et de l'ARN pour une règle dans laquelle une ressource est fournie, telle que le AWS LambdaRule.
2. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

   Les noms et l'état de tous les pipelines qui vous Compte AWS sont associés sont affichés.
3. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier.
4. Sur la page des détails du pipeline, choisissez Edit.
5. Sur la page Modifier, pour l'action que vous souhaitez modifier, choisissez Modifier l'étape.
6. Choisissez Ajouter une condition de réussite. La carte des conditions de réussite sur scène s'affiche. Choisissez l'option Rollback ou Fail qui apparaît comme les résultats disponibles pour ce type de condition.
7. Choisissez Ajouter une règle, puis procédez comme suit.

   a. Dans Nom de la règle, entrez le nom de votre condition. Pour cet exemple, entrez `MyDeploymentRule`.

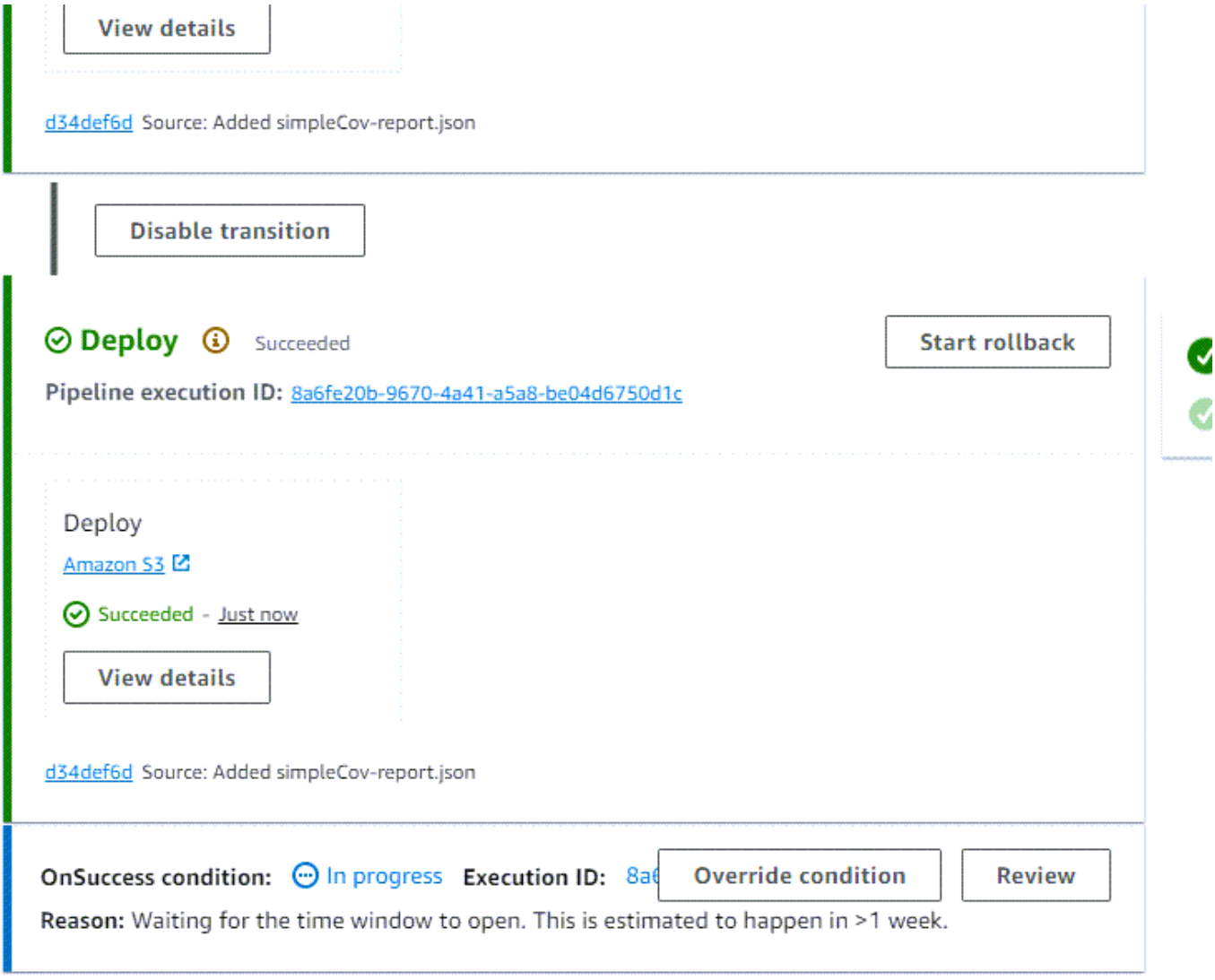
b. Dans Fournisseur de règles, choisissez la règle préconfigurée à ajouter à votre condition. Pour cet exemple, choisissez AWS DeploymentWindow, puis effectuez les étapes suivantes.

c. Dans Région, choisissez la région correspondant à votre état ou conservez la valeur par défaut.

d. Dans Cron, entrez l'expression cron pour la fenêtre de déploiement. L'expression cron définit les jours et les heures auxquels le déploiement doit être autorisé. Pour obtenir des informations de référence sur les expressions cron, voir Utilisation des expressions cron et rate pour planifier des règles.

e. (Facultatif) Dans TimeZone, entrez le fuseau horaire de la fenêtre de déploiement.

8. Après la course, visualisez le résultat.



Création de conditions de réussite (CLI)

Pour utiliser la condition AWS CLI Pour configurer une condition En cas de réussite, utilisez les commandes pour créer ou mettre à jour un pipeline comme indiqué dans Création d'un pipeline, d'étapes et d'actions etModifier un pipeline dans CodePipeline.

Configuration de la condition et de la règle ou des règles (CLI)

- Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et utilisez le AWS CLI pour exécuter la `update-pipeline` commande, en spécifiant la condition de défaillance dans la structure du pipeline. L'exemple suivant configure une condition On Success pour une étape nomméeDeploy, où la règle est nommée `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"
                    }
                ]
            }
        ]
    }
}
```

Pour plus d'informations sur la configuration des conditions de réussite, consultez SuccessConditions la référence de l'CodePipeline API.

Création d'une condition de réussite (CFN)

CloudFormation Pour configurer une condition En cas de réussite, utilisez le `OnSuccess` paramètre. En cas de succès, la scène appliquera la règle et exécutera le résultat.

```
OnSuccess:
     Result: ROLLBACK
```

- Mettez à jour le modèle comme indiqué dans l'extrait suivant. L'exemple suivant configure une OnSuccess condition avec une règle nommée `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
```

Pour plus d'informations sur la configuration des conditions de défaillance pour la restauration des étapes, consultez OnFailure le Guide `StageDeclaration` de l'CloudFormation utilisateur ci-dessous.

Supprimer les conditions de l'étape

Vous pouvez supprimer les conditions d'étape qui ont été configurées pour votre pipeline.

Pour supprimer une condition d'étape

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

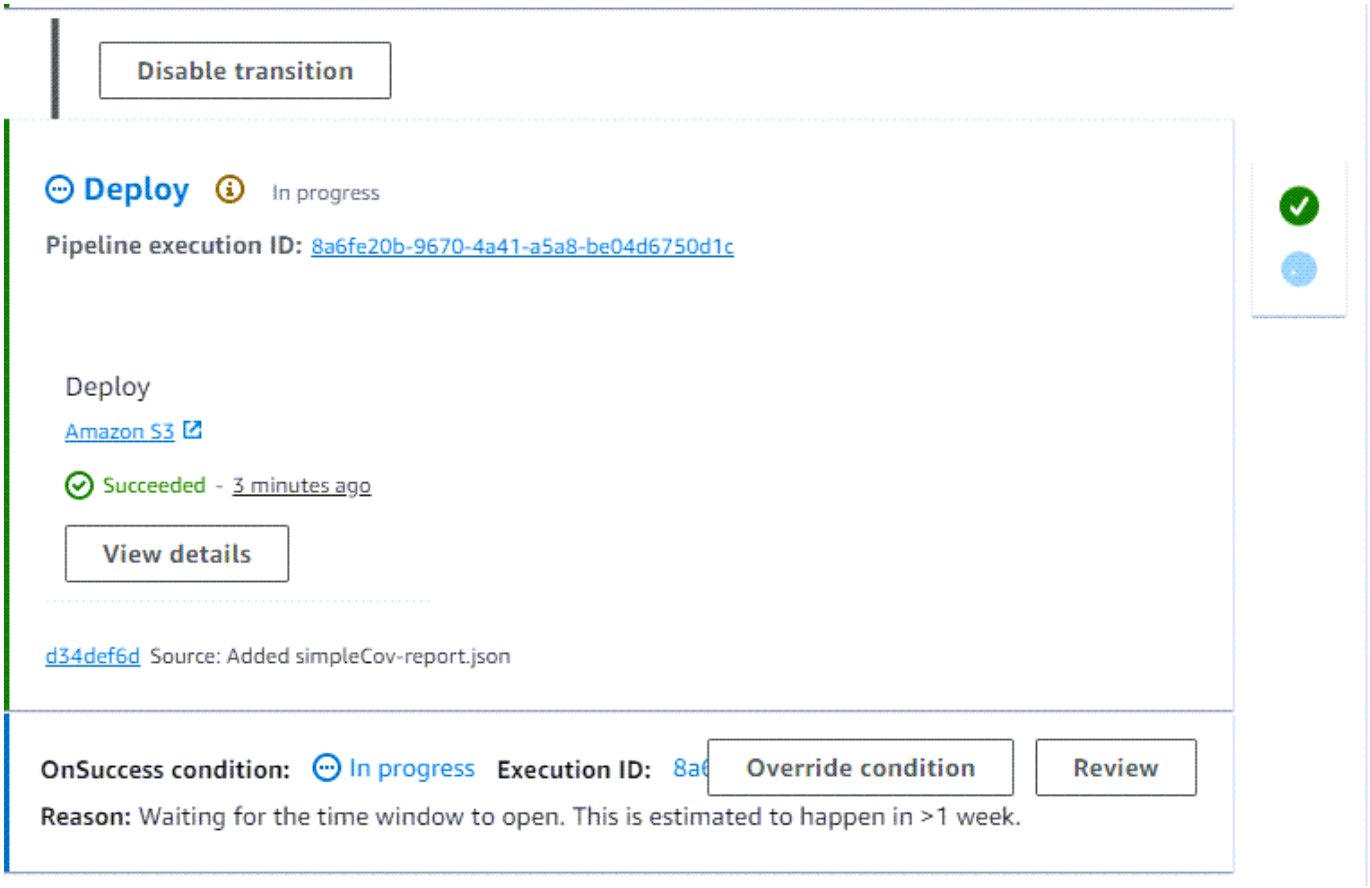
   Les noms et l'état de tous les pipelines qui vous Compte AWS sont associés sont affichés.
2. Dans Name, choisissez le nom du pipeline que vous souhaitez modifier.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Sur la page Modifier, pour la condition que vous souhaitez modifier, choisissez Modifier l'étape.
5. À côté de la condition que vous souhaitez supprimer, choisissez Supprimer la condition.

Conditions de phase dérogatoires

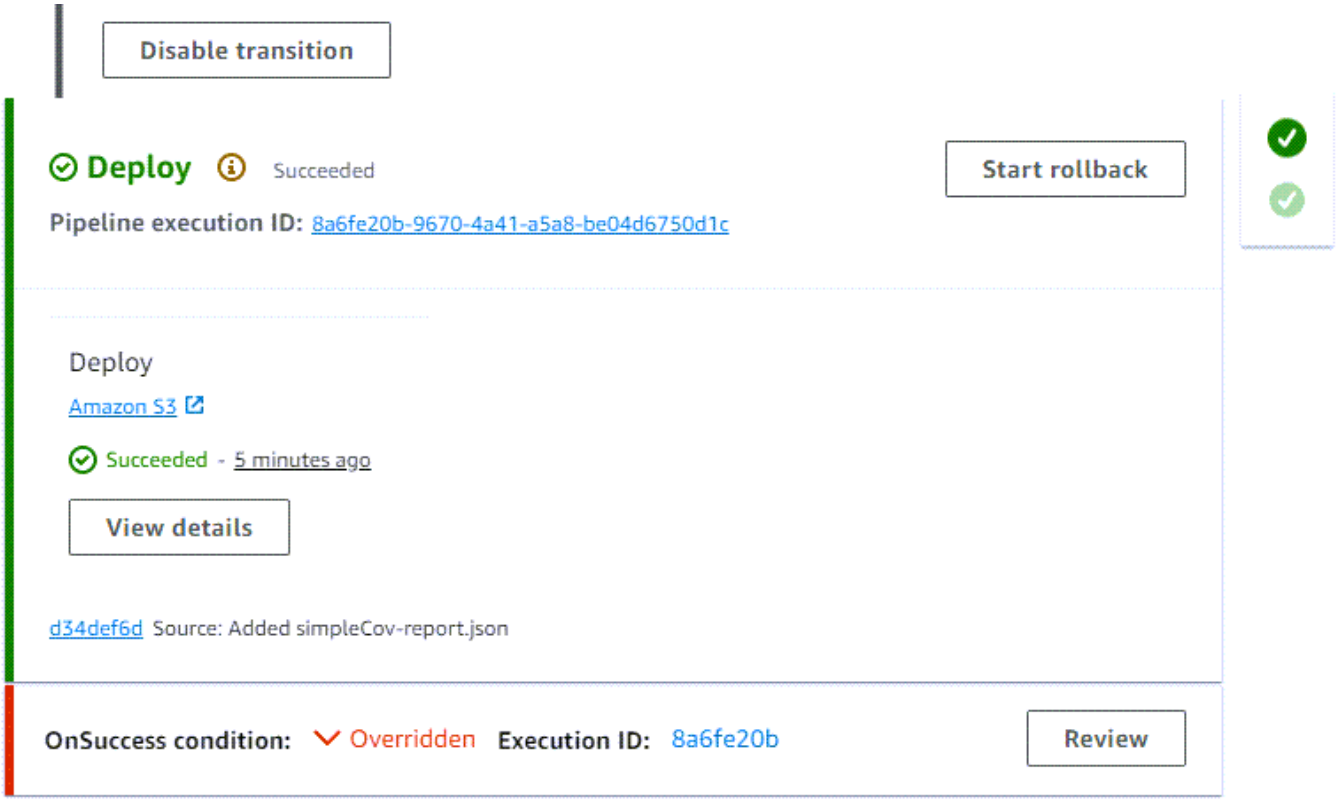
Vous pouvez modifier les conditions d'étape qui ont été configurées pour votre pipeline. Dans la console, lorsque la phase et la règle sont en cours d'exécution, vous pouvez choisir de modifier la condition de la phase. Cela entraîne le déroulement de l'étape.

Pour annuler une condition d'étape

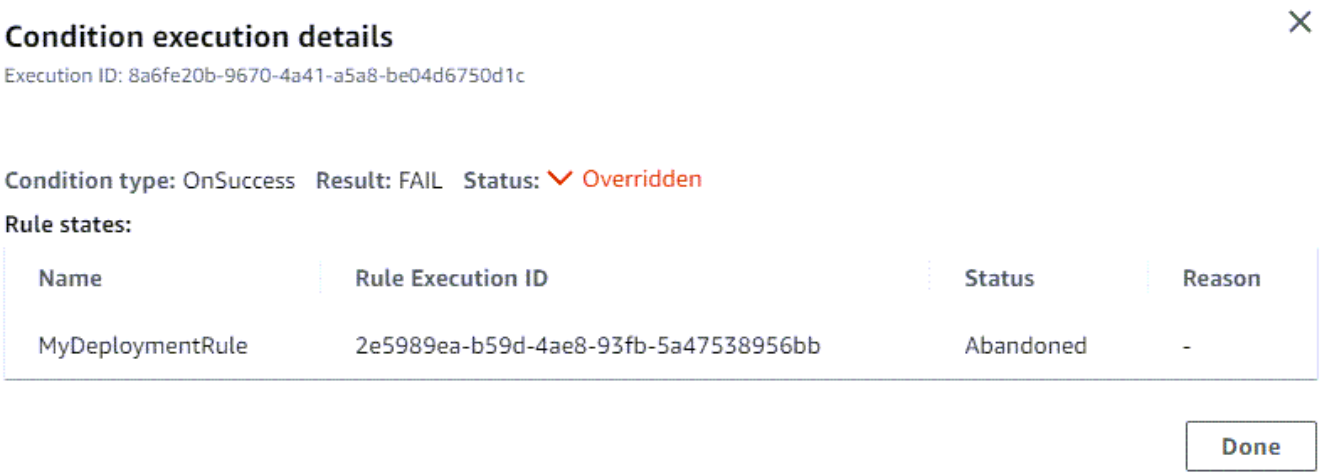
1. Dans cet exemple, l'étape du pipeline est exécutée avec une condition. Le bouton Ignorer est activé.



2. À côté de la condition que vous souhaitez annuler, choisissez Remplacer.



3. Pour consulter les détails, choisissez Réviser. Le détail de l'exemple suivant montre que le résultat configuré pour la condition est Fail, qui a été remplacé. Le statut de la règle est Abandonné en raison de cette dérogation.



Utiliser des types d'actions, des actions personnalisées et des actions d'approbation

Dans AWS CodePipeline, une action fait partie de la séquence d'une étape d'un pipeline. Une action est une tâche effectuée sur l'artéfact de l'étape. Les actions de pipeline interviennent dans un ordre précis, en séquence ou en parallèle, en fonction de ce qui a été défini dans la configuration de l'étape.

CodePipeline fournit un soutien pour six types d'actions :

- Source
- Génération
- test
- Déploiement
- Approbation
- Invocation

Pour plus d'informations sur les Service AWS produits et services partenaires que vous pouvez intégrer à votre pipeline en fonction du type d'action, voirIntégrations avec les types CodePipeline d'action.

Rubriques

- Utilisation des types d'actions
- Créez et ajoutez une action personnalisée dans CodePipeline
- Marquer une action personnalisée dans CodePipeline
- Invoquer un AWS Lambda fonction dans un pipeline dans CodePipeline
- Ajouter une action d'approbation manuelle à une étape
- Ajouter une action interrégionale dans CodePipeline
- Utilisation des variables

Utilisation des types d'actions

Les types d'actions sont des actions préconfigurées que vous créez, en tant que fournisseur, pour les clients à l'aide de l'un des modèles d'intégration pris en charge dans AWS CodePipeline.

Vous pouvez demander, consulter et mettre à jour des types d'actions. Si le type d'action est créé pour votre compte en tant que propriétaire, vous pouvez l'utiliser AWS CLI pour afficher ou mettre à jour les propriétés et la structure de votre type d'action. Si vous êtes le fournisseur ou le propriétaire du type d'action, vos clients peuvent choisir l'action et l'ajouter à leurs pipelines une fois qu'elle sera disponible dans CodePipeline.

> **Note**
>
> Vous créez des `custom` actions `owner` sur le terrain à exécuter avec un collaborateur. Vous ne les créez pas à l'aide d'un modèle d'intégration. Pour plus d'informations sur les actions personnalisées, consultezCréez et ajoutez une action personnalisée dans CodePipeline.

Composants du type d'action

Les composants suivants constituent un type d'action.

- ID du type d'action — L'ID comprend la catégorie, le propriétaire, le fournisseur et la version. L'exemple suivant montre un ID de type d'action dont le propriétaire`ThirdParty`, la catégorie`Test`, le fournisseur est nommé `TestProvider` et une version de celui-ci`1`.

```
{
    "Category": "Test",
    "Owner": "ThirdParty",
    "Provider": "TestProvider",
    "Version": "1"
},
```

- Configuration de l'exécuteur : modèle d'intégration, ou moteur d'action, spécifié lors de la création de l'action. Lorsque vous spécifiez l'exécuteur d'un type d'action, vous choisissez l'un des deux types suivants :
  - Lambda : le propriétaire du type d'action écrit l'intégration sous la forme d'une fonction Lambda, qui est invoquée CodePipeline chaque fois qu'une tâche est disponible pour l'action.

- JobWorker: le propriétaire du type d'action écrit l'intégration sous la forme d'un assistant qui interroge les offres d'emploi disponibles sur les pipelines de clients. L'assistant exécute ensuite la tâche et renvoie le résultat de la tâche à l'aide CodePipeline d' CodePipeline API.

  > **Note**
  >
  > Le modèle d'intégration des travailleurs n'est pas le modèle d'intégration préféré.

- Artefacts d'entrée et de sortie : limites pour les artefacts que le propriétaire du type d'action désigne pour les clients de l'action.
- Autorisations : stratégie d'autorisations qui désigne les clients qui peuvent accéder au type d'action tiers. Les stratégies d'autorisations disponibles dépendent du modèle d'intégration choisi pour le type d'action.
- URL : liens profonds vers des ressources avec lesquelles le client peut interagir, telles que la page de configuration du propriétaire du type d'action.

Rubriques

- Demander un type d'action
- Ajouter un type d'action disponible à un pipeline (console)
- Afficher un type d'action
- Mettre à jour un type d'action

Demander un type d'action

Lorsqu'un nouveau type CodePipeline d'action est demandé par un fournisseur tiers, le type d'action est créé pour le propriétaire du type d'action dans CodePipeline, et le propriétaire peut gérer et afficher le type d'action.

Un type d'action peut être une action privée ou publique. Lorsque votre type d'action est créé, il est privé. Pour demander qu'un type d'action soit transformé en action publique, contactez l'équipe CodePipeline de service.

Avant de créer votre fichier de définition d'action, vos ressources d'exécuteur et votre demande de type d'action pour l' CodePipeline équipe, vous devez choisir un modèle d'intégration.

Étape 1 : Choisissez votre modèle d'intégration

Choisissez votre modèle d'intégration, puis créez la configuration pour ce modèle. Après avoir choisi le modèle d'intégration, vous devez configurer vos ressources d'intégration.

- Pour le modèle d'intégration Lambda, vous créez une fonction Lambda et vous ajoutez des autorisations. Ajoutez des autorisations à la fonction Lambda de votre intégrateur pour fournir au CodePipeline service les autorisations nécessaires pour l'invoquer à l'aide du principal de CodePipeline service :. `codepipeline.amazonaws.com` Les autorisations peuvent être ajoutées à l'aide CloudFormation de la ligne de commande.
  - Exemple d'ajout d'autorisations à l'aide de 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
```

  - Documentation pour la ligne de commande
- Pour le modèle d'intégration des assistants professionnels, vous créez une intégration avec une liste de comptes autorisés sur lesquels l'assistant interroge les offres d'emploi à l'aide CodePipeline des API.

Étape 2 : Création d'un fichier de définition du type d'action

Vous définissez un type d'action dans un fichier de définition de type d'action à l'aide du format JSON. Dans le fichier, vous incluez la catégorie d'action, le modèle d'intégration utilisé pour gérer le type d'action et les propriétés de configuration.

> **Note**
>
> Une fois que vous avez créé une action publique, vous ne pouvez pas modifier la propriété du type d'action sous `properties` de `optional` à`required`. Vous ne pouvez pas non plus modifier le`owner`.

Pour plus d'informations sur les paramètres du fichier de définition des types d'action, consultez ActionTypeDeclaration et UpdateActionType dans le Guide de référence de l'CodePipeline API.

Le fichier de définition des types d'action comporte huit sections :

- `description`: description du type d'action à mettre à jour.
- `executor`: Informations sur l'exécuteur d'un type d'action créé avec un modèle d'intégration pris en charge, `Lambda` soit`job worker`. Vous ne pouvez fournir que l'un `jobWorkerExecutorConfiguration` ou l'autre`lambdaExecutorConfiguration`, en fonction de votre type d'exécuteur testamentaire.
  - `configuration`: Ressources pour la configuration du type d'action, en fonction du modèle d'intégration choisi. Pour le modèle d'intégration Lambda, utilisez la fonction Lambda ARN. Pour le modèle d'intégration des travailleurs, utilisez le compte ou la liste des comptes à partir desquels ils s'exécutent.
  - `jobTimeout`: délai d'attente en secondes pour la tâche. L'exécution d'une action peut consister en plusieurs tâches. Il s'agit du délai d'attente pour une seule tâche, et non pour l'exécution complète de l'action.

    > **Note**
    >
    > Pour le modèle d'intégration Lambda, le délai maximum est de 15 minutes.

  - `policyStatementsTemplate`: déclaration de politique qui spécifie les autorisations du compte du CodePipeline client qui sont nécessaires pour exécuter correctement une action.
  - `type`: modèle d'intégration utilisé pour créer et mettre à jour le type d'action, `Lambda` soit`JobWorker`.
- `id`: catégorie, propriétaire, fournisseur et ID de version pour le type d'action :
  - `category`: Le type d'action peut être effectué à l'étape suivante : Source, Création, Déploiement, Test, Invocation ou Approbation.
  - `provider`: fournisseur du type d'action appelé, tel que le nom de la société ou du produit du fournisseur. Le nom du fournisseur est fourni lors de la création du type d'action.
  - `owner`: créateur du type d'action appelé : `AWS` ou`ThirdParty`.
  - `version`: chaîne utilisée pour versionner le type d'action. Pour la première version, définissez le numéro de version sur 1.
- `inputArtifactDetails`: le nombre d'artefacts à attendre lors de l'étape précédente du pipeline.

- `outputArtifactDetails`: le nombre d'artefacts à attendre du résultat de l'étape de type d'action.
- `permissions`: détails identifiant les comptes autorisés à utiliser le type d'action.
- `properties`: les paramètres nécessaires à l'exécution des tâches de votre projet.
  - `description`: description de la propriété qui est affichée aux utilisateurs.
  - `optional`: si la propriété de configuration est facultative.
  - `noEcho`: si la valeur du champ saisie par le client est omise du journal. Si`true`, la valeur est supprimée lorsqu'elle est renvoyée avec une demande d' GetPipeline API.
  - `key`: si la propriété de configuration est une clé.
  - `queryable`: indique si la propriété est utilisée pour les sondages. Un type d'action peut avoir jusqu'à une propriété interrogeable. Si c'est le cas, cette propriété doit être à la fois obligatoire et non secrète.
  - `name`: nom de propriété affiché aux utilisateurs.
- `urls`: une liste des URL CodePipeline s'affiche pour vos utilisateurs.
  - `entityUrlTemplate`: URL vers les ressources externes pour le type d'action, comme une page de configuration.
  - `executionUrlTemplate`: URL vers les détails de la dernière exécution de l'action.
  - `revisionUrlTemplate`: URL affichée dans la CodePipeline console vers la page où les clients peuvent mettre à jour ou modifier la configuration de l'action externe.
  - `thirdPartyConfigurationUrl`: URL d'une page sur laquelle les utilisateurs peuvent s'inscrire à un service externe et effectuer la configuration initiale de l'action fournie par ce service.

Le code suivant présente un exemple de fichier de définition de type d'action.

```
{
   "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"
   }
 }
}
```

Étape 3 : Enregistrez votre intégration auprès de CodePipeline

Pour enregistrer votre type d'action CodePipeline, vous devez contacter l'équipe CodePipeline de service avec votre demande.

L'équipe CodePipeline de service enregistre l'intégration du nouveau type d'action en apportant des modifications à la base de code du service. CodePipeline enregistre deux nouvelles actions : une action publique et une action privée. Vous utilisez l'action privée à des fins de test, puis lorsque vous êtes prêt, vous activez l'action publique pour gérer le trafic client.

Pour enregistrer une demande d'intégration Lambda

- Envoyez une demande à l'équipe CodePipeline de service en utilisant le formulaire suivant.

```
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 Régions AWS where your action will be available

5. Will this be available as a public action?
```

Pour enregistrer une demande d'intégration des travailleurs

- Envoyez une demande à l'équipe CodePipeline de service en utilisant le formulaire suivant.

```
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 Régions AWS where your action will be available

5. Will this be available as a public action?
```

Étape 4 : Activez votre nouvelle intégration

Contactez l'équipe CodePipeline de service lorsque vous serez prêt à utiliser publiquement la nouvelle intégration.

Ajouter un type d'action disponible à un pipeline (console)

Vous ajoutez votre type d'action à un pipeline afin de pouvoir le tester. Vous pouvez le faire en créant un nouveau pipeline ou en modifiant un pipeline existant.

> **Note**
>
> Si votre type d'action est une action de catégorie source, de génération ou de déploiement, vous pouvez l'ajouter en créant un pipeline. Si votre type d'action appartient à la catégorie test, vous devez l'ajouter en modifiant un pipeline existant.

Pour ajouter votre type d'action à un pipeline existant depuis la CodePipeline console

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Dans la liste des pipelines, choisissez le pipeline auquel vous souhaitez ajouter le type d'action.
3. Sur la page d'affichage récapitulatif du pipeline, choisissez Modifier.
4. Choisissez de modifier la scène. À l'étape où vous souhaitez ajouter votre type d'action, choisissez Ajouter un groupe d'actions. La page Modifier l'action s'affiche.
5. Sur la page Modifier l'action, dans Nom de l'action, entrez le nom de l'action. Il s'agit du nom qui s'affiche pour l'étape de votre pipeline.
6. Dans Fournisseur d'actions, choisissez votre type d'action dans la liste.

   Notez que la valeur de la liste est basée sur celle `provider` spécifiée dans le fichier de définition du type d'action.
7. Dans Artefacts d'entrée, entrez le nom de l'artefact au format suivant :

   `Artifactname``::`*`FileName`*

   Notez que les quantités minimales et maximales autorisées sont définies en fonction de ce qui est `inputArtifactDetails` spécifié dans le fichier de définition du type d'action.
8. Choisissez Se connecter à < Action_Name >.

   Une fenêtre de navigateur s'ouvre et permet de se connecter au site Web que vous avez créé pour votre type d'action.
9. Connectez-vous à votre site Web en tant que client et suivez les étapes suivies par un client pour utiliser votre type d'action. Vos étapes varient en fonction de votre catégorie d'action, de votre site Web et de votre configuration, mais incluent généralement une action d'achèvement qui renvoie le client vers la page Modifier l'action.

10. Sur la page CodePipeline Modifier l'action, les champs de configuration supplémentaires pour l'action s'affichent. Les champs qui s'affichent sont les propriétés de configuration que vous avez spécifiées dans le fichier de définition des actions. Entrez les informations dans les champs personnalisés pour votre type d'action.

    Par exemple, si le fichier de définition d'action a spécifié une propriété nomméeHost, un champ portant le libellé Hôte s'affiche sur la page Modifier l'action pour votre action.

11. Dans Artefacts de sortie, entrez le nom de l'artefact au format suivant :

    Artifactname::*FileName*

    Notez que les quantités minimales et maximales autorisées sont définies en fonction de ce qui est `outputArtifactDetails` spécifié dans le fichier de définition du type d'action.

12. Choisissez Terminé pour revenir à la page des détails du pipeline.

    > **Note**
    >
    > Vos clients peuvent éventuellement utiliser la CLI pour ajouter le type d'action à leur pipeline.

13. Pour tester votre action, modifiez la source spécifiée dans l'étape source du pipeline ou suivez les étapes décrites dans Démarrer manuellement un pipeline.

Pour créer un pipeline avec votre type d'action, suivez les étapes décrites Création d'un pipeline, d'étapes et d'actions et choisissez votre type d'action parmi autant d'étapes que vous souhaitez tester.

Afficher un type d'action

Vous pouvez utiliser l'interface de ligne de commande pour afficher votre type d'action. Utilisez la get-action-type commande pour afficher les types d'actions créés à l'aide d'un modèle d'intégration.

Pour afficher un type d'action

1. Créez un fichier JSON d'entrée et nommez-le`file.json`. Ajoutez l'ID de votre type d'action au format JSON, comme indiqué dans l'exemple suivant.

```
{
    "category": "Test",
```

```
    "owner": "ThirdParty",
    "provider": "TestProvider",
    "version": "1"
}
```

2. Dans une fenêtre de terminal ou sur la ligne de commande, exécutez la get-action-type commande.

```
aws codepipeline get-action-type --cli-input-json file://file.json
```

Cette commande renvoie la sortie de définition d'action pour un type d'action. Cet exemple montre un type d'action créé avec le modèle d'intégration 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": []
    }
}
```

Mettre à jour un type d'action

Vous pouvez utiliser l'interface de ligne de commande pour modifier les types d'actions créés à l'aide d'un modèle d'intégration.

Pour un type d'action publique, vous ne pouvez pas mettre à jour le propriétaire, vous ne pouvez pas remplacer les propriétés facultatives par des propriétés obligatoires et vous pouvez uniquement ajouter de nouvelles propriétés facultatives.

1. Utilisez la `get-action-type` commande pour obtenir la structure de votre type d'action. Copiez la structure.
2. Créez un fichier JSON d'entrée et nommez-le`action.json`. Collez-y la structure de type d'action que vous avez copiée à l'étape précédente. Mettez à jour tous les paramètres que vous souhaitez modifier. Vous pouvez également ajouter des paramètres facultatifs.

   Pour plus d'informations sur les paramètres du fichier d'entrée, consultez la description du fichier de définition d'action dansÉtape 2 : Création d'un fichier de définition du type d'action.

   L'exemple suivant montre comment mettre à jour un exemple de type d'action créé avec le modèle d'intégration Lambda. Cet exemple apporte les modifications suivantes :

   - Change le `provider` nom en`TestProvider1`.
   - Ajoutez une limite de délai d'expiration des tâches de 900 secondes.
   - Ajoute une propriété de configuration d'action nommée `Host` qui est affichée au client à l'aide de l'action.

   ```
   {
       "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. Sur le terminal ou la ligne de commande, exécutez la update-action-type commande

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

Cette commande renvoie la sortie du type d'action correspondant à vos paramètres mis à jour.

Créez et ajoutez une action personnalisée dans CodePipeline

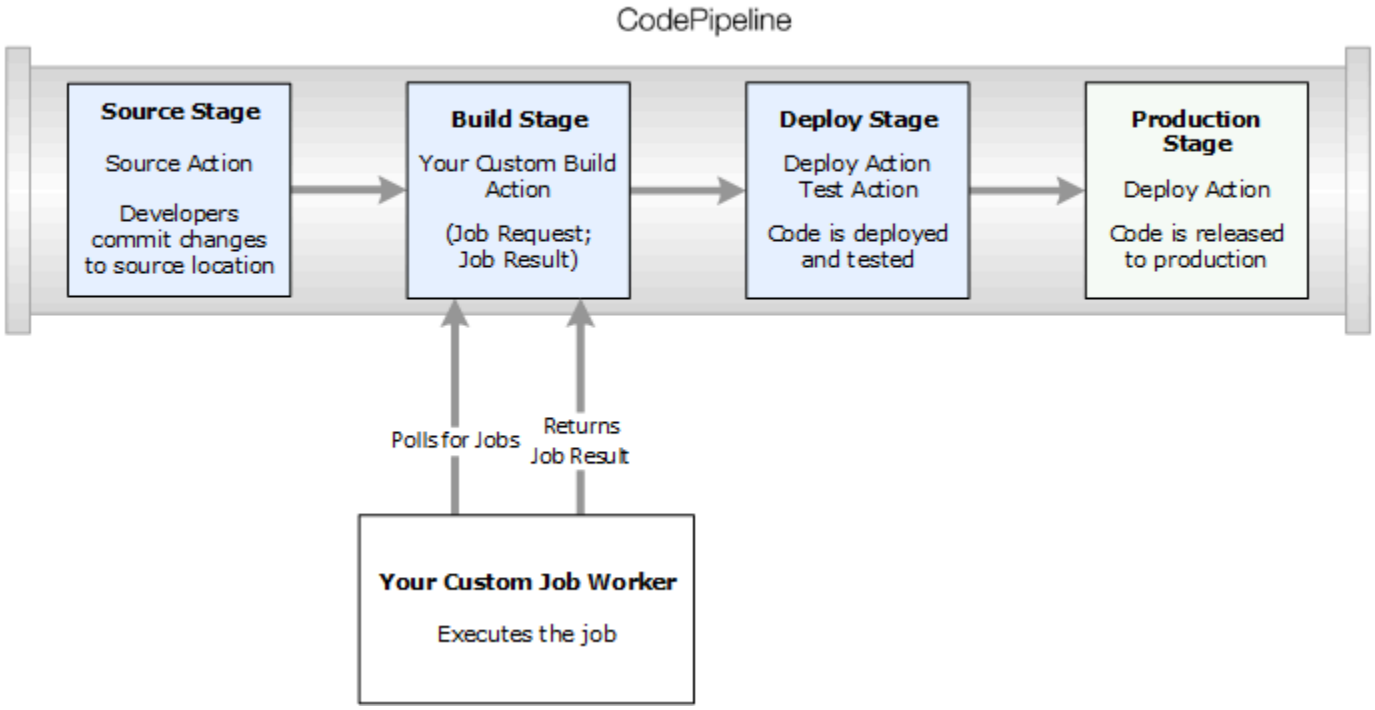
AWS CodePipeline inclut un certain nombre d'actions qui vous aident à configurer les ressources de création, de test et de déploiement pour votre processus de publication automatique. Si votre processus de publication comprend des activités qui ne sont pas incluses dans les actions par défaut, par exemple un processus de génération développé en interne ou une suite de tests, vous pouvez créer une action personnalisée à ces fins et l'inclure dans votre pipeline. Vous pouvez l'utiliser AWS CLI pour créer des actions personnalisées dans les pipelines associés à votre AWS compte.

Vous pouvez créer des actions personnalisées pour les catégories AWS CodePipeline d'actions suivantes :

- Une action de génération personnalisée qui crée ou transforme les éléments
- Une action de déploiement personnalisée qui déploie des éléments sur un ou plusieurs serveurs, sites web ou référentiels
- Une action de test personnalisée qui configure et exécute les tests automatisés
- Une action d'appel personnalisée qui exécute des fonctions

Lorsque vous créez une action personnalisée, vous devez également créer un assistant de travail qui interrogera CodePipeline les demandes de travail pour cette action personnalisée, exécutera la tâche et renverra le résultat du statut à CodePipeline. Cet exécutant de tâches peut se trouver sur n'importe quel ordinateur ou n'importe quelle ressource ayant accès au point de terminaison public pour CodePipeline. Pour gérer facilement l'accès et la sécurité, pensez à héberger votre collaborateur sur une instance Amazon EC2.

Le schéma suivant montre une vue globale d'un pipeline qui inclut une action de génération personnalisée :



Lorsqu'un pipeline inclut une action personnalisée dans le cadre d'une étape, le pipeline crée une demande de travail. Un exécutant de tâches personnalisé détecte cette demande et exécute la tâche (dans cet exemple, un processus personnalisé s'appuyant sur un logiciel tiers). Lorsque l'action est terminée, l'exécutant de tâches affiche un résultat de réussite ou d'échec. Si un résultat positif est obtenu, le pipeline fournira la révision et ses artefacts à l'action suivante. Si un échec est renvoyé, le pipeline ne fournira pas la révision à l'action suivante du pipeline.

> **Note**
>
> Ces instructions supposent que vous avez déjà réalisé les étapes dans Commencer avec CodePipeline.

Rubriques

- Création d'une action personnalisée
- Création d'un exécutant de tâches pour l'action personnalisée
- Ajout d'une action personnalisée à un pipeline

Création d'une action personnalisée

Pour créer une action personnalisée avec AWS CLI

1. Ouvrez un éditeur de texte et créez un fichier JSON pour votre action personnalisée qui inclut la catégorie de l'action, le fournisseur de l'action et tous les paramètres requis par l'action personnalisée. Par exemple, pour créer une action de génération personnalisée qui ne nécessite qu'une seule propriété, votre fichier JSON devrait ressembler à ceci :

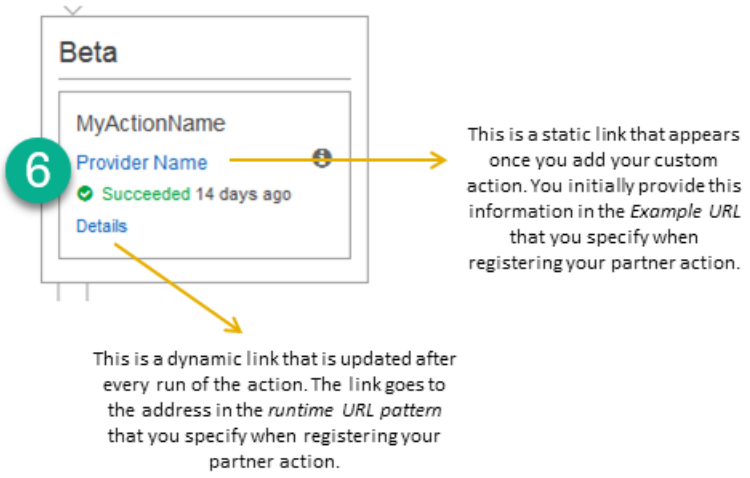
```
{
    "category": "Build",
    "provider": "My-Build-Provider-Name",
    "version": "1",
    "settings": {
        "entityUrlTemplate": "https://my-build-instance/job/{Config:ProjectName}/",
        "executionUrlTemplate": "https://my-build-instance/job/
{Config:ProjectName}/lastSuccessfulBuild/{ExternalExecutionId}/"
    },
    "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"
    }]
}
```

Cet exemple ajoute le balisage à l'action personnalisée en incluant la clé de balise `Project` et la valeur `ProjectA` sur l'action personnalisée. Pour plus d'informations sur le balisage des ressources CodePipeline, consultezBalisage des ressources.

Deux propriétés sont incluses dans le fichier JSON, `entityUrlTemplate` et `executionUrlTemplate`. Vous pouvez désigner un nom dans les propriétés de configuration de l'action personnalisée dans les modèles d'URL, en suivant le format de `{Config:`*`name`*`}`, tant que la propriété de configuration est à la fois obligatoire et non secrète. Par exemple, dans l'exemple ci-dessus, la `entityUrlTemplate` valeur fait référence à la propriété de configuration*`ProjectName`*.

- `entityUrlTemplate` : le lien statique qui fournit des informations sur le fournisseur de services pour l'action. Dans cet exemple, le système de génération comprend un lien statique pour chaque projet de génération. Le format du lien varie en fonction de votre fournisseur de génération (ou, si vous créez un type d'action différent, comme un test, l'autre fournisseur de services). Vous devez fournir ce format de lien de sorte à ce que lorsque l'action personnalisée est ajoutée, l'utilisateur puisse choisir ce lien pour ouvrir un navigateur sur une page de votre site web qui fournit les détails concernant le projet de génération (ou l'environnement de test).
- `executionUrlTemplate` : le lien dynamique qui sera mis à jour avec des informations sur l'exécution actuelle ou la plus récente de l'action. Lorsque votre assistant de tâches personnalisé met à jour l'état d'une tâche (par exemple, réussite, échec, ou en cours), il fournit également un `externalExecutionId` qui sera utilisé pour finaliser le lien. Ce lien peut être utilisé pour fournir les détails relatifs à l'exécution d'une action.

Par exemple, lorsque vous consultez l'action dans le pipeline, vous voyez les deux liens suivants :



1

Ce lien statique s'affiche lorsque vous avez ajouté votre action personnalisée et pointe vers l'adresse dans `entityUrlTemplate`, que vous spécifiez lorsque vous créez votre action personnalisée.

2

Ce lien dynamique est mis à jour après chaque exécution de l'action et pointe vers l'adresse dans `executionUrlTemplate`, que vous spécifiez lorsque vous créez votre action personnalisée.

Pour plus d'informations sur ces types de liens, ainsi que sur `RevisionURLTemplate` et`ThirdPartyURL`, consultez ActionTypeSettings et CreateCustomActionType dans la référence de l'CodePipeline API. Pour plus d'informations sur les exigences en termes de structure de l'action et sur la manière de créer une action, consultez CodePipeline référence de la structure du pipeline.

2. Enregistrez le fichier JSON et donnez-lui un nom facile à mémoriser (par exemple, *`MyCustomAction`* .json).
3. Ouvrez une session de terminal (Linux, OS X, Unix) ou une invite de commande (Windows) sur un ordinateur où vous avez installé l' AWS CLI.
4. Utilisez AWS CLI pour exécuter la aws codepipeline create-custom-action-type commande, en spécifiant le nom du fichier JSON que vous venez de créer.

Par exemple, pour créer une action personnalisée de création :

> **Important**
>
> N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

```
aws codepipeline create-custom-action-type --cli-input-json
 file://MyCustomAction.json
```

5. Cette commande affiche l'intégralité de la structure de l'action personnalisée que vous avez créée, ainsi que la `JobList` propriété de la configuration de l'action, qui est ajoutée automatiquement. Lorsque vous ajoutez l'action personnalisée à un pipeline, vous pouvez utiliser `JobList` pour spécifier les projets du fournisseur, que vous pouvez solliciter pour les tâches. Si vous ne configurez pas ce réglage, toutes les tâches disponibles s'afficheront lorsque vous rechercherez des tâches avec l'exécutant de tâches personnalisé.

   Par exemple, la commande précédente peut afficher une structure similaire à ce qui suit :

```
{
    "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**
>
> Dans le cadre de la sortie de la create-custom-action-type commande, la `id` section inclut`"owner": "Custom"`. CodePipeline désigne automatiquement les types d'actions personnalisés en `Custom` tant que propriétaire. Cette valeur ne peut pas être attribuée ou modifiée lorsque vous utilisez la commande create-custom-action-type ou la commande update-pipeline.

Création d'un exécutant de tâches pour l'action personnalisée

Les actions personnalisées nécessitent un assistant de travail qui interrogera CodePipeline les demandes de travail pour l'action personnalisée, exécutera la tâche et renverra le résultat du statut à CodePipeline. Cet exécutant de tâches peut se trouver sur n'importe quel ordinateur ou n'importe quelle ressource ayant accès au point de terminaison public pour CodePipeline.

Il existe plusieurs façons de concevoir votre exécutant de tâches. Les sections suivantes fournissent des conseils pratiques sur la conception de votre exécutant de tâches personnalisé dans CodePipeline.

Rubriques

- Choix et configuration d'une stratégie de gestion des autorisations pour votre exécutant de tâches
- Développement d'un exécutant de tâches pour votre action personnalisée
- Architecture d'exécutant de tâches personnalisé et exemples

Choix et configuration d'une stratégie de gestion des autorisations pour votre exécutant de tâches

Pour développer un assistant de travail personnalisé pour votre action personnalisée dans CodePipeline, vous aurez besoin d'une stratégie d'intégration de la gestion des utilisateurs et des autorisations.

La stratégie la plus simple consiste à ajouter l'infrastructure dont vous avez besoin pour votre assistant personnalisé en créant des instances Amazon EC2 dotées d'un rôle d'instance IAM, ce qui vous permet d'augmenter facilement les ressources dont vous avez besoin pour votre intégration. Vous pouvez utiliser l'intégration intégrée AWS pour simplifier l'interaction entre votre assistant personnalisé et CodePipeline.

Pour configurer des instances Amazon EC2

1. Apprenez-en plus sur Amazon EC2 et déterminez s'il s'agit du bon choix pour votre intégration. Pour plus d'informations, consultez Amazon EC2 - Hébergement de serveurs virtuels.
2. Commencez à créer vos instances Amazon EC2. Pour plus d'informations, consultez la section Démarrage avec les instances Linux Amazon EC2.

Une autre stratégie à envisager consiste à utiliser la fédération d'identités avec IAM pour intégrer le système et les ressources de votre fournisseur d'identité existants. Cette stratégie est particulièrement utile si vous disposez déjà d'un fournisseur d'identité d'entreprise ou d'une configuration permettant de prendre en charge les utilisateurs à l'aide de fournisseurs d'identité web. La fédération d'identités vous permet d'accorder un accès sécurisé aux AWS ressources CodePipeline, notamment sans avoir à créer ou à gérer des utilisateurs IAM. Vous pouvez utiliser ces fonctionnalités et ces stratégies pour mettre en place des mots de passe à des fins de sécurité et pour créer une rotation des informations d'identification. Vous pouvez vous appuyer sur des modèles d'application pour créer votre propre modèle.

Mise en place de la fédération d'identité

1. En savoir plus sur la fédération d'identités IAM. Pour plus d'informations, consultez Gestion de fédération.
2. Passez en revue les exemples fournis dans Scénarios d'attribution d'accès temporaire afin d'identifier le scénario d'accès temporaire qui correspond le mieux aux besoins de votre action personnalisée.

3. Passez en revue les exemples de code de fédération d'identité appropriés à votre infrastructure, tels que :

   - Exemple d'application de fédération d'identité pour un cas d'utilisation d'Active Directory

4. Commencez à configurer la fédération d'identité. Pour plus d'informations, consultez le Guide de l'utilisateur d'IAM consacré aux fournisseurs d'identité et à la fédération.

Créez l'un des outils suivants à utiliser sous votre nom Compte AWS lors de l'exécution de votre action personnalisée et de votre assistant de travail.

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec des AWS personnes extérieures au Console de gestion AWS. La manière d'accorder l'accès par programmation dépend du type d'utilisateur qui accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
IAM	(Recommandé) Utilisez les informations d'identification de la console comme informations d'identification temporaires pour signer les AWS CLI demandes programmatiques adressées aux AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour cela AWS CLI, consultez la section Connexion pour le développement AWS local dans le Guide de AWS Command Line Interface l'utilisateur. • Pour les AWS kits de développement, consultez la section Connexion pour le développement AWS local dans le Guide de référence AWS des kits de développement logiciel et des outils.

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour cela AWS CLI, reportez-vous AWS CLI à la section Configuration de l'utilisation AWS IAM Identity Center dans le Guide de AWS Command Line Interface l'utilisateur. • Pour les AWS kits de développement logiciel, les outils et les AWS API, consultez la section Authentification IAM Identity Center dans le Guide de référence AWS des kits de développement logiciel et des outils.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de la section Utilisation des informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	À	Méthode
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour cela AWS CLI, consultez la section Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les AWS kits de développement logiciel et les outils, consultez la section Authentification à l'aide d'informations d'identification à long terme dans le Guide de référence AWS des kits de développement logiciel et des outils. • Pour les AWS API, consultez la section Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Voici un exemple de stratégie que vous pouvez créer afin de l'utiliser avec votre exécutant de tâches personnalisé. Cette stratégie n'est fournie qu'à titre d'exemple, et « telle quelle ».

JSON

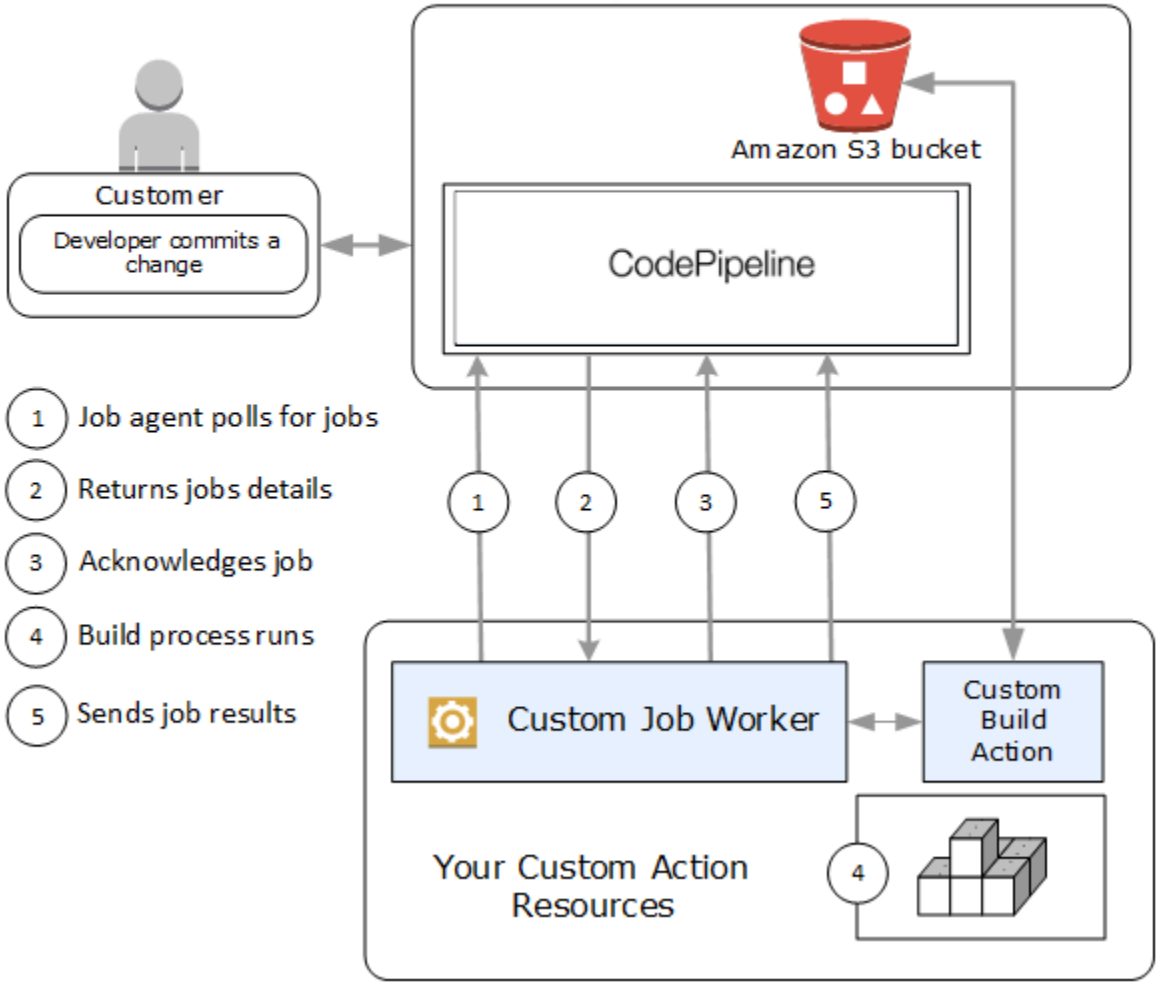
```
{
```

```
  "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**
>
> Envisagez d'utiliser la politique `AWSCodePipelineCustomActionAccess` gérée.

Développement d'un exécutant de tâches pour votre action personnalisée

Après avoir choisi votre stratégie de gestion des autorisations, vous devez réfléchir à la manière dont votre collaborateur interagira avec lui CodePipeline. Le diagramme de haut niveau suivant montre le flux de travail d'une action personnalisée et un assistant de travail pour un processus de création.



1. Vos collaborateurs effectuent des sondages CodePipeline pour les offres d'emploi en utilisant`PollForJobs`.
2. Lorsqu'un pipeline est déclenché suite à une modification apportée à son étape source (par exemple, lorsqu'un développeur valide une modification), le processus de publication automatique commence. Le processus se poursuit jusqu'à l'étape où votre action personnalisée a été configurée. Lorsque vous atteignez votre action à ce stade, une tâche est mise en CodePipeline file d'attente. Cette tâche apparaît si votre exécutant de tâches interroge à nouveau `PollForJobs` pour obtenir l'état. Recueillez les détails de la tâche sur `PollForJobs` et communiquez-les à votre exécutant de tâches.
3. Le travailleur appelle `AcknowledgeJob` pour envoyer CodePipeline un accusé de réception. CodePipeline renvoie un accusé de réception qui indique que le collaborateur doit poursuivre la tâche (`InProgress`), ou, si plusieurs travailleurs recherchent des offres d'emploi et qu'un autre

a déjà réclamé la tâche, une réponse `InvalidNonceException` d'erreur sera renvoyée. Après l'`InProgress`accusé de réception, CodePipeline attend que les résultats soient renvoyés.

4. L'assistant de travail lance votre action personnalisée sur la révision, puis votre action s'exécute. Comme toutes les autres actions, votre action personnalisée renvoie un résultat au collaborateur. Dans l'exemple d'une action de création personnalisée, l'action extrait des artefacts du compartiment Amazon S3, les construit et renvoie les artefacts correctement créés vers le compartiment Amazon S3.
5. Pendant que l'action est en cours d'exécution, le collaborateur peut appeler `PutJobSuccessResult` avec un jeton de continuation (la sérialisation de l'état de la tâche générée par le job worker, par exemple un identifiant de build au format JSON ou une clé d'objet Amazon S3), ainsi que les `ExternalExecutionId` informations qui seront utilisées pour renseigner le lien. `executionUrlTemplate` Cette opération mettra à jour la vue de la console du pipeline, qui contiendra ainsi un lien valide redirigeant vers les détails de l'action en cours. Bien que cela ne soit pas obligatoire, il s'agit d'une bonne pratique car cela permet aux utilisateurs de voir l'état de votre action personnalisée pendant son exécution.

   Une fois que `PutJobSuccessResult` est sollicité, la tâche est considérée comme étant terminée. Une nouvelle tâche est créée dans CodePipeline laquelle le jeton de continuation est inclus. Cette tâche apparaît si votre exécutant de tâches sollicite à nouveau `PollForJobs`. Cette nouvelle tâche permet notamment de vérifier l'état de l'action. Une fois celle-ci terminée, la tâche affiche un résultat avec ou sans jeton de poursuite.

   > **Note**
   >
   > Si votre exécutant de tâches effectue toutes les tâches associées à une action personnalisée, envisagez de diviser le traitement de votre exécutant de tâches en deux étapes minimum. La première étape consiste à établir la page des détails de votre action. Une fois que vous aurez créé la page des détails, vous pourrez sérialiser l'état de l'exécutant de tâches et le renvoyer sous la forme d'un jeton de poursuite en tenant compte des restrictions de taille (voir Quotas dans AWS CodePipeline). Par exemple, vous pouvez écrire l'état de l'action dans la chaîne que vous utilisez comme jeton de poursuite. La seconde étape (et les étapes suivantes) du traitement de votre exécutant de tâches réalise la tâche réelle de l'action. La dernière étape renvoie la réussite ou l'échec à CodePipeline, sans aucun jeton de continuation sur la dernière étape.

Pour plus d'informations sur l'utilisation du jeton de poursuite, consultez les spécifications de `PutJobSuccessResult` dans la Référence de l'API CodePipeline .

6. Une fois l'action personnalisée terminée, le job worker renvoie le résultat de l'action personnalisée à CodePipeline en appelant l'une des deux API suivantes :
   - `PutJobSuccessResult`sans jeton de continuation, ce qui indique que l'action personnalisée a été exécutée avec succès
   - `PutJobFailureResult`, ce qui indique que l'action personnalisée n'a pas été exécutée correctement

   Selon le résultat, le pipeline passera à l'action suivante (réussite) ou s'interrompra (échec).

Architecture d'exécutant de tâches personnalisé et exemples

Une fois que vous avez cartographié votre workflow général, vous pouvez créer votre exécutant de tâches. Bien que les détails de l'action personnalisée détermineront en dernier lieu ce qui est nécessaire pour votre exécutant de tâches, la plupart des exécutants de tâches des actions personnalisées incluent les fonctionnalités suivantes :

- Sondage pour les offres d'emploi liées à CodePipeline l'utilisation`PollForJobs`.
- Reconnaître les emplois et renvoyer les résultats à CodePipeline l'utilisation de `AcknowledgeJob`PutJobSuccessResult, et`PutJobFailureResult`.
- Récupération d'artefacts en les and/or plaçant dans le compartiment Amazon S3 pour le pipeline. Pour télécharger des artefacts depuis le compartiment Amazon S3, vous devez créer un client Amazon S3 qui utilise la signature Signature Version 4 (Sig V4). Le sig V4 est requis pour AWS KMS.

  Pour télécharger des artefacts dans le compartiment Amazon S3, vous devez également configurer la `PutObject` demande Amazon S3 pour qu'elle utilise le chiffrement. Actuellement, seul le service de gestion des AWS clés (AWS KMS) est pris en charge pour le chiffrement. AWS KMS utilisations AWS KMS keys. Afin de savoir s'il convient d'utiliser une clé gérée par le client Clé gérée par AWS ou une clé gérée par le client pour télécharger des artefacts, votre assistant personnalisé doit examiner les données de la tâche et vérifier la propriété de la clé de cryptage. Si la propriété est définie, vous devez utiliser cet ID de clé géré par le client lors de la configuration AWS KMS. Si la propriété key est nulle, vous utilisez le Clé gérée par AWS. CodePipeline utilise le Clé gérée par AWS sauf configuration contraire.

Pour un exemple qui montre comment créer les AWS KMS paramètres en Java ou .NET, consultez la section Spécifier les paramètres AWS Key Management Service dans Amazon S3 à l'aide AWS des kits SDK. Pour plus d'informations sur le compartiment Amazon S3 pour CodePipeline, consultezCodePipeline concepts.

Un exemple plus complexe de travailleur personnalisé est disponible sur GitHub. Ce modèle est en open source et est fourni tel quel.

- Exemple de Job Worker pour CodePipeline : Téléchargez l'exemple depuis le GitHub référentiel.

Ajout d'une action personnalisée à un pipeline

Une fois que vous avez un collaborateur, vous pouvez ajouter votre action personnalisée à un pipeline en en créant une nouvelle et en la choisissant lorsque vous utilisez l'assistant de création de pipeline, en modifiant un pipeline existant et en ajoutant l'action personnalisée AWS CLI, ou en utilisant les SDK ou les API.

> **Note**
>
> Vous pouvez créer un pipeline dans l'assistant Create Pipeline qui inclut une action personnalisée s'il s'agit d'une action de génération ou de déploiement. Si votre action personnalisée est dans la catégorie test, vous devez l'ajouter en modifiant un pipeline existant.

Rubriques

- Ajout d'une action personnalisée à un pipeline existant (interface de ligne de commande)

Ajout d'une action personnalisée à un pipeline existant (interface de ligne de commande)

Vous pouvez utiliser le AWS CLI pour ajouter une action personnalisée à un pipeline existant.

1. Ouvrez une session de terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et exécutez la get-pipeline commande pour copier la structure du pipeline que vous souhaitez

modifier dans un fichier JSON. Par exemple, pour un pipeline nommé **MyFirstPipeline**, saisissez la commande suivante :

```
aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
```

Cette commande ne renvoie rien, mais le fichier que vous avez créé doit apparaître dans le répertoire où vous avez exécuté la commande.

2. Ouvrez le fichier JSON dans un éditeur de texte et modifiez la structure du fichier pour ajouter votre action personnalisée à une étape existante.

> **Note**
>
> Si vous souhaitez que votre action soit exécutée en parallèle d'une autre action de l'étape, assurez-vous que vous lui attribuez la même valeur `runOrder` que cette action.

Par exemple, pour modifier la structure d'un pipeline afin d'ajouter une étape nommée Génération et ajouter ensuite une action personnalisée de génération à cette même étape, vous pouvez modifier le JSON pour ajouter l'étape Génération avant une étape de déploiement, en procédant comme suit :

```
,
    {
      "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. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline d'une manière similaire à l'exemple suivant :

> **Important**
>
> N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

Cette commande affiche toute la structure du pipeline mise à jour.

4. Ouvrez la CodePipeline console et choisissez le nom du pipeline que vous venez de modifier.

   Le pipeline affiche vos modifications. La prochaine fois que vous apporterez une modification à l'emplacement source, le pipeline exécutera cette modification via sa structure révisée.

Marquer une action personnalisée dans CodePipeline

Les balises sont des paires clé-valeur associées à des AWS ressources. Vous pouvez utiliser la console ou l'interface de ligne de commande pour appliquer des balises à vos actions personnalisées dans CodePipeline. Pour plus d'informations sur le balisage CodePipeline des ressources, les cas d'utilisation, les contraintes de clé et de valeur de balise et les types de ressources pris en charge, consultezBalisage des ressources.

Vous pouvez ajouter, supprimer et mettre à jour les valeurs des balises dans une action personnalisée. Vous pouvez ajouter jusqu'à 50 balises à chaque action personnalisée.

Rubriques

- Ajout de balises à une action personnalisée
- Affichage des balises pour une action personnalisée
- Modification des balises d'une action personnalisée
- Suppression de balises d'une action personnalisée

Ajout de balises à une action personnalisée

Suivez ces étapes pour utiliser le AWS CLI pour ajouter une balise à une action personnalisée. Pour ajouter une balise à une action personnalisée lors de sa création, veuillez consulter Créez et ajoutez une action personnalisée dans CodePipeline.

Dans ces étapes, nous supposons que vous avez déjà installé une version récente de l' AWS CLI ou que vous avez procédé à une mise à jour vers la version actuelle. Pour plus d'informations, consultez Installing the AWS Command Line Interface (Installation de).

Depuis le terminal ou la ligne de commande, exécutez la commande tag-resource, en spécifiant l'ARN (Amazon Resource Name) de l'action personnalisée dans laquelle vous souhaitez ajouter des balises ainsi que la clé et la valeur de la balise que vous souhaitez ajouter. Vous pouvez ajouter plusieurs balises à une action personnalisée. Par exemple, pour baliser une action personnalisée avec deux balises, une clé de balise nommée *TestActionType* avec la valeur de *UnitTest* balise et une clé de balise nommée *ApplicationName* avec la valeur de balise de *MyApplication* :

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-
id:actiontype:Owner/Category/Provider/Version --tags key=TestActionType,value=UnitTest
 key=ApplicationName,value=MyApplication
```

Si elle aboutit, cette commande ne renvoie rien.

Affichage des balises pour une action personnalisée

Suivez ces étapes pour utiliser le AWS CLI pour afficher les AWS balises d'une action personnalisée. Si aucune balise n'a été ajoutée, la liste renvoyée est vide.

Depuis le terminal ou la ligne de commande, exécutez la commande list-tags-for-resource. Par exemple, pour afficher une liste des clés et des valeurs de balise pour une action personnalisée avec l'ARN `arn:aws:codepipeline:us-west-2:account-id:actiontype:Owner/Category/Provider/Version` :

```
aws codepipeline list-tags-for-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:actiontype:Owner/Category/Provider/Version
```

Si elle aboutit, cette commande renvoie des informations similaires à ce qui suit :

```
{
```

```
    "tags": {
        "TestActionType": "UnitTest",
        "ApplicationName": "MyApplication"
    }
}
```

Modification des balises d'une action personnalisée

Suivez ces étapes pour utiliser le AWS CLI pour modifier une balise pour une action personnalisée. Vous pouvez modifier la valeur d'une clé existante ou ajouter une autre clé. Vous pouvez également supprimer des balises d'une action personnalisée, comme indiqué dans la section suivante.

Depuis le terminal ou la ligne de commande, exécutez la commande tag-resource, en spécifiant l'ARN (Amazon Resource Name) de l'action personnalisée dans laquelle vous souhaitez mettre à jour une balise et spécifiez la clé de balise et la valeur de balise :

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:actiontype:Owner/Category/Provider/Version --tags
 key=TestActionType,value=IntegrationTest
```

Suppression de balises d'une action personnalisée

Suivez ces étapes pour AWS CLI supprimer un tag d'une action personnalisée. Lorsque vous supprimez des balises de la ressource associée, les balises sont supprimées.

> **Note**
>
> Si vous supprimez une action personnalisée, toutes les associations de balises sont supprimées de l'action personnalisée supprimée. Vous n'avez pas besoin de supprimer les balises avant de supprimer une action personnalisée.

Depuis le terminal ou la ligne de commande, exécutez la commande untag-resource, en spécifiant l'ARN de l'action personnalisée dans laquelle vous souhaitez supprimer des balises et la clé de balise de la balise que vous souhaitez supprimer. Par exemple, pour supprimer une étiquette sur une action personnalisée à l'aide de la clé de balise *TestActionType* :

```
aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-
id:actiontype:Owner/Category/Provider/Version --tag-keys TestActionType
```

Si elle aboutit, cette commande ne renvoie rien. Pour vérifier quelles balises sont associées à l'action personnalisée, exécutez la commande list-tags-for-resource.

Invoquer un AWS Lambda fonction dans un pipeline dans CodePipeline

AWS Lambda est un service informatique qui vous permet d'exécuter un code sans demander la mise en service ou la gestion des serveurs. Vous pouvez créer des fonctions Lambda et les ajouter en tant qu'actions dans vos pipelines. Comme Lambda vous permet d'écrire des fonctions pour effectuer presque toutes les tâches, vous pouvez personnaliser le fonctionnement de votre pipeline.

> **Important**
>
> N'enregistrez pas l'événement JSON CodePipeline envoyé à Lambda car cela peut entraîner la journalisation des informations d'identification de l'utilisateur dans CloudWatch les journaux. Le CodePipeline rôle utilise un événement JSON pour transmettre des informations d'identification temporaires à Lambda `artifactCredentials` sur le terrain. Pour voir un exemple d'événement, consultez la section Exemple d'événement JSON.

Voici quelques manières dont les fonctions Lambda peuvent être utilisées dans les pipelines :

- Créer des ressources à la demande à une étape d'un pipeline en les utilisant CloudFormation et en les supprimant à une autre étape.
- Déployer des versions d'applications sans interruption de service à l' AWS Elastic Beanstalk aide d'une fonction Lambda qui échange les valeurs CNAME.
- Pour déployer sur des instances Amazon ECS Docker.
- Pour sauvegarder les ressources avant une génération ou un déploiement en créant un instantané de l'AMI.
- Pour ajouter l'intégration avec des produits tiers à votre pipeline, comme l'envoi de messages à un client IRC.

> **Note**
>
> La création et l'exécution de fonctions Lambda peuvent entraîner des frais sur votre AWS compte. Pour plus d'informations, consultez Tarification.

Cette rubrique part du principe que vous connaissez AWS CodePipeline AWS Lambda et savez comment créer des pipelines, des fonctions, ainsi que les politiques et rôles IAM dont ils dépendent. Cette rubrique vous montre comment :

- Créez une fonction Lambda qui teste si une page Web a été déployée avec succès.
- Configurez les rôles d'exécution CodePipeline et Lambda et les autorisations requises pour exécuter la fonction dans le cadre du pipeline.
- Modifiez un pipeline pour ajouter la fonction Lambda en tant qu'action.
- Testez l'action en publiant manuellement une modification.

> **Note**
>
> Lorsque vous utilisez l'action d'appel Lambda interrégion dans CodePipeline, le statut de l'exécution Lambda utilisant le PutJobSuccessResult et PutJobFailureResult doit être envoyé à la AWS région dans laquelle la fonction Lambda est présente et non à la région dans laquelle elle existe. CodePipeline

Cette rubrique inclut des exemples de fonctions pour démontrer la flexibilité de l'utilisation des fonctions Lambda dans CodePipeline les domaines suivants :

- Basic Lambda function
  - Création d'une fonction Lambda de base à utiliser avec CodePipeline.
  - Si vous confirmez la réussite ou l'échec CodePipeline , vous accédez au lien Détails de l'action.
- Exemple de fonction Python utilisant un AWS CloudFormation modèle
  - Utilisation des paramètres JSON-encoded utilisateur pour transmettre plusieurs valeurs de configuration à la fonction (`get_user_params`).
  - Interaction avec des artéfacts .zip dans un compartiment d'artéfacts (`get_template`).

- Utilisation d'un jeton de poursuite pour surveiller un processus asynchrone longue durée (`continue_job_later`). Cela permet à l'action de continuer et à la fonction de réussir même si elle dépasse un délai d'exécution de 15 minutes (limite en Lambda).

Chaque modèle de fonction inclut des informations sur les autorisations, que vous devez ajouter au rôle. Pour plus d'informations sur les limites AWS Lambda, consultez la section Limites dans le guide du AWS Lambda développeur.

> **Important**
>
> L'exemple de code, les rôles et les stratégies inclus dans cette rubrique sont fournis à titre d'exemple uniquement et tels quels.

Rubriques

- Étape 1 : Créer un pipeline
- Étape 2 : Création de la fonction Lambda
- Étape 3 : ajouter la fonction Lambda à un pipeline dans la console CodePipeline
- Étape 4 : Testez le pipeline à l'aide de la fonction Lambda
- Étape 5 : étapes suivantes
- Exemple d'événement JSON
- Autres modèles de fonctions

Étape 1 : Créer un pipeline

Au cours de cette étape, vous créez un pipeline auquel vous ajouterez ultérieurement la fonction Lambda. Il s'agit du même pipeline que vous avez créé dans CodePipeline tutoriels. Si ce pipeline est toujours configuré pour votre compte et se trouve dans la même région que celle dans laquelle vous prévoyez de créer la fonction Lambda, vous pouvez ignorer cette étape.

Pour créer le pipeline

1. Suivez les trois premières étapes Didacticiel : Création d'un pipeline simple (compartiment S3) pour créer un compartiment Amazon S3, CodeDeploy des ressources et un pipeline en deux étapes. Choisissez l'option Amazon Linux pour vos types d'instances. Vous pouvez

utiliser le nom de votre choix pour le pipeline, mais dans les étapes de cette rubrique le nom MyLambdaTestPipeline est utilisé.

2. Sur la page d'état de votre pipeline, dans l' CodeDeploy action, choisissez Détails. Sur la page des détails du déploiement pour le groupe de déploiement, choisissez un identifiant d'instance dans la liste.
3. Dans la console Amazon EC2, dans l'onglet Détails de l'instance, copiez l'adresse IP dans Adresse IPv4 publique (par exemple,**192.0.2.4**). Vous utilisez cette adresse comme cible de la fonction dans AWS Lambda.

> **Note**
>
> La politique de rôle de service par défaut pour CodePipeline inclut les autorisations Lambda requises pour appeler la fonction. Toutefois, si vous avez modifié le rôle du service par défaut ou que vous en avez sélectionné un autre, assurez-vous que la stratégie du rôle permet les autorisations `lambda:InvokeFunction` et `lambda:ListFunctions`. Dans le cas contraire, les pipelines qui incluent des actions Lambda échouent.

Étape 2 : Création de la fonction Lambda

Au cours de cette étape, vous allez créer une fonction Lambda qui effectue une requête HTTP et vérifie la présence d'une ligne de texte sur une page Web. Dans le cadre de cette étape, vous devez également créer une politique IAM et un rôle d'exécution Lambda. Pour de plus amples informations, veuillez consulter Modèle d'autorisations dans le Manuel du développeur AWS Lambda .

Pour créer le rôle d'exécution

1. Connectez-vous au Console de gestion AWS et ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Sélectionnez Policies (Politiques), puis Create Policy (Créer une politique). Choisissez l'onglet JSON, puis collez la stratégie suivante dans le champ.

   JSON

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

```
    {
      "Action": [
        "logs:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Action": [
        "codepipeline:PutJobSuccessResult",
        "codepipeline:PutJobFailureResult"
        ],
        "Effect": "Allow",
        "Resource": "*"
     }
  ]
}
```

3. Choisissez Examiner une politique.
4. Sur la page Examiner, dans Nom, tapez un nom pour la stratégie (par exemple, **`CodePipelineLambdaExecPolicy`**). Dans Description, entrez **`Enables Lambda to execute code`**.

   Choisissez Create Policy (Créer une politique).

   > **Note**
   >
   > Il s'agit des autorisations minimales requises pour qu'une fonction Lambda interagisse avec Amazon CodePipeline . CloudWatch Si vous souhaitez étendre cette politique pour autoriser les fonctions qui interagissent avec d'autres AWS ressources, vous devez la modifier pour autoriser les actions requises par ces fonctions Lambda.

5. Sur la page du tableau de bord de la stratégie, choisissez Rôles, puis Créer un rôle .
6. Sur la page Créer un rôle, choisissez Service AWS. Choisissez Lambda, puis Suivant : Autorisations.
7. Sur la page Attacher des stratégies d'autorisation, cochez la case à côté de CodePipelineLambdaExecPolicy, puis choisissez Suivant : Balises. Choisissez Suivant : Vérification.
8. Sur la page Review (Examiner), dans Role name (Nom du rôle), saisissez le nom, puis choisissez Create role (Créer le rôle).

Pour créer l'exemple de fonction Lambda à utiliser avec CodePipeline

1. Connectez-vous à Console de gestion AWS et ouvrez la AWS Lambda console à l'adresse https://console.aws.amazon.com/lambda/.
2. Sur la page Fonctions, choisissez Créer une fonction.

   > **Note**
   >
   > Si vous voyez une page d'accueil au lieu de la page Lambda, choisissez Commencer maintenant.

3. Sur la page Create function, sélectionnez Author from scratch. Dans Nom de la fonction, entrez le nom de votre fonction Lambda (par exemple,**`MyLambdaFunctionForAWSCodePipeline`**). Dans Runtime, choisissez Node.js 20.x.
4. Dans Rôle, sélectionnez Choisissez un rôle existant. Dans Existing role (Rôle existant), choisissez le rôle, puis Create function (Créer la fonction).

   La page de détails de la fonction que vous avez créée s'ouvre.
5. Copiez le code suivant dans la zone Code de fonction :

   > **Note**
   >
   > L'objet de l'événement, situé sous la CodePipeline.job touche, contient les détails de la tâche. Pour un exemple complet de CodePipeline retour de l'événement JSON à Lambda, consultezExemple d'événement 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. Conservez la valeur par défaut Handler (Gestionnaire) et conservez Role (Rôle)**`CodePipelineLambdaExecRole`**.
7. Dans Paramètres de base, pour Délai d'expiration, entrez **`20`** secondes.
8. Choisissez Enregistrer.

Étape 3 : ajouter la fonction Lambda à un pipeline dans la console CodePipeline

Au cours de cette étape, vous ajoutez une nouvelle étape à votre pipeline, puis vous ajoutez une action Lambda qui appelle votre fonction à cette étape.

Pour ajouter une étape

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Sur la page Welcome (Bienvenue), choisissez le pipeline que vous avez créé.
3. Sur la page de la vue du pipeline, choisissez Edit.
4. Sur la page Modifier, choisissez + Ajouter une étape pour ajouter une étape après l'étape de déploiement avec l' CodeDeploy action. Saisissez un nom pour l'étape (par exemple, **`LambdaStage`**), puis choisissez Ajouter une étape.

   > **Note**
   >
   > Vous pouvez également choisir d'ajouter votre action Lambda à une étape existante. À des fins de démonstration, nous ajoutons la fonction Lambda comme seule action d'une étape afin de vous permettre de visualiser facilement sa progression au fur et à mesure que les artefacts progressent dans un pipeline.

5. Choisissez + Ajouter un groupe d'actions. Dans Modifier l'action, dans Nom de l'action, entrez le nom de votre action Lambda (par exemple,**`MyLambdaAction`**). Dans Fournisseur, choisissez AWS Lambda. Dans Nom de la fonction, choisissez ou entrez le nom de votre fonction Lambda (par exemple,**`MyLambdaFunctionForAWSCodePipeline`**). Dans Paramètres utilisateur, spécifiez l'adresse IP de l'instance Amazon EC2 que vous avez copiée précédemment (par exemple**`http://192.0.2.4`**), puis choisissez OK.

> **Note**
>
> Cette rubrique utilise une adresse IP, mais dans un cas réel, vous pourriez renseigner le nom de votre site web à la place (par exemple, **`http://www.example.com`**). Pour plus d'informations sur les données d'événements et les gestionnaires dans AWS Lambda, consultez la section Modèle de programmation dans le Guide du AWS Lambda développeur.

6. Sur la page Modifier l'action, choisissez Enregistrer.

Étape 4 : Testez le pipeline à l'aide de la fonction Lambda

Pour tester la fonction, publiez la modification la plus récente dans le pipeline.

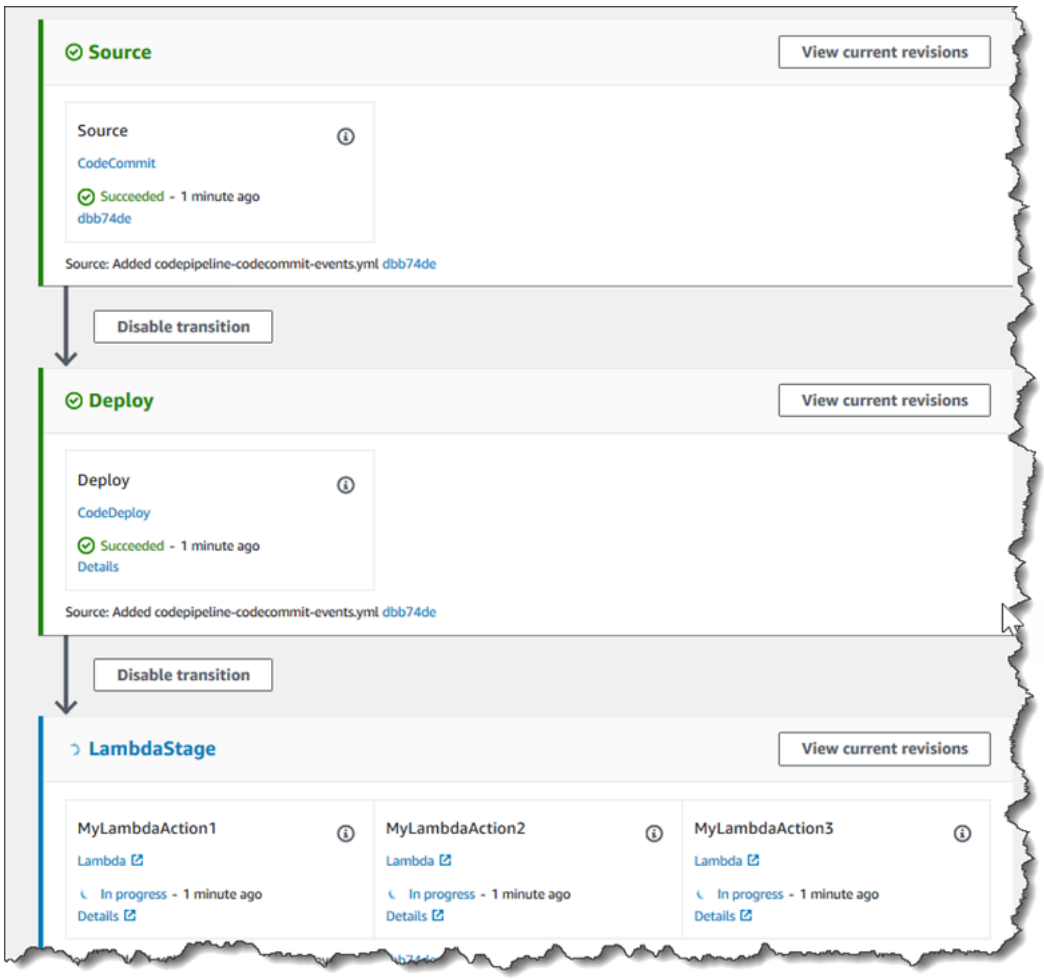
Pour utiliser la console afin d'exécuter la version la plus récente d'un artéfact dans un pipeline

1. Sur la page des détails du pipeline, choisissez Release change. Cette opération exécute la révision la plus récente disponible dans chaque emplacement source spécifié d'une action source à travers le pipeline.
2. Lorsque l'action Lambda est terminée, cliquez sur le lien Détails pour afficher le flux de journal de la fonction sur Amazon CloudWatch, y compris la durée facturée de l'événement. En cas d'échec de la fonction, le CloudWatch journal fournit des informations sur la cause.

Étape 5 : étapes suivantes

Maintenant que vous avez créé une fonction Lambda et que vous l'avez ajoutée en tant qu'action dans un pipeline, vous pouvez essayer ce qui suit :

- Ajoutez d'autres actions Lambda à votre stage pour consulter d'autres pages Web.
- Modifiez la fonction Lambda pour rechercher une chaîne de texte différente.
- Explorez les fonctions Lambda et créez et ajoutez vos propres fonctions Lambda aux pipelines.



Après avoir testé la fonction Lambda, pensez à la supprimer de votre pipeline, à la supprimer et à AWS Lambda supprimer le rôle d'IAM pour éviter d'éventuels frais. Pour plus d'informations, consultez Modifier un pipeline dans CodePipeline, Supprimer un pipeline dans CodePipeline et Suppression des rôles ou des profils d'instance.

Exemple d'événement JSON

L'exemple suivant montre un exemple d'événement JSON envoyé à Lambda par CodePipeline. La structure de cet événement est semblable à la réponse à l'`GetJobDetails API`, mais sans les types de données `actionTypeId` et `pipelineContext`. Deux détails de la configuration de l'action, `FunctionName` et `UserParameters`, sont inclus dans l'événement JSON et dans la réponse à l'API `GetJobDetails`. Les valeurs indiquées *`red italic text`* sont des exemples ou des explications, et non des valeurs réelles.

```
{
    "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"
            }
        }
    }
}
```

Autres modèles de fonctions

Les exemples de fonctions Lambda suivants présentent des fonctionnalités supplémentaires que vous pouvez utiliser pour vos pipelines. CodePipeline Pour utiliser ces fonctions, vous devrez peut-être modifier la politique du rôle d'exécution Lambda, comme indiqué dans l'introduction de chaque exemple.

Rubriques

- Exemple de fonction Python utilisant un AWS CloudFormation modèle

Exemple de fonction Python utilisant un AWS CloudFormation modèle

L'exemple suivant montre une fonction qui crée ou met à jour une pile en fonction d'un CloudFormation modèle fourni. Le modèle crée un compartiment Amazon S3. Il est présenté à des fins de démonstration seulement, pour réduire les coûts. Idéalement, vous devez supprimer la pile avant de charger quoi que ce soit dans le compartiment. Si vous chargez des fichiers dans le compartiment, vous ne pouvez pas supprimer ce dernier lorsque vous supprimez la pile. Vous devrez supprimer manuellement tout ce qui se trouve dans le compartiment avant de pouvoir supprimer le compartiment lui-même.

Cet exemple Python part du principe que vous disposez d'un pipeline qui utilise un compartiment Amazon S3 comme action source, ou que vous avez accès à un compartiment Amazon S3 versionné que vous pouvez utiliser avec le pipeline. Vous créez le CloudFormation modèle, vous le compressez et vous le chargez dans ce compartiment sous forme de fichier .zip. Vous devez ensuite ajouter une action source à votre pipeline qui récupère ce fichier .zip dans le compartiment.

> **Note**
>
> Lorsqu'Amazon S3 est le fournisseur source de votre pipeline, vous pouvez compresser votre ou vos fichiers source dans un seul fichier .zip et charger le fichier .zip dans votre compartiment source. Vous pouvez également charger un seul fichier décompressé ; toutefois, les actions en aval qui attendent un fichier .zip échoueront.

Ce modèle illustre :

- L'utilisation de paramètres JSON-encoded utilisateur pour transmettre plusieurs valeurs de configuration à la fonction (`get_user_params`).
- L'interaction avec des artéfacts .zip dans un compartiment d'artéfacts (`get_template`).
- L'utilisation d'un jeton de poursuite pour surveiller un processus asynchrone longue durée (`continue_job_later`). Cela permet à l'action de continuer et à la fonction de réussir même si elle dépasse un délai d'exécution de 15 minutes (limite en Lambda).

Pour utiliser cet exemple de fonction Lambda, la politique du rôle d'exécution Lambda doit disposer d'`Allow`autorisations dans CloudFormation Amazon S3 et CodePipeline, comme indiqué dans cet exemple de politique :

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": "*"
        }
    ]
}
```

Pour créer le CloudFormation modèle, ouvrez n'importe quel éditeur de texte brut et copiez et collez le code suivant :

```
{
  "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"
    }
  }
}
```

Enregistrez ce dernier sous la forme d'un fichier JSON avec le nom **`template.json`** dans un répertoire nommé **`template-package`**. Créez un fichier compressé (.zip) contenant ce répertoire et nommez-le**`template-package.zip`**, puis chargez le fichier compressé dans un compartiment Amazon S3 versionné. Si vous possédez déjà un compartiment configuré pour votre pipeline, vous pouvez l'utiliser. Ensuite, modifiez votre pipeline pour ajouter une action source qui récupérera le fichier .zip. Donnez un nom à la sortie pour cette action*`MyTemplate`*. Pour de plus amples informations, veuillez consulter Modifier un pipeline dans CodePipeline.

> **Note**
>
> L'exemple de fonction Lambda attend ces noms de fichiers et cette structure compressée. Vous pouvez toutefois remplacer cet exemple par votre propre CloudFormation modèle. Si vous utilisez votre propre modèle, veillez à modifier la politique du rôle d'exécution Lambda afin d'autoriser toute fonctionnalité supplémentaire requise par votre CloudFormation modèle.

Pour ajouter le code suivant en tant que fonction dans Lambda

1. Ouvrez la console Lambda et choisissez Créer une fonction.
2. Sur la page Create function, sélectionnez Author from scratch. Dans Nom de la fonction, entrez le nom de votre fonction Lambda.
3. Dans Runtime, choisissez Python 2.7.
4. Sous Choisir ou créer un rôle d'exécution, sélectionnez Utiliser un rôle existant. Dans Existing role (Rôle existant), choisissez le rôle, puis Create function (Créer la fonction).

   La page de détails de la fonction que vous avez créée s'ouvre.
5. Copiez le code suivant dans la zone Code de fonction :

```
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. Laissez Handler à la valeur par défaut et laissez Role au nom que vous avez sélectionné ou créé précédemment. **`CodePipelineLambdaExecRole`**
7. Dans Paramètres de base, pour Délai d'expiration, remplacez la valeur par défaut de 3 secondes par **`20`**.
8. Choisissez Enregistrer.
9. Depuis la CodePipeline console, modifiez le pipeline pour ajouter la fonction en tant qu'action à une étape de votre pipeline. Choisissez Modifier pour l'étape du pipeline que vous souhaitez modifier, puis choisissez Ajouter un groupe d'actions. Sur la page Modifier l'action, dans Nom de l'action, entrez le nom de votre action. Dans Action provider, choisissez Lambda.

   Sous Artefacts d'entrée, choisissez`MyTemplate`. Dans UserParameters, vous devez fournir une chaîne JSON avec trois paramètres :

   - Nom de la pile
   - CloudFormation nom du modèle et chemin d'accès au fichier
   - Artefact d'entrée

Utilisez des accolades ({}) et séparez les paramètres à l'aide de virgules. Par exemple, pour créer une pile nommée*MyTestStack*, pour un pipeline contenant l'artefact d'entrée *MyTemplate* UserParameters, entrez : {"stack » : » *MyTestStack* «, "file » :"template-package/template .json », "artifact » : » «}. *MyTemplate*

> **Note**
>
> Même si vous avez spécifié l'artefact d'entrée dans UserParameters, vous devez également spécifier cet artefact d'entrée pour l'action dans Artefacts d'entrée.

10. Enregistrez les modifications apportées au pipeline, puis publiez manuellement une modification pour tester l'action et la fonction Lambda.

Ajouter une action d'approbation manuelle à une étape

Dans AWS CodePipeline, vous pouvez ajouter une action d'approbation à une étape d'un pipeline au point où vous souhaitez que l'exécution du pipeline s'arrête afin qu'une personne disposant des Gestion des identités et des accès AWS autorisations requises puisse approuver ou rejeter l'action.

Si l'action est approuvée, l'exécution du pipeline reprend. Si l'action est rejetée, ou si personne ne l'approuve ou ne la rejette dans les sept jours suivant l'atteinte de l'action et son arrêt par le pipeline, le résultat est identique à un échec de l'action et l'exécution du pipeline ne se poursuit pas.

Vous pouvez utiliser des approbations manuelles pour les raisons suivantes :

- Vous souhaitez que quelqu'un vérifie le code ou une modification avant qu'une telle révision ne soit autorisée à passer à l'étape suivante d'un pipeline.
- Vous souhaitez que quelqu'un procède à un test manuel d'assurance qualité sur la dernière version d'une application, ou bien qu'il vérifie l'intégrité d'un artefact de génération avant de le publier.
- Vous souhaitez que quelqu'un passe en revue un texte, nouveau ou mis à jour, avant sa publication sur un site web professionnel.

Options de configuration pour les actions d'approbation manuelles dans CodePipeline

CodePipeline propose trois options de configuration que vous pouvez utiliser pour informer les approbateurs de l'action d'approbation.

Publier des notifications d'approbation Vous pouvez configurer une action d'approbation pour publier un message dans une rubrique Amazon Simple Notification Service lorsque le pipeline s'arrête au niveau de l'action. Amazon SNS transmet le message à chaque point de terminaison inscrit à la rubrique. Vous devez utiliser un sujet créé dans la même AWS région que le pipeline qui inclura l'action d'approbation. Lorsque vous créez une rubrique, nous vous recommandons de lui donner un nom qui explicite son objectif, dans des formats tels que `MyFirstPipeline-us-east-2-approval`.

Lorsque vous publiez des notifications d'approbation sur des sujets Amazon SNS, vous pouvez choisir parmi des formats tels que les destinataires des e-mails ou des SMS, les files d'attente SQS, les HTTP/HTTPS points de terminaison ou les AWS Lambda fonctions que vous invoquez à l'aide d'Amazon SNS. Pour plus d'informations sur les notifications thématiques Amazon SNS, consultez les rubriques suivantes :

- Qu'est-ce qu'Amazon Simple Notification Service ?
- Créer une rubrique dans Amazon SNS
- Envoi de messages d'Amazon SNS aux files d'attente Amazon SQS
- Abonnement d'une file d'attente à une rubrique Amazon SNS
- Envoi de messages Amazon SNS aux terminaux HTTP/HTTPS
- Invocation des fonctions lambda en utilisant des notifications Amazon SNS

Pour découvrir la structure des données JSON générées pour une notification d'action d'approbation, consultez Format de données JSON pour les notifications d'approbation manuelles dans CodePipeline.

Spécification d'une URL pour la révision Dans le cadre de la configuration de l'action d'approbation, vous pouvez spécifier une adresse URL à réviser. L'URL peut être un lien dirigeant vers une application web que vous souhaitez que les approbateurs testent, ou une page contenant davantage d'informations sur votre demande d'approbation. L'URL est incluse dans la notification publiée dans la rubrique Amazon SNS. Les approbateurs peuvent utiliser la console ou l'interface de ligne de commande pour la visualiser.

Saisie de commentaires pour les approbateurs Lorsque vous créez une action d'approbation, vous pouvez également ajouter des commentaires qui s'affichent à l'attention des utilisateurs recevant les notifications ou visualisant l'action dans la réponse de console ou de l'interface de ligne de commande.

Aucune option de configuration Vous pouvez également décider de ne configurer aucune de ces trois options. Celles-ci peuvent ne pas être nécessaires si par exemple vous informez directement un utilisateur qu'il doit passer en revue une action ou si vous souhaitez simplement interrompre le pipeline tant que vous n'avez pas décidé d'approuver l'action vous-même.

Présentation de la configuration et du flux de travail pour les actions d'approbation dans CodePipeline

Voici une vue d'ensemble de la mise en place et de l'utilisation des approbations manuelles.

1. Vous accordez les autorisations IAM requises pour approuver ou rejeter les actions d'approbation à un ou plusieurs rôles IAM au sein de votre organisation.
2. (Facultatif) Si vous utilisez les notifications Amazon SNS, assurez-vous que le rôle de service que vous utilisez dans le cadre de vos CodePipeline opérations est autorisé à accéder aux ressources Amazon SNS.
3. (Facultatif) Si vous utilisez les notifications Amazon SNS, vous créez une rubrique Amazon SNS et vous y ajoutez un ou plusieurs abonnés ou points de terminaison.
4. Lorsque vous utilisez la AWS CLI pour créer le pipeline ou après avoir utilisé la CLI ou la console pour créer le pipeline, vous ajoutez une action d'approbation à une étape du pipeline.

   Si vous utilisez des notifications, vous devez inclure le nom de ressource Amazon (ARN) de la rubrique Amazon SNS dans la configuration de l'action. (Un ARN est un identifiant unique pour une ressource Amazon. Les sujets ARN pour Amazon SNS sont structurés de la manière suivante :. *arn:aws:sns:us-east-2:80398EXAMPLE:MyApprovalTopic* Pour plus d'informations, consultez Amazon Resource Names (ARN) et Service AWS namespaces dans le Référence générale d'Amazon Web Services.)
5. Le pipeline s'interrompt lorsqu'il atteint l'action d'approbation. Si un ARN de rubrique Amazon SNS a été inclus dans la configuration de l'action, une notification est publiée dans la rubrique Amazon SNS et un message est envoyé à tous les abonnés à la rubrique ou aux points de terminaison abonnés, avec un lien permettant de consulter l'action d'approbation dans la console.
6. Un approbateur examine l'URL cible et passe en revue les commentaires, le cas échéant.

7. L'approbateur utilise la console, l'interface de ligne de commande ou le kit de développement logiciel pour fournir un commentaire récapitulatif et envoyer une réponse:
   - Approbation : reprise de l'exécution du pipeline.
   - Rejet : le statut de l'étape devient « Échec » et l'exécution du pipeline est interrompue.

   Si aucune réponse n'est envoyée dans les sept jours, l'action est marquée en tant que « Échec ».

Accorder des autorisations d'approbation à un utilisateur IAM dans CodePipeline

Avant que les utilisateurs IAM de votre organisation puissent approuver ou rejeter les actions d'approbation, ils doivent être autorisés à accéder aux pipelines et à mettre à jour le statut des actions d'approbation. Vous pouvez autoriser l'accès à tous les pipelines et à toutes les actions d'approbation de votre compte en associant la politique `AWSCodePipelineApproverAccess` gérée à un utilisateur, un rôle ou un groupe IAM ; vous pouvez également accorder des autorisations limitées en spécifiant les ressources individuelles auxquelles un utilisateur, un rôle ou un groupe IAM peuvent accéder.

> **Note**
>
> Les autorisations décrites dans cette rubrique offrent un accès très limité. Si vous souhaitez qu'un utilisateur, un rôle ou un groupe puisse faire plus que de simplement approuver ou rejeter des actions d'approbation, vous pouvez joindre d'autres stratégies gérées. Pour plus d'informations sur les politiques gérées disponibles pour CodePipeline, consultezAWS politiques gérées pour AWS CodePipeline.

Octroi de l'autorisation d'approbation à tous les pipelines et à toutes les actions d'approbation

Pour les utilisateurs qui doivent effectuer des actions d'approbation dans CodePipeline, utilisez la politique `AWSCodePipelineApproverAccess` gérée.

Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

Créez un jeu d'autorisations. Suivez les instructions de la rubrique Création d'un jeu d'autorisations du Guide de l'utilisateur AWS IAM Identity Center .

- Utilisateurs gérés dans IAM par un fournisseur d'identité :

  Créez un rôle pour la fédération d'identité. Suivez les instructions de la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM.

- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM dans le Guide de l'utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

Spécification d'une autorisation d'approbation pour les pipelines et les actions d'approbation spécifiques

Pour les utilisateurs qui doivent effectuer des actions d'approbation dans CodePipeline, utilisez la politique personnalisée suivante. Dans la politique ci-dessous, spécifiez les ressources individuelles auxquelles un utilisateur peut accéder. Par exemple, la politique suivante accorde aux utilisateurs le pouvoir d'approuver ou de rejeter uniquement l'action nommée `MyApprovalAction` dans le `MyFirstPipeline` pipeline dans la région USA Est (Ohio) (us-east-2) :

> **Note**
>
> L'`codepipeline:ListPipelines`autorisation n'est requise que si les utilisateurs IAM doivent accéder au CodePipeline tableau de bord pour afficher cette liste de pipelines. Si l'accès à la console n'est pas obligatoire, vous pouvez ignorer `codepipeline:ListPipelines`.

Pour utiliser l'éditeur de politique JSON afin de créer une politique

1. Connectez-vous au Console de gestion AWS et ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le panneau de navigation de gauche, sélectionnez Policies (Politiques).

Si vous sélectionnez Politiques pour la première fois, la page Bienvenue dans les politiques gérées s'affiche. Sélectionnez Mise en route.

3. En haut de la page, sélectionnez Créer une politique.
4. Dans la section Éditeur de politique, choisissez l'option JSON.
5. Entrez le document de politique JSON suivant :

```
 {
    "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. Choisissez Suivant.

> **Note**
>
> Vous pouvez basculer à tout moment entre les options des éditeurs visuel et JSON. Toutefois, si vous apportez des modifications ou si vous choisissez Suivant dans l'éditeur visuel, IAM peut restructurer votre politique afin de l'optimiser pour l'éditeur visuel. Pour plus d'informations, consultez la page Restructuration de politique dans le Guide de l'utilisateur IAM.

7. Sur la page Vérifier et créer, saisissez un Nom de politique et une Description (facultative) pour la politique que vous créez. Vérifiez les Autorisations définies dans cette politique pour voir les autorisations accordées par votre politique.
8. Choisissez Create policy (Créer une politique) pour enregistrer votre nouvelle politique.

Accorder des autorisations Amazon SNS à un rôle CodePipeline de service

Si vous envisagez d'utiliser Amazon SNS pour publier des notifications sur des sujets lorsque des actions d'approbation nécessitent une révision, le rôle de service que vous utilisez dans vos CodePipeline opérations doit être autorisé à accéder aux ressources Amazon SNS. Vous pouvez utiliser la console IAM pour ajouter cette autorisation à votre rôle de service.

Dans la politique ci-dessous, spécifiez la politique de publication avec SNS. Vous pouvez donner un nom à la politique suivante`SNSPublish`. Utilisez la politique suivante en l'associant à votre rôle de service.

> **Important**
>
> Assurez-vous d'être connecté Console de gestion AWS avec les mêmes informations de compte que celles que vous avez utiliséesCommencer avec CodePipeline.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "sns:Publish",
            "Resource": "*"
        }
    ]
}
```

Pour utiliser l'éditeur de politique JSON afin de créer une politique

1. Connectez-vous au Console de gestion AWS et ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le panneau de navigation de gauche, sélectionnez Policies (Politiques).

   Si vous sélectionnez Politiques pour la première fois, la page Bienvenue dans les politiques gérées s'affiche. Sélectionnez Mise en route.
3. En haut de la page, sélectionnez Créer une politique.
4. Dans la section Éditeur de politique, choisissez l'option JSON.
5. Saisissez ou collez un document de politique JSON. Pour de plus amples informations sur le langage de la stratégie IAM, consultez la référence de politique JSON IAM.
6. Résolvez les avertissements de sécurité, les erreurs ou les avertissements généraux générés durant la validation de la politique, puis choisissez Suivant.

   > **Note**
   >
   > Vous pouvez basculer à tout moment entre les options des éditeurs visuel et JSON. Toutefois, si vous apportez des modifications ou si vous choisissez Suivant dans l'éditeur visuel, IAM peut restructurer votre politique afin de l'optimiser pour l'éditeur visuel. Pour plus d'informations, consultez la page Restructuration de politique dans le Guide de l'utilisateur IAM.
7. (Facultatif) Lorsque vous créez ou modifiez une politique dans le Console de gestion AWS, vous pouvez générer un modèle de stratégie JSON ou YAML que vous pouvez utiliser dans les CloudFormation modèles.

   Pour ce faire, dans l'éditeur de politiques, choisissez Actions, puis sélectionnez Générer un CloudFormation modèle. Pour en savoir plus CloudFormation, consultez la référence Gestion des identités et des accès AWS des types de ressources dans le Guide de AWS CloudFormation l'utilisateur.

8. Lorsque vous avez fini d'ajouter des autorisations à la politique, choisissez Suivant.
9. Sur la page Vérifier et créer, tapez un Nom de politique et une Description (facultative) pour la politique que vous créez. Vérifiez les Autorisations définies dans cette politique pour voir les autorisations accordées par votre politique.
10. (Facultatif) Ajoutez des métadonnées à la politique en associant les balises sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la section Balises pour les Gestion des identités et des accès AWS ressources dans le Guide de l'utilisateur IAM.
11. Choisissez Create policy (Créer une politique) pour enregistrer votre nouvelle politique.

Ajoutez une action d'approbation manuelle à un pipeline dans CodePipeline

Vous pouvez ajouter une action d'approbation à une étape d'un CodePipeline pipeline au point où vous souhaitez que le pipeline s'arrête afin que quelqu'un puisse approuver ou rejeter manuellement l'action.

> **Note**
>
> Les actions d'approbation ne peuvent pas être ajoutées aux étapes Source. Les étapes Source peuvent uniquement contenir des actions source.

Si vous souhaitez utiliser Amazon SNS pour envoyer des notifications lorsqu'une action d'approbation est prête à être révisée, vous devez d'abord remplir les conditions préalables suivantes :

- Autorisez votre rôle CodePipeline de service à accéder aux ressources Amazon SNS. Pour plus d'informations, consultez Accorder des autorisations Amazon SNS à un rôle CodePipeline de service.
- Autorisez une ou plusieurs identités IAM de votre organisation à mettre à jour le statut d'une action d'approbation. Pour plus d'informations, consultez Accorder des autorisations d'approbation à un utilisateur IAM dans CodePipeline.

Dans cet exemple, vous créez une nouvelle étape d'approbation et vous y ajoutez une action d'approbation manuelle. Vous pouvez également ajouter une action d'approbation manuelle à une étape existante qui contient d'autres actions.

Ajouter une action d'approbation manuelle à un CodePipeline pipeline (console)

Vous pouvez utiliser la CodePipeline console pour ajouter une action d'approbation à un CodePipeline pipeline existant. Vous devez utiliser l' AWS interface de ligne de commande si vous souhaitez ajouter des actions d'approbation lorsque vous créez un nouveau pipeline.

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Dans Nom, choisissez le pipeline.
3. Sur la page des détails du pipeline, choisissez Edit.
4. Si vous souhaitez ajouter une action d'approbation à une nouvelle étape, choisissez + Add stage (+ Ajouter une étape) à l'endroit du pipeline où vous voulez ajouter une demande d'approbation et nommez l'étape. Dans la page Add stage (Ajouter une étape), dans Stage name (Nom de l'étape), entrez le nom de la nouvelle étape. Par exemple, ajoutez une nouvelle étape et nommez-la `Manual_Approval`.

   Si vous souhaitez ajouter une action d'approbation à une étape existante, choisissez Modifier une étape.
5. Dans l'étape où vous souhaitez ajouter l'action d'approbation, choisissez + Add action group (+ Ajouter un groupe d'actions).
6. Sur la page Modifier l'action, procédez comme suit :

   1. Dans Action name (Nom de l'action), saisissez un nom pour identifier l'action.
   2. Dans Fournisseur d'action, sous Approbation, choisissez Approbation manuelle.
   3. (Facultatif) Dans ARN de la rubrique SNS, choisissez le nom de la rubrique à utiliser pour envoyer des notifications pour l'action d'approbation.
   4. (Facultatif) Dans URL pour la révision, saisissez l'URL de la page ou de l'application que vous souhaitez que l'approbateur examine. Les approbateurs peuvent accéder à cette URL via un lien inclus dans la vue de la console du pipeline.
   5. (Facultatif) Dans Commentaires, saisissez toute autre information que vous souhaitez partager avec le vérificateur.
   6. Choisissez Enregistrer.

Ajouter une action d'approbation manuelle à un CodePipeline pipeline (CLI)

Vous pouvez utiliser l'interface de ligne de commande pour ajouter une action d'approbation à un pipeline existant, ou lorsque vous en créez un. Pour ce faire, vous devez inclure une action

d'approbation avec le type Approbation manuelle à une étape que vous êtes en train de créer ou de modifier.

Pour plus d'informations sur la création et la modification des pipelines, consultez Création d'un pipeline, d'étapes et d'actions et Modifier un pipeline dans CodePipeline.

Pour ajouter une étape à un pipeline qui inclut uniquement une action d'approbation, vous devez inclure un élément similaire à l'exemple suivant, lors de la création ou de la mise à jour du pipeline.

> **Note**
>
> La section `configuration` est facultative. Il s'agit juste d'une partie, et non pas de toute la structure, du fichier. Pour de plus amples informations, veuillez consulter CodePipeline référence de la structure du pipeline.

```
{
    "name": "MyApprovalStage",
    "actions": [
        {
            "name": "MyApprovalAction",
            "actionTypeId": {
                "category": "Approval",
                "owner": "AWS",
                "version": "1",
                "provider": "Manual"
            },
            "inputArtifacts": [],
            "outputArtifacts": [],
            "configuration": {
                "NotificationArn": "arn:aws:sns:us-
east-2:80398EXAMPLE:MyApprovalTopic",
                "ExternalEntityLink": "http://example.com",
                "CustomData": "The latest changes include feedback from Bob."},
            "runOrder": 1
        }
    ]
}
```

Si l'action d'approbation se trouve dans une étape qui comporte d'autres actions, la section de votre fichier JSON contenant l'étape peut être similaire à l'exemple suivant.

> **Note**
>
> La section `configuration` est facultative. Il s'agit juste d'une partie, et non pas de toute la structure, du fichier. Pour de plus amples informations, veuillez consulter CodePipeline référence de la structure du pipeline.

```
,
{
    "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
        }
    ]
}
```

Approuver ou rejeter une action d'approbation dans CodePipeline

Lorsqu'un pipeline comprend une action d'approbation, l'exécution du pipeline s'interrompt à l'endroit où l'action a été ajoutée. Le pipeline ne reprend que lorsqu'une personne approuve manuellement l'action. Si un approbateur rejette l'action, ou si aucune réponse d'approbation n'est reçue dans les sept jours suivant l'interruption du pipeline en vue d'une action d'approbation, l'état du pipeline devient « Échec ».

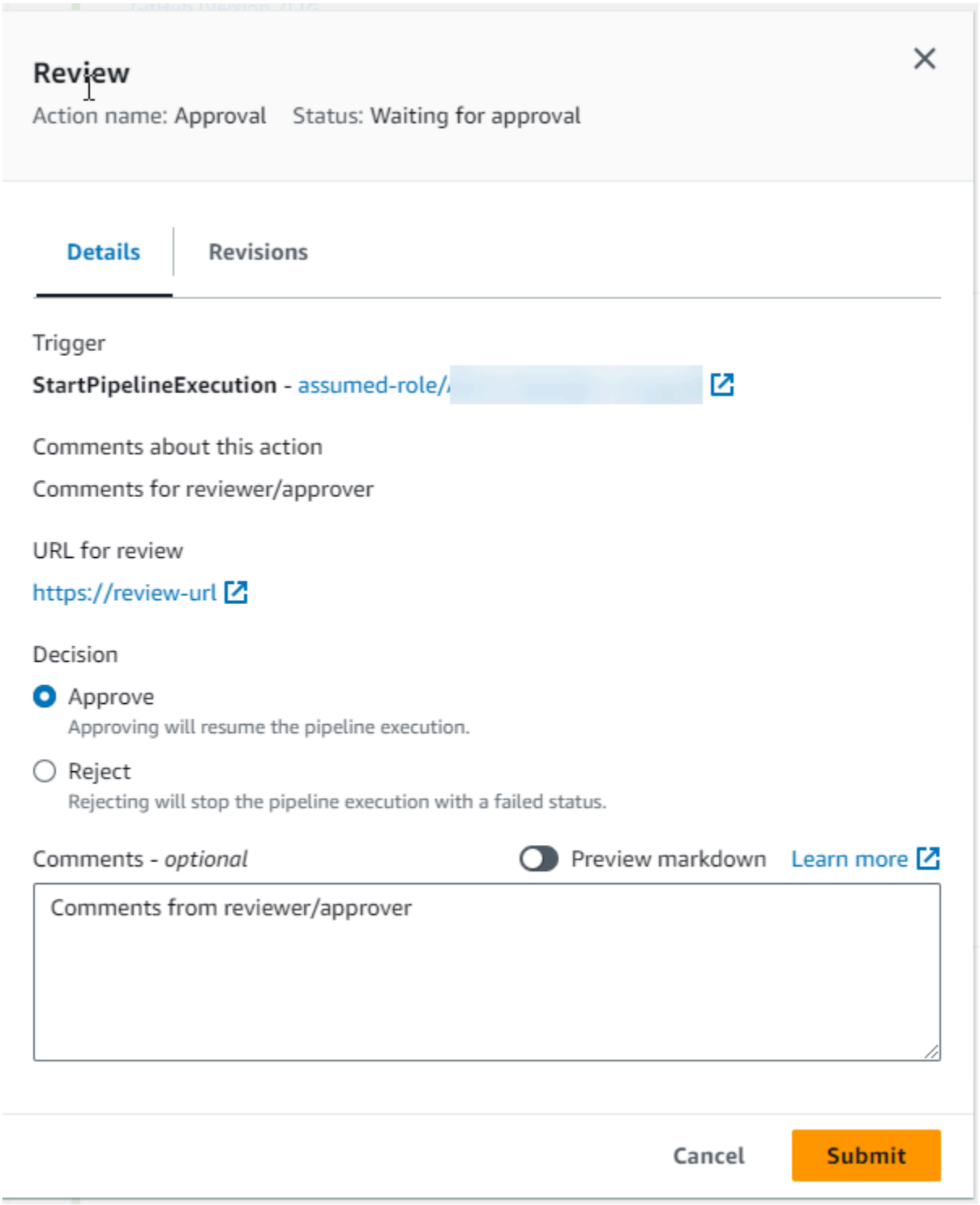
Si la personne qui a ajouté l'action d'approbation au pipeline a configuré des notifications, il est possible que vous receviez un e-mail contenant les informations relatives au pipeline et le statut de l'approbation.

Approbation ou rejet d'une action d'approbation (console)

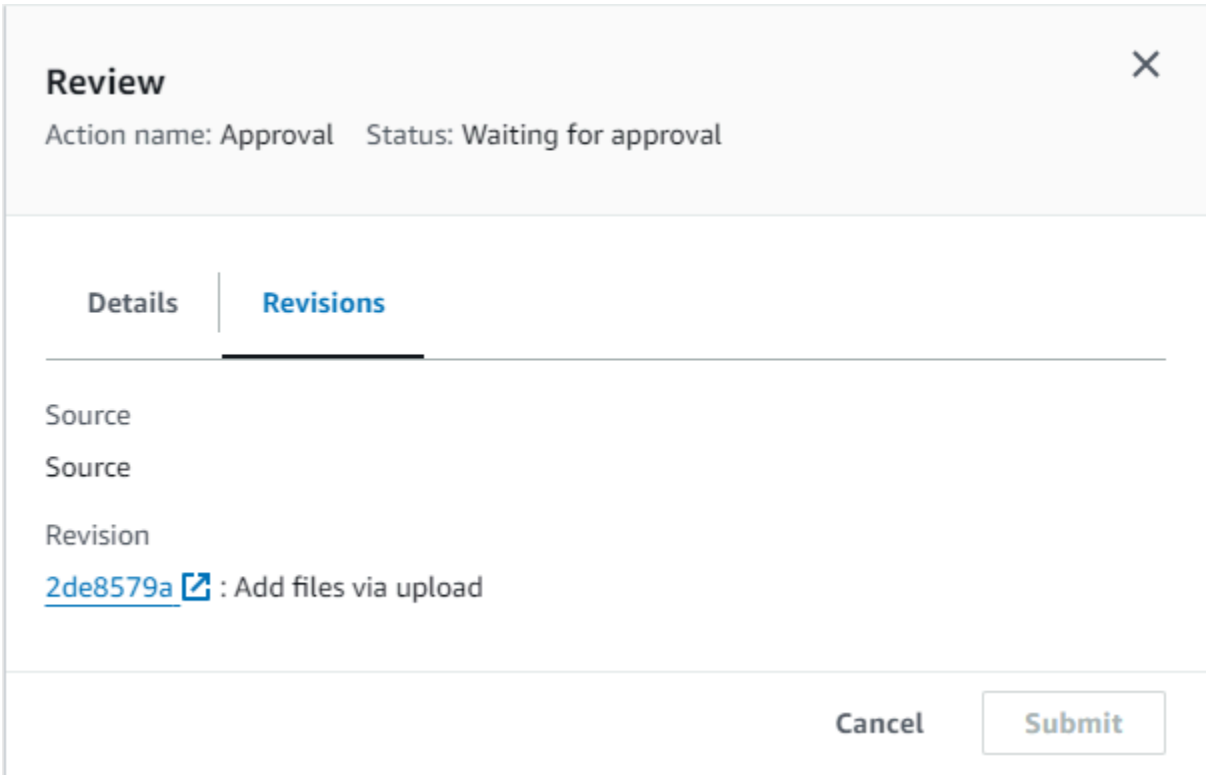
Si vous recevez une notification incluant un lien direct vers une action d'approbation, sélectionnez le lien Approve or reject (Approuver ou refuser) connectez-vous à la console, puis passez à l'étape 7 ici. Sinon, suivez toutes ces étapes.

1. Ouvrez la CodePipeline console à l'adresse https://console.aws.amazon.com/codepipeline/.
2. Sur la page Tous les pipelines, choisissez le nom du pipeline.
3. Recherchez l'étape contenant l'action d'approbation. Choisissez Examiner.

   La boîte de dialogue Révision s'affiche. L'onglet Détails affiche le contenu et les commentaires de l'évaluation.



L'onglet Révisions affiche les révisions sources pour l'exécution.



4. Dans l'onglet Détails, consultez les commentaires et l'URL, le cas échéant. Le message affiche également l'URL du contenu que vous devez passer en revue, le cas échéant.
5. Si une URL a été fournie, choisissez l'URL du lien de révision dans l'action pour ouvrir la page Web cible, puis passez en revue le contenu.
6. Dans la fenêtre Révision, saisissez des commentaires de révision, tels que la raison pour laquelle vous approuvez ou rejetez l'action, puis choisissez Approuver ou Rejeter.
7. Sélectionnez Soumettre.

Approbation ou rejet d'une demande d'approbation (interface de ligne de commande)

Pour utiliser l'interface de ligne de commande afin de répondre à une action d'approbation, vous devez d'abord utiliser la commande get-pipeline-state pour récupérer le jeton associé à la dernière exécution de l'action d'approbation.

1. Sur un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows), exécutez la commande get-pipeline-state sur le pipeline qui contient l'action d'approbation. Par exemple, pour un pipeline nommé *MyFirstPipeline*, saisissez ce qui suit :

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

2. Dans la réponse à la commande, recherchez la valeur `token`, qui apparaît dans `latestExecution` dans la section `actionStates` pour l'action d'approbation, comme indiqué ici :

```
{
    "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. Dans un éditeur de texte brut, créez un fichier où vous ajoutez ce qui suit, au format JSON :

   - Le nom du pipeline qui contient l'action d'approbation.
   - Le nom de l'étape qui contient l'action d'approbation.
   - Le nom de l'action d'approbation.
   - La valeur du jeton que vous avez recueilli lors de l'étape précédente.
   - Votre réponse à l'action (Approuvé ou Rejeté). La réponse doit être en majuscules.
   - Vos commentaires récapitulatifs.

   Pour l'exemple *MyFirstPipeline* précédent, votre fichier ressemblera à ce qui suit :

```
{
```

```
  "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. Enregistrez le fichier avec un nom tel que **`approvalstage-approved.json`**.
5. Exécutez la commande put-approval-result, en précisant le nom du fichier JSON d'approbation, comme suit :

> **Important**
>
> N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

```
aws codepipeline put-approval-result --cli-input-json file://approvalstage-approved.json
```

Format de données JSON pour les notifications d'approbation manuelles dans CodePipeline

Pour les actions d'approbation qui utilisent les notifications Amazon SNS, les données JSON relatives à l'action sont créées et publiées sur Amazon SNS lorsque le pipeline s'arrête. Vous pouvez utiliser la sortie JSON pour envoyer des messages aux files d'attente Amazon SQS ou invoquer des fonctions dans. AWS Lambda

> **Note**
>
> Ce guide ne traite pas de la façon de configurer les notifications à l'aide de JSON. Pour plus d'informations, consultez les sections Envoyer des messages Amazon SNS aux files d'attente Amazon SQS et Invoquer des fonctions Lambda à l'aide des notifications Amazon SNS dans le guide du développeur Amazon SNS.

L'exemple suivant illustre la structure de la sortie JSON disponible avec les approbations 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."
    }
}
```

Ajouter une action interrégionale dans CodePipeline

AWS CodePipeline inclut un certain nombre d'actions qui vous aident à configurer les ressources de création, de test et de déploiement pour votre processus de publication automatique. Vous pouvez ajouter à votre pipeline des actions situées dans une AWS région différente de votre pipeline. Lorsque an Service AWS est le fournisseur d'une action et que ce type/provider type d'action se trouve dans une AWS région différente de celle de votre pipeline, il s'agit d'une action interrégionale.

> **Note**
>
> Cross-region les actions sont prises en charge et ne peuvent être créées que dans les AWS régions où elles CodePipeline sont prises en charge. Pour obtenir la liste des AWS régions prises en charge pour CodePipeline, consultezQuotas dans AWS CodePipeline.

Vous pouvez utiliser la console ou CloudFormation ajouter des actions interrégionales dans les pipelines. AWS CLI

> **Note**
>
> Certains types d'actions CodePipeline peuvent n'être disponibles que dans certaines AWS régions. Notez également qu'il peut y avoir AWS des régions où un type d'action est disponible, mais aucun AWS fournisseur spécifique pour ce type d'action n'est disponible.

Lorsque vous créez ou modifiez un pipeline, vous devez avoir un compartiment d'artefact dans le pipeline Région, puis vous devez disposer d'un compartiment d'artefact par région dans laquelle vous prévoyez d'exécuter une action. Pour plus d'informations sur le paramètre `ArtifactStores`, consultez CodePipeline référence de la structure du pipeline.

> **Note**
>
> CodePipeline gère la copie des artefacts d'une AWS région vers les autres régions lors de l'exécution d'actions interrégionales.

Si vous utilisez la console pour créer un pipeline ou des actions interrégionales, les compartiments d'artefacts par défaut sont configurés CodePipeline dans les régions où vous avez des actions. Lorsque vous utilisez le AWS CLI CloudFormation, ou un SDK pour créer un pipeline ou des actions interrégionales, vous fournissez le compartiment d'artefacts pour chaque région dans laquelle vous avez des actions.

> **Note**
>
> Vous devez créer le compartiment d'artefacts et la clé de chiffrement dans la même AWS région que l'action interrégionale et dans le même compte que votre pipeline.

Vous ne pouvez pas créer d'actions inter-régions pour les types d'action suivants :

- Actions source
- Third-party actions
- Actions personnalisées

> **Note**
>
> Lorsque vous utilisez l'action d'appel Lambda interrégion dans CodePipeline, le statut de l'exécution Lambda utilisant le PutJobSuccessResult et PutJobFailureResult doit être envoyé à la AWS région dans laquelle la fonction Lambda est présente et non à la région dans laquelle elle existe. CodePipeline

Lorsqu'un pipeline inclut une action interrégionale dans le cadre d'une étape, il CodePipeline reproduit uniquement les artefacts d'entrée de l'action interrégionale de la région du pipeline vers la région de l'action.

> **Note**
>
> La région du pipeline et la région dans laquelle vos ressources de détection des changements d' CloudWatch événements sont gérées restent les mêmes. La région dans laquelle votre pipeline est hébergé ne change pas.

Gérer les actions interrégionales dans un pipeline (console)

Vous pouvez utiliser la CodePipeline console pour ajouter une action interrégionale à un pipeline existant. Pour créer un nouveau pipeline avec des actions inter-régions à l'aide de l'assistant Créer un pipeline, consultez Création d'un pipeline personnalisé (console).

Dans la console, vous pouvez créer une action interrégionale dans une étape de pipeline en choisissant le fournisseur d'actions et le champ Région, qui répertorie les ressources que vous avez créées dans cette région pour ce fournisseur. Lorsque vous ajoutez une action interrégionale, CodePipeline utilise un compartiment d'artefacts distinct dans la région de l'action. Pour plus d'informations sur les compartiments d'artefacts inter-régions, consultez CodePipeline référence de la structure du pipeline.

Ajouter une action interrégionale à une étape du pipeline (console)

Utilisez la console pour ajouter une action inter-région à un pipeline.

> **Note**
>
> Si le pipeline est en cours d'exécution lorsque les modifications sont enregistrées, l'exécution ne se termine pas.

Pour ajouter une action inter-région

1. Connectez-vous à la console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Sélectionnez votre pipeline, puis choisissez Modifier.
3. En bas du schéma, choisissez +Ajouter une étape si vous ajoutez une nouvelle étape ou Modifier l'étape si vous voulez ajouter l'action à une étape existante.
4. Dans Modifier : <Étape>, choisissez +Ajouter groupe d'action pour ajouter une action en série. Ou choisissez +Ajouter une action pour ajouter une action en parallèle.
5. Sur la page Modifier l'action :
   a. Dans Nom de l'action, saisissez un nom pour l'action inter-région.
   b. Dans Fournisseur d'action, choisissez le fournisseur d'action.
   c. Dans Région, choisissez la AWS région dans laquelle vous avez créé ou prévoyez de créer la ressource pour l'action. Une fois la région sélectionnée, les ressources disponibles pour cette région sont répertoriées pour la sélection. Le champ Région indique où les AWS ressources sont créées pour ce type d'action et ce type de fournisseur. Ce champ s'affiche uniquement pour les actions dont le fournisseur d'actions est un Service AWS. Par défaut, le champ Région est Région AWS identique à celui de votre pipeline.
   d. Dans Artefact d'entrée, choisissez l'entrée adéquate à partir de l'étape précédente. Par exemple, si l'étape précédente est une étape source, choisissez SourceArtifact.
   e. Complétez tous les champs obligatoires pour le fournisseur d'action que vous configurez.
   f. Dans Artefact de sortie, choisissez la sortie adaptée à la prochaine étape. Par exemple, si l'étape suivante est une étape de déploiement, choisissez BuildArtifact.
   g. Choisissez Enregistrer.
6. Dans Modifier : <Étape>, choisissez Effectué.
7. Choisissez Enregistrer.

Modifier une action interrégionale dans une étape du pipeline (console)

Utilisez la console pour modifier une action inter-région existante dans un pipeline.

> **Note**
>
> Si le pipeline est en cours d'exécution lorsque les modifications sont enregistrées, l'exécution ne se termine pas.

Pour modifier une action inter-région

1. Connectez-vous à la console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Sélectionnez votre pipeline, puis choisissez Modifier.
3. Choisissez Modifier l'étape.
4. Dans Modifier : <Étape>, choisissez l'icône de modification d'une action existante.
5. Sur la page Modifier l'action, modifiez les champs comme il convient.
6. Dans Modifier : <Étape>, choisissez Effectué.
7. Choisissez Enregistrer.

Supprimer une action interrégionale d'une étape du pipeline (console)

Utilisez la console pour supprimer une action inter-région existante dans un pipeline.

> **Note**
>
> Si le pipeline est en cours d'exécution lorsque les modifications sont enregistrées, l'exécution ne se termine pas.

Pour supprimer une action inter-région

1. Connectez-vous à la console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Sélectionnez votre pipeline, puis choisissez Modifier.

3. Choisissez Modifier l'étape.
4. Dans Modifier : <Étape>, choisissez l'icône de suppression d'une action existante.
5. Dans Modifier : <Étape>, choisissez Effectué.
6. Choisissez Enregistrer.

Ajouter une action interrégionale à un pipeline (CLI)

Vous pouvez utiliser le AWS CLI pour ajouter une action interrégionale à un pipeline existant.

Pour créer une action interrégionale dans une phase de pipeline avec le AWS CLI, vous ajoutez l'action de configuration ainsi qu'un `region` champ facultatif. Vous devez également avoir déjà créé un compartiment d'artefacts dans la région de l'action. Au lieu de fournir le paramètre `artifactStore` du pipeline de la région unique, vous utilisez le paramètre `artifactStores` pour inclure une liste de chaque compartiment d'artefact de la région.

> **Note**
>
> Dans cette procédure pas à pas et ses exemples, vous *`RegionA`* trouverez la région dans laquelle le pipeline est créé. Il a accès au compartiment *`RegionA`* Amazon S3 utilisé pour stocker les artefacts du pipeline et au rôle de service utilisé par CodePipeline. *`RegionB`*est la région dans laquelle l' CodeDeploy application, le groupe de déploiement et le rôle de service utilisés par CodeDeploy sont créés.

Conditions préalables

Vous devez avoir créé ce qui suit :

- Un oléoduc dans*`RegionA`*.
- Un compartiment d'artefacts Amazon S3 dans. *`RegionB`*
- Les ressources nécessaires à votre action, telles que votre CodeDeploy application et votre groupe de déploiement pour une action de déploiement interrégionale, se trouvent dans*`RegionB`*.

Ajouter une action interrégionale à un pipeline (CLI)

Utilisez le AWS CLI pour ajouter une action interrégionale à un pipeline.

Pour ajouter une action inter-région

1. Pour un pipeline dans*RegionA*, exécutez la get-pipeline commande pour copier la structure du pipeline dans un fichier JSON. Par exemple, pour un pipeline nommé `MyFirstPipeline`, exécutez la commande suivante :

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Cette commande ne renvoie rien, mais le fichier que vous avez créé doit apparaître dans le répertoire où vous avez exécuté la commande.

2. Ajoutez le champ `region` pour ajouter une nouvelle étape avec votre action inter-région qui inclut la région et les ressources de votre action. L'exemple JSON suivant ajoute une étape de déploiement avec une action de déploiement interrégionale où le fournisseur se trouve CodeDeploy dans une nouvelle région`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. Dans la structure de pipeline, supprimez le champ `artifactStore` et ajoutez la carte `artifactStores` pour votre nouvelle action inter-région. Le mappage doit inclure une entrée pour chaque AWS région dans laquelle vous avez des actions. Pour chaque entrée de la cartographie, les ressources doivent se trouver dans la AWS région correspondante. Dans l'exemple ci-dessous, `ID-A` est l'ID de clé de cryptage pour*RegionA*, et `ID-B` est l'ID de clé de cryptage pour*RegionB*.

```
"artifactStores":{
   "RegionA":{
      "encryptionKey":{
         "id":"ID-A",
         "type":"KMS"
      },
      "location":"Location1",
      "type":"S3"
   },
   "RegionB":{
      "encryptionKey":{
         "id":"ID-B",
         "type":"KMS"
      },
      "location":"Location2",
      "type":"S3"
   }
}
```

L'exemple JSON suivant présente le compartiment us-west-2 en tant que `my-storage-bucket` et ajoute le nouveau compartiment us-east-1 nommé `my-storage-bucket-us-east-1`.

```
        "artifactStores": {
            "us-west-2": {
                "type": "S3",
                "location": "my-storage-bucket"
            },
            "us-east-1": {
                "type": "S3",
                "location": "my-storage-bucket-us-east-1"
            }
        },
```

4. Si vous utilisez la structure de pipeline extraite à l'aide de la commande get-pipeline, supprimez les lignes `metadata` du fichier JSON. Sinon, la commande update-pipeline ne peut pas l'utiliser. Supprimez les lignes `"metadata": { }` et les champs `"updated"`, `"created"` et `"pipelineARN"`.

   Par exemple, supprimez les lignes suivantes de la structure :

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

   Enregistrez le fichier.

5. Pour appliquer les modifications, exécutez la commande update-pipeline en spécifiant le fichier JSON du pipeline :

   > **Important**
   >
   > N'oubliez pas d'inclure `file://` devant le nom du fichier. Il est nécessaire dans cette commande.

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

   Cette commande affiche toute la structure du pipeline mise à jour. La sortie est similaire à ce qui suit.

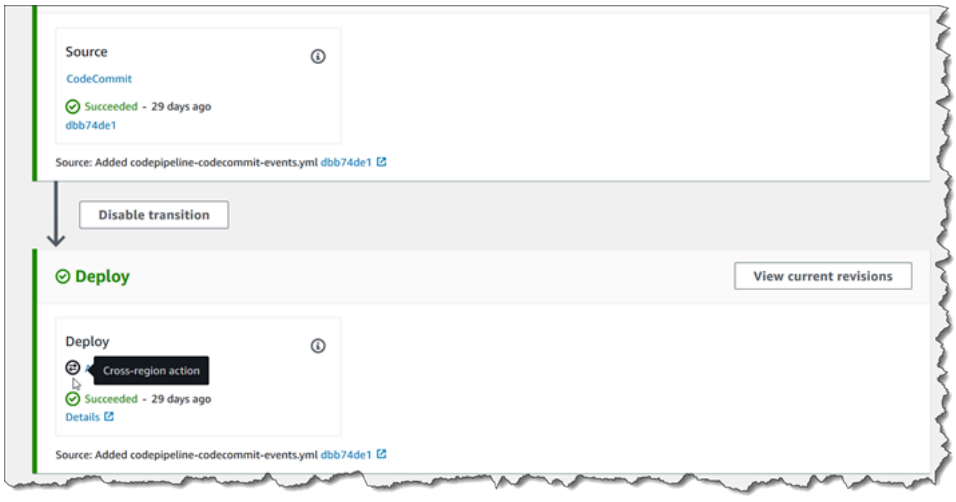
   ```
   {
       "pipeline": {
           "version": 4,
           "roleArn": "ARN",
           "stages": [
               {
                   "name": "Source",
                   "actions": [
                       {
                           "inputArtifacts": [],
                           "name": "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**
>
> La commande update-pipeline interrompt le pipeline. Si une révision est exécutée dans le pipeline lorsque vous exécutez la commande update-pipeline celle-ci est interrompue. Vous devez lancer manuellement le pipeline pour exécuter cette révision dans le pipeline mis à jour. Utilisez la commande **`start-pipeline-execution`** pour démarrer manuellement votre pipeline.

6. Une fois que vous avez mis à jour votre pipeline, l'action interrégionale s'affiche dans la console.



Ajoutez une action interrégionale à un pipeline (CloudFormation)

Vous pouvez l'utiliser CloudFormation pour ajouter une action interrégionale à un pipeline existant.

Pour ajouter une action interrégionale avec CloudFormation

1. Ajoutez le paramètre `Region` à la ressource `ActionDeclaration` dans votre modèle, comme illustré dans l'exemple suivant :

```
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. Sous `Mappings`, ajoutez le mappage de région comme indiqué dans cet exemple pour un mappage nommé `SecondRegionMap` qui mappe les valeurs des clés `RegionA` et `RegionB`. Sous la ressource `Pipeline`, sous le champ `artifactStore`, ajoutez la carte `artifactStores` pour votre nouvelle action inter-région comme suit :

```
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
```

L'exemple YAML suivant montre le *RegionA* compartiment en tant que tel `us-west-2` et ajoute le nouveau *RegionB* compartiment : `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. Enregistrez le modèle mis à jour sur votre ordinateur local, puis ouvrez la console CloudFormation .
4. Choisissez votre pile, puis Créer un jeu de modifications pour la pile actuelle.
5. Chargez le modèle mis à jour, puis affichez les modifications répertoriées dans CloudFormation. Il s'agit des modifications apportées à la pile. Vos nouvelles ressources doivent figurer dans la liste.
6. Sélectionnez Execute (Exécuter).

Utilisation des variables

Certaines actions permettent de CodePipeline générer des variables. Pour utiliser des variables :

- Vous affectez un espace de noms à une action pour rendre les variables qu'elle produit disponibles pour une configuration d'action en aval.
- Vous configurez l'action en aval pour utiliser les variables générées par l'action.

  Vous pouvez afficher les détails de chaque exécution d'action pour voir les valeurs de chaque variable de sortie générée par l'action au moment de l'exécution.

Pour voir step-by-step des exemples d'utilisation de variables :

- Pour un didacticiel présentant une action Lambda qui utilise les variables d'une action en amont (CodeCommit) et génère des variables de sortie, voir. Tutoriel : Utilisation de variables avec des actions d'invocation Lambda
- Pour un didacticiel présentant une CloudFormation action qui fait référence à des variables de sortie de pile issues d'une CloudFormation action en amont, consultezTutoriel : Création d'un pipeline qui utilise des variables provenant de AWS CloudFormation actions de déploiement.
- Pour un exemple d'action d'approbation manuelle avec un texte de message faisant référence à des variables de sortie correspondant à l'ID de CodeCommit validation et au message de validation, voirExemple : Utiliser des variables dans les approbations manuelles.
- Pour un exemple CodeBuild d'action avec une variable d'environnement qui correspond au nom de la GitHub branche, consultezExemple : utilisation d'une BranchName variable avec des variables d' CodeBuild environnement.

- CodeBuild les actions produisent sous forme de variables toutes les variables d'environnement exportées dans le cadre de la construction. Pour de plus amples informations, veuillez consulter CodeBuild variables de sortie d'action. Pour obtenir la liste des variables d'environnement que vous pouvez utiliser CodeBuild, consultez la section Variables d'environnement dans les environnements de construction dans le Guide de AWS CodeBuild l'utilisateur.

Rubriques

- Configuration d'actions pour des variables
- Affichage des variables de sortie
- Exemple : Utiliser des variables dans les approbations manuelles
- Exemple : utilisation d'une BranchName variable avec des variables d' CodeBuild environnement

Configuration d'actions pour des variables

Lorsque vous ajoutez une action à votre pipeline, vous pouvez lui affecter un espace de noms et le configurer pour utiliser les variables des actions précédentes.

Configuration d'actions avec les variables (console)

Cet exemple crée un pipeline avec une action CodeCommit source et une action CodeBuild build. L' CodeBuild action est configurée pour consommer les variables produites par l' CodeCommit action.

Si l'espace de noms n'est pas spécifié, les variables ne sont pas disponibles pour référence dans la configuration de l'action. Lorsque vous utilisez la console pour créer un pipeline, l'espace de noms de chaque action est généré automatiquement.

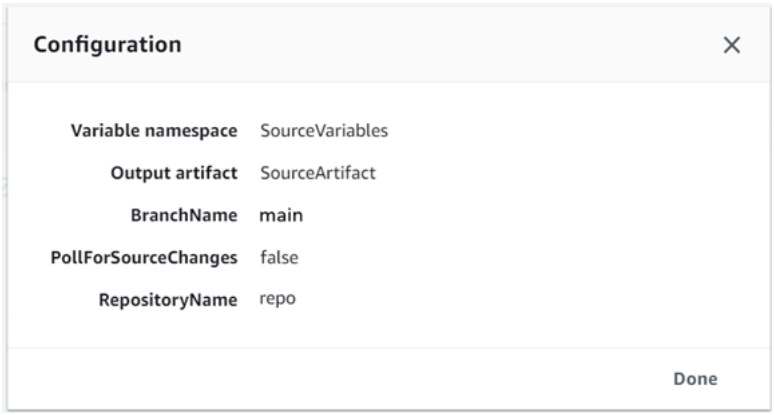
Pour créer un pipeline avec des variables

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Choisissez Créer un pipeline. Entrez un nom pour votre pipeline, puis choisissez Next (Suivant).
3. Dans Source, dans Fournisseur, sélectionnez CodeCommit. Choisissez le CodeCommit référentiel et la branche pour l'action source, puis choisissez Next.
4. Dans Build, dans Provider, choisissez CodeBuild. Choisissez le nom d'un projet de CodeBuild construction existant ou choisissez Créer un projet. Dans Créer un projet de construction, créez un projet de construction, puis choisissez Retourner à CodePipeline.

Sous Environment variables (Variables d'environnement), choisissez Add environment variables (Ajouter des variables d'environnement). Par exemple, entrez l'ID d'exécution avec la syntaxe variable `#{codepipeline.PipelineExecutionId}` et l'ID de validation avec la syntaxe variable`#{SourceVariables.CommitId}`.

> **Note**
>
> Vous pouvez entrer la syntaxe de variable dans n'importe quel champ de configuration d'action de l'assistant.

5. Choisissez Créer.
6. Une fois le pipeline créé, vous pouvez afficher l'espace de noms créé par l'assistant. Sur le pipeline, choisissez l'icône de l'étape dont vous souhaitez afficher l'espace de noms. Dans cet exemple, l'espace de noms généré automatiquement de l'action source, `SourceVariables`, est affiché.



Pour modifier l'espace de noms d'une action existante

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Choisissez le pipeline à modifier, puis choisissez Edit (Modifier). Pour l'étape source, choisissez Edit stage (Modifier l'étape). Ajoutez l' CodeCommit action.
3. Dans Edit action (Modifier l'action), affichez le champ Variable namespace (Espace de noms de variable). Si l'action existante a été créée précédemment ou sans utiliser l'assistant, vous devez ajouter un espace de noms. Dans Variable namespace (Espace de noms de variable), entrez un nom d'espace de noms, puis choisissez Save (Enregistrer).

Pour afficher les variables de sortie

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Une fois le pipeline créé et exécuté avec succès, vous pouvez afficher les variables sur la page Action execution details (Détails de l'exécution de l'action). Pour plus d'informations, consultez Affichage des variables (console).

Configuration d'actions pour les variables (interface de ligne de commande)

Lorsque vous utilisez la create-pipeline commande pour créer un pipeline ou la update-pipeline commande pour modifier un pipeline, vous pouvez utiliser reference/use des variables dans la configuration d'une action.

Si l'espace de noms n'est pas spécifié, les variables produites par l'action ne peuvent être référencées dans des configurations d'action en aval.

Pour configurer une action avec un espace de noms

1. Suivez les étapes décrites dans Création d'un pipeline, d'étapes et d'actions pour créer un pipeline à l'aide de l'interface de ligne de commande. Démarrez un fichier d'entrée pour fournir le paramètre `--cli-input-json` à la commande create-pipeline. Dans la structure du pipeline, ajoutez le paramètre `namespace` et spécifiez un nom, tel que `SourceVariables`.

```
. . .
{
          "inputArtifacts": [],
          "name": "Source",
          "region": "us-west-2",
          "namespace": "SourceVariables",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeCommit"
          },
          "outputArtifacts": [

. . .
```

2. Enregistrez le fichier avec un nom tel que **`MyPipeline.json`**.

3. Sur un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows), exécutez la create-pipelinecommande et créez le pipeline.

   Utilisez le fichier que vous avez créé lorsque vous avez exécuté la commande create-pipeline. Par exemple :

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

Pour configurer des actions en aval pour utiliser des variables

1. Modifiez un fichier d'entrée pour fournir le paramètre `--cli-input-json` à la commande update-pipeline. Dans l'action en aval, ajoutez la variable à la configuration pour cette action. Une variable est constituée d'un espace de noms et d'une clé, séparés par un point. Par exemple, pour ajouter des variables pour l'ID d'exécution du pipeline et l'ID de validation source, spécifiez l'espace de noms `codepipeline` pour la variable `#{codepipeline.PipelineExecutionId}`. Spécifiez l'espace de noms `SourceVariables` pour la variable `#{SourceVariables.CommitId}`.

```
{
    "name": "Build",
    "actions": [
        {
            "outputArtifacts": [
                {
                    "name": "BuildArtifacts"
                }
            ],
            "name": "Build",
            "configuration": {
                "EnvironmentVariables": "[{\"name\":\"Execution_ID\",\"value
\":\"#{codepipeline.PipelineExecutionId}\",\"type\":\"PLAINTEXT\"},{\"name\":
\"Commit_ID\",\"value\":\"#{SourceVariables.CommitId}\",\"type\":\"PLAINTEXT\"}]",
                "ProjectName": "env-var-test"
            },
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
                }
            ],
            "region": "us-west-2",
```

```
            "actionTypeId": {
                "provider": "CodeBuild",
                "category": "Build",
                "version": "1",
                "owner": "AWS"
            },
            "runOrder": 1
        }
    ]
},
```

2. Enregistrez le fichier avec un nom tel que **`MyPipeline.json`**.
3. Sur un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows), exécutez la create-pipelinecommande et créez le pipeline.

   Utilisez le fichier que vous avez créé lorsque vous avez exécuté la commande create-pipeline. Par exemple :

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

Affichage des variables de sortie

Vous pouvez afficher les détails de l'exécution de l'action pour voir les variables de cette action, spécifiques à chaque exécution.

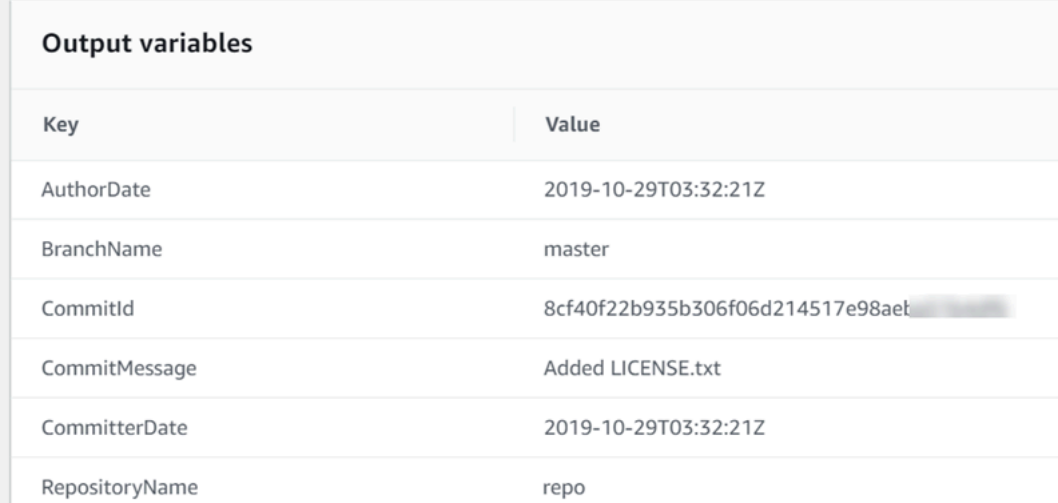
Affichage des variables (console)

Vous pouvez utiliser la console pour afficher les variables d'une action.

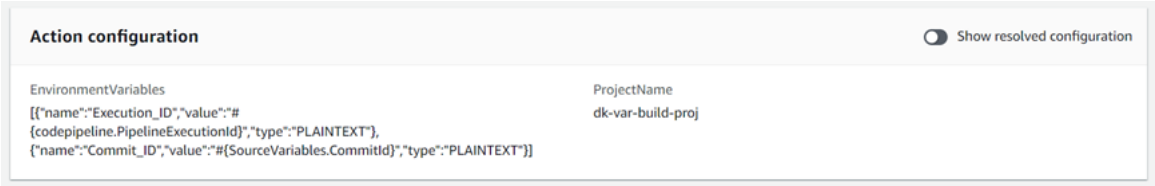
1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés.
2. Dans Name, choisissez le nom du pipeline.
3. Choisissez Afficher l'historique.
4. Une fois le pipeline exécuté avec succès, vous pouvez afficher les variables produites par l'action source. Choisissez Afficher l'historique. Choisissez Source dans la liste d'actions pour

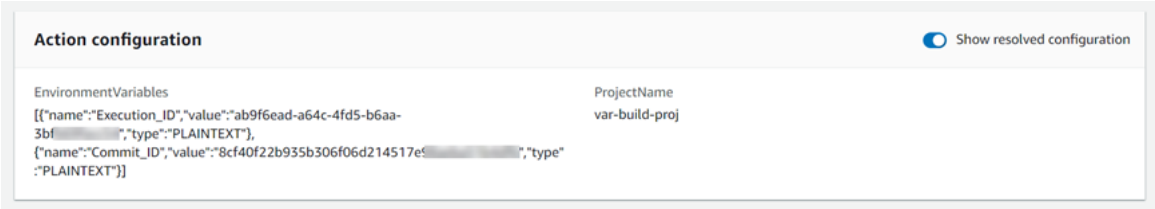
l'exécution du pipeline afin d'afficher les détails de l'exécution de l' CodeCommit action. Dans l'écran des détails de l'action, affichez les variables sous Output variables (Variables de sortie).



5. Une fois le pipeline exécuté avec succès, vous pouvez afficher les variables utilisées par l'action de génération. Choisissez Afficher l'historique. Dans la liste des actions pour l'exécution du pipeline, choisissez Build pour afficher les détails de l'exécution de l' CodeBuild action. Sur la page de détails de l'action, affichez les variables sous Action configuration (Configuration de l'action). L'espace de noms généré automatiquement s'affiche.



Par défaut, Action configuration (Configuration d'action) affiche la syntaxe de la variable. Vous pouvez choisir Show resolved configuration (Afficher la configuration résolue) pour faire basculer la liste de manière à afficher les valeurs produites lors de l'exécution de l'action.



Affichage des variables (interface de ligne de commande)

Vous pouvez utiliser la commande list-action-executions pour afficher les variables d'une action.

1. Utilisez la commande suivante :

```
aws codepipeline list-action-executions
```

La sortie affiche le paramètre `outputVariables` comme indiqué ici.

```
"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. Utilisez la commande suivante :

```
aws codepipeline get-pipeline --name <pipeline-name>
```

Dans la configuration de l' CodeBuild action, vous pouvez visualiser les variables :

```
{
    "name": "Build",
    "actions": [
        {
            "outputArtifacts": [
                {
                    "name": "BuildArtifact"
                }
            ],
            "name": "Build",
            "configuration": {
                "EnvironmentVariables": "[{\"name\":\"Execution_ID\",\"value
\":\"#{codepipeline.PipelineExecutionId}\",\"type\":\"PLAINTEXT\"},{\"name\":
\"Commit_ID\",\"value\":\"#{SourceVariables.CommitId}\",\"type\":\"PLAINTEXT\"}]",
                "ProjectName": "env-var-test"
            },
            "inputArtifacts": [
                {
                    "name": "SourceArtifact"
```

```
                }
            ],
            "region": "us-west-2",
            "actionTypeId": {
                "provider": "CodeBuild",
                "category": "Build",
                "version": "1",
                "owner": "AWS"
            },
            "runOrder": 1
        }
    ]
},
```

Exemple : Utiliser des variables dans les approbations manuelles

Lorsque vous spécifiez un espace de noms pour une action et que cette action produit des variables de sortie, vous pouvez ajouter une approbation manuelle qui affiche les variables dans le message d'approbation. Cet exemple montre comment ajouter une syntaxe de variable à un message d'approbation manuelle.

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés. Choisissez le pipeline auquel vous souhaitez ajouter l'approbation.
2. Pour modifier votre pipeline, choisissez Modifier. Ajoutez une approbation manuelle après l'action source. Dans Nom de l'action, entrez le nom de l'action d'approbation.
3. Dans Fournisseur d'action, choisissez Approbation manuelle.
4. Dans URL pour révision, ajoutez la syntaxe variable pour `CommitId` à votre CodeCommit URL. Assurez-vous d'utiliser l'espace de noms attribué à votre action source. Par exemple, la syntaxe de variable pour une action CodeCommit avec l'espace de noms par défaut `SourceVariables` est `#{SourceVariables.CommitId}`.

   Dans Commentaires, dans`CommitMessage`, entrez le message de validation :

   ```
   Please approve this change. Commit message: #{SourceVariables.CommitMessage}
   ```

5. Une fois le pipeline exécuté avec succès, vous pouvez afficher les valeurs des variables dans le message d'approbation.

Exemple : utilisation d'une BranchName variable avec des variables d' CodeBuild environnement

Lorsque vous ajoutez une CodeBuild action à votre pipeline, vous pouvez utiliser des variables d' CodeBuild environnement pour référencer une variable de `BranchName` sortie issue d'une action source en amont. Avec une variable de sortie provenant d'une action CodePipeline, vous pouvez créer vos propres variables d' CodeBuildenvironnement à utiliser dans vos commandes de construction.

Cet exemple montre comment ajouter la syntaxe d'une variable de sortie d'une action GitHub source à une variable d' CodeBuild environnement. Dans cet exemple, la syntaxe de la variable de sortie représente la variable de sortie de l'action GitHub source pour`BranchName`. Une fois l'action exécutée avec succès, la variable est résolue pour afficher le nom de la GitHub branche.

1. Connectez-vous à la CodePipeline console Console de gestion AWS et ouvrez-la à l'adresse http://console.aws.amazon. com/codesuite/codepipeline/home.

   Les noms de tous les pipelines associés à votre AWS compte sont affichés. Choisissez le pipeline auquel vous souhaitez ajouter l'approbation.
2. Pour modifier votre pipeline, choisissez Modifier. Sur la scène qui contient votre CodeBuild action, choisissez Modifier l'étape.
3. Cliquez sur l'icône pour modifier votre CodeBuild action.
4. Sur la page Modifier l'action, sous Variables d'environnement, entrez ce qui suit :

   - Dans Nom, entrez le nom de votre variable d'environnement.
   - Dans Value, entrez la syntaxe de la variable de sortie de votre pipeline, qui inclut l'espace de noms attribué à votre action source. Par exemple, la syntaxe de la variable de sortie pour une GitHub action avec l'espace de noms par défaut `SourceVariables` est`#{SourceVariables.BranchName}`.
   - Dans Type, choisissez Plaintext.
5. Une fois le pipeline exécuté avec succès, vous pouvez voir comment la variable de sortie résolue est la valeur de la variable d'environnement. Sélectionnez l'une des méthodes suivantes :

- CodePipeline console : choisissez votre pipeline, puis sélectionnez Historique. Choisissez l'exécution de pipeline la plus récente.
  - Sous Chronologie, choisissez le sélecteur pour Source. Il s'agit de l'action source qui génère les variables GitHub de sortie. Choisissez Afficher les détails de l'exécution. Sous Variables de sortie, consultez la liste des variables de sortie générées par cette action.
  - Sous Chronologie, choisissez le sélecteur pour Build. Il s'agit de l'action de génération qui spécifie les variables d' CodeBuild environnement de votre projet de génération. Choisissez Afficher les détails de l'exécution. Sous Configuration des actions, consultez vos variables d' CodeBuildenvironnement. Choisissez Afficher la configuration résolue. La valeur de votre variable d'environnement est la variable de `BranchName` sortie résolue de l'action GitHub source. Dans cet exemple, la valeur résolue est`main`.

    Pour de plus amples informations, veuillez consulter Affichage des variables (console).
- CodeBuild console : Choisissez votre projet de build et choisissez le lien pour votre exécution de build. Sous Variables d'environnement, votre variable de sortie résolue est la valeur de la variable d' CodeBuild environnement. Dans cet exemple, la variable d'environnement Name est `BranchName` et la valeur est la variable de `BranchName` sortie résolue à partir de l'action GitHub source. Dans cet exemple, la valeur résolue est`main`.



Travailler avec les transitions d'étape dans CodePipeline

Les transitions sont des liens entre les étapes d'un pipeline qui peut être activés ou désactivés. Ils sont activés par défaut. Lorsque vous réactivez une transition désactivée, la dernière révision s'exécute dans les étapes restantes du pipeline, sauf si moins de 30 jours se sont écoulés. L'exécution du pipeline ne reprendra pas pour une transition qui a été désactivée pendant plus de 30 jours, sauf si une nouvelle modification est détectée ou si vous réexécutez manuellement le pipeline.

Vous pouvez utiliser la AWS CodePipeline console ou le AWS CLI pour activer ou désactiver les transitions entre les étapes d'un pipeline.

> **Note**
>
> Vous pouvez utiliser une action d'approbation pour interrompre l'exécution d'un pipeline jusqu'à ce que ce dernier soit approuvé manuellement pour continuer. Pour de plus amples informations, veuillez consulter Ajouter une action d'approbation manuelle à une étape.

Rubriques

- Désactivation ou activation des transitions (console)
- Désactivation ou activation des transitions (interface de ligne de commande)

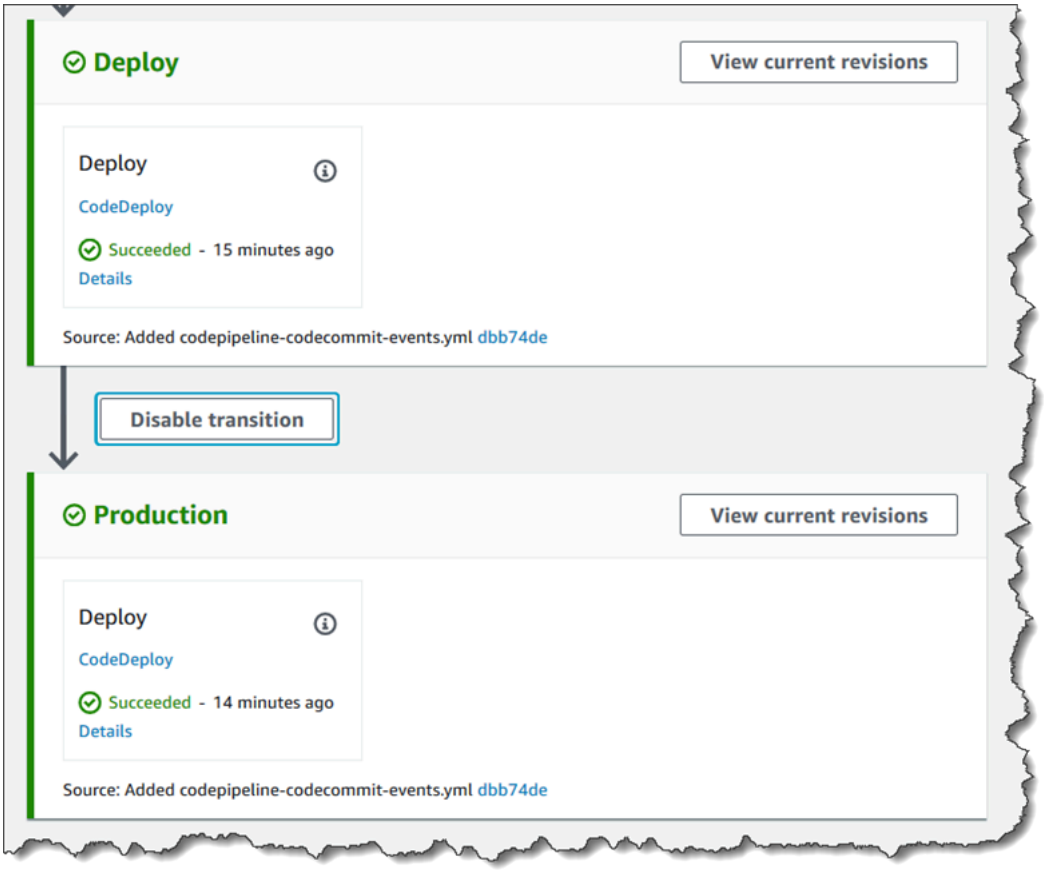
Désactivation ou activation des transitions (console)

Pour désactiver ou activer les transitions dans un pipeline

1. Connectez-vous à Console de gestion AWS et ouvrez la CodePipeline console à l'adresse http://console.aws.amazon.com/codesuite/codepipeline/home.

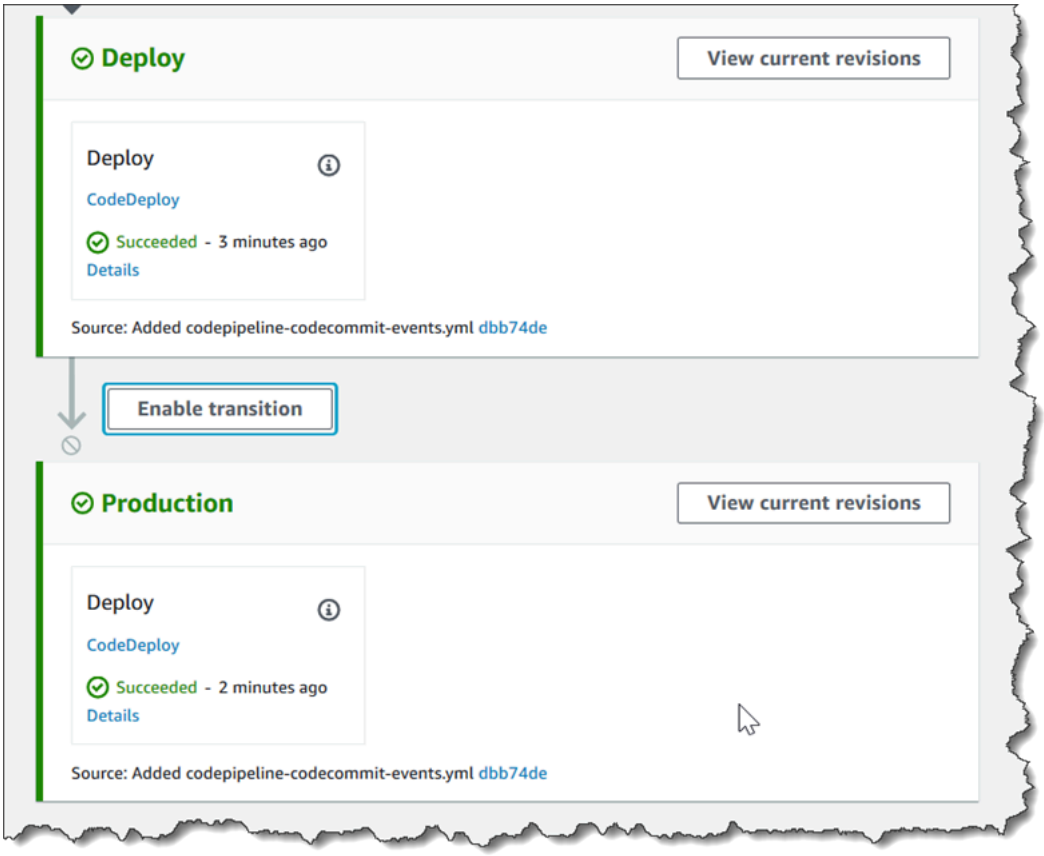
   Les noms de tous les pipelines associés à votre compte AWS s'affichent.
2. Dans Name, choisissez le nom du pipeline dans lequel vous souhaitez activer ou désactiver les transitions. Cette opération ouvre une vue détaillée du pipeline, y compris des transitions entre les étapes du pipeline.
3. Trouvez la flèche après la dernière étape que vous souhaitez exécuter, et choisissez le bouton à côté de cette dernière. Par exemple, dans le pipeline suivant fourni à titre d'exemple, si vous

souhaitez que les actions de l'étape nommée Intermédiaire soient exécutées, contrairement aux actions de l'étape nommée Production, choisissez le bouton Désactiver la transition entre ces deux étapes :



4. Dans la boîte de dialogue Désactiver la transition, renseignez la raison pour laquelle vous désactivez la transition, et choisissez Désactiver.

   Le bouton change pour indiquer que les transitions sont désactivées entre l'étape qui précède la flèche et celle qui la suit. Toutes les révisions qui étaient déjà en cours d'exécution dans les étapes postérieures à la transition désactivée continuent dans le pipeline, mais les révisions suivantes sont arrêtées par la transition désactivée.



5. Choisissez le bouton Activer la transition en regard de la flèche. Dans la boîte de dialogue Activation d'une transition, choisissez Activer. Le pipeline active immédiatement la transition entre les deux étapes. Si des révisions ont été exécutées dans les étapes précédentes après la désactivation de la transition, le pipeline commence au bout de quelques instants à exécuter la dernière révision à travers les étapes subséquentes à la transition préalablement désactivée. Le pipeline exécute la révision à travers toutes les étapes restantes du pipeline.

> **Note**
>
> L'affichage des modifications dans la CodePipeline console peut prendre quelques secondes une fois la transition activée.

Désactivation ou activation des transitions (interface de ligne de commande)

Pour désactiver une transition entre les étapes à l'aide de AWS CLI, exécutez la disable-stage-transition commande. Pour activer une transition désactivée, exécutez la commande enable-stage-transition.

Désactivation d'une transition

1. Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et exécutez la AWS CLI commande disable-stage-transition, en spécifiant le nom du pipeline, le nom de l'étape vers laquelle vous souhaitez désactiver les transitions, le type de transition et la raison pour laquelle vous désactivez les transitions vers cette étape. Contrairement à la console, vous devez également préciser si vous désactivez les transitions vers l'étape (entrantes) ou les transitions depuis cette étape une fois que toutes les actions sont terminées (sortantes).

   Par exemple, pour désactiver la transition vers une étape nommée *Staging* dans un pipeline nommé*MyFirstPipeline*, vous devez taper une commande similaire à la suivante :

   ```
   aws codepipeline disable-stage-transition --pipeline-name MyFirstPipeline --stage-name Staging --transition-type Inbound --reason "My Reason"
   ```

   La commande n'affiche rien.
2. Pour vérifier que la transition a été désactivée, affichez le pipeline dans la console CodePipeline ou exécutez la commande get-pipeline-state. Pour plus d'informations, consultez Afficher les pipelines (console) et Affichage des détails et de l'historique d'un pipeline (interface de ligne de commande).

Pour activer une transition

1. Ouvrez un terminal (Linux, macOS ou Unix) ou une invite de commande (Windows) et exécutez la AWS CLI commande enable-stage-transition, en spécifiant le nom du pipeline, le nom de l'étape vers laquelle vous souhaitez activer les transitions et le type de transition.

   Par exemple, pour activer la transition vers une étape nommée *Staging* dans un pipeline nommé*MyFirstPipeline*, vous devez taper une commande similaire à la suivante :

```
aws codepipeline enable-stage-transition --pipeline-name MyFirstPipeline --stage-
name Staging  --transition-type Inbound
```

La commande n'affiche rien.

2. Pour vérifier que la transition a été activée, visualisez le pipeline dans la CodePipeline console ou exécutez la get-pipeline-state commande. Pour plus d'informations, consultez Afficher les pipelines (console) et Affichage des détails et de l'historique d'un pipeline (interface de ligne de commande).

Surveillance des pipelines

La surveillance est un enjeu important pour assurer la fiabilité, la disponibilité et les performances de AWS CodePipeline. Vous devez collecter des données de surveillance provenant de toutes les parties de votre AWS solution afin de pouvoir corriger plus facilement une défaillance multipoint, le cas échéant. Avant de commencer le suivi, vous devez créer un plan de surveillance qui réponde aux questions suivantes :

- Quels sont les objectifs de la surveillance ?
- Quelles sont les ressources à surveiller ?
- A quelle fréquence les ressources doivent-elles être surveillées ?
- Quels outils de surveillance sont disponibles à l'emploi ?
- Qui exécute les tâches de supervision ?
- Qui doit être informé en cas de problème ?

Vous pouvez utiliser les outils suivants pour surveiller vos CodePipeline pipelines et leurs ressources :

- EventBridge événements du bus d'événements : vous pouvez surveiller les CodePipeline événements dans EventBridge, qui détectent les modifications de l'état d'exécution de votre pipeline, de votre étape ou de votre action. EventBridge achemine ces données vers des cibles telles qu' AWS Lambda Amazon Simple Notification Service. EventBridge les événements sont les mêmes que ceux qui apparaissent dans Amazon CloudWatch Events.
- Notifications relatives aux événements du pipeline dans la console Developer Tools : vous pouvez surveiller les CodePipeline événements à l'aide des notifications que vous avez configurées dans la console, puis créer un sujet et un abonnement Amazon Simple Notification Service pour lesquels vous vous abonnez. Pour plus d'informations, voir Que sont les notifications dans le guide de l'utilisateur de la console Developer Tools.
- AWS CloudTrail— CloudTrail À utiliser pour capturer les appels d'API effectués par ou pour le compte de votre AWS compte et envoyer les fichiers journaux CodePipeline dans un compartiment Amazon S3. Vous pouvez choisir de CloudWatch publier des notifications Amazon SNS lorsque de nouveaux fichiers journaux sont livrés afin de pouvoir agir rapidement.
- Console et CLI : vous pouvez utiliser la CodePipeline console et l'interface de ligne de commande pour afficher les détails relatifs à l'état d'un pipeline ou à l'exécution d'un pipeline en particulier.

Rubriques

- Surveillance des CodePipeline événements
- Référence des compartiments d'espace réservé pour les événements
- Journalisation des appels d' CodePipeline API avec AWS CloudTrail
- CodePipeline CloudWatch métriques

Surveillance des CodePipeline événements

Vous pouvez surveiller CodePipeline les événements dans EventBridge, qui fournit un flux de données en temps réel provenant de vos propres applications, applications SaaS et. Services AWS EventBridge achemine ces données vers des cibles telles qu' AWS Lambda Amazon Simple Notification Service. Ces événements sont les mêmes que ceux qui apparaissent dans Amazon CloudWatch Events, qui diffuse en temps quasi réel un flux d'événements système décrivant l'évolution des AWS ressources. Pour plus d'informations, consultez Qu'est-ce qu'Amazon EventBridge ? dans le guide de EventBridge l'utilisateur Amazon.

> **Note**
>
> Amazon EventBridge est la solution préférée pour gérer vos événements. Amazon CloudWatch Events et Amazon EventBridge constituent le même service sous-jacent et la même API, mais EventBridge proposent davantage de fonctionnalités. Les modifications que vous apportez dans CloudWatch Événements ou EventBridge apparaîtront dans chaque console.

Les événements sont composés de règles. Une règle est configurée en choisissant les options suivantes :

- Schéma des événements. Chaque règle est exprimée sous la forme d'un modèle d'événement avec la source et le type d'événements à surveiller, ainsi que les cibles des événements. Pour surveiller les événements, vous créez une règle avec le service que vous surveillez comme source d'événements, par exemple CodePipeline. Par exemple, vous pouvez créer une règle avec un modèle d'événement qui est utilisé CodePipeline comme source d'événement pour déclencher la règle en cas de modification de l'état d'un pipeline, d'une étape ou d'une action.
- Cibles La nouvelle règle reçoit un service sélectionné en tant que cible d'événement. Vous souhaiterez peut-être configurer un service cible pour envoyer des notifications, capturer des

informations d'état, prendre des mesures correctives, lancer des événements ou prendre d'autres mesures. Lorsque vous ajoutez votre cible, vous devez également lui accorder des autorisations EventBridge pour lui permettre d'invoquer le service cible sélectionné.

Chaque type d'événement de changement d'état d'exécution émet des notifications avec un contenu de message spécifique dans lequel :

- L'`version`entrée initiale indique le numéro de version de l'événement.
- L'entrée `version` sous `detail` du pipeline indique le numéro de version de la structure du pipeline.
- L'entrée `execution-id` sous `detail` du pipeline indique l'ID d'exécution pour l'exécution du pipeline ayant causé le changement d'état. Reportez-vous à l'appel d'GetPipelineExecutionAPI dans la référence AWS CodePipeline d'API.
- L'`pipeline-execution-attempt`entrée indique le nombre de tentatives, ou de nouvelles tentatives, pour l'ID d'exécution spécifique.

CodePipeline signale un événement EventBridge chaque fois que l'état d'une ressource de votre site Compte AWS change. Les événements sont émis au moins une fois de manière garantie pour les ressources suivantes :

- Exécutions de pipeline
- Exécutions scéniques
- Exécutions d'actions

Les événements sont émis selon EventBridge le modèle et le schéma d'événements détaillés ci-dessus. Pour les événements traités, tels que les événements que vous recevez par le biais de notifications que vous avez configurées dans la console Developer Tools, le message d'événement inclut des champs de modèle d'événement avec certaines variantes. Par exemple, le `detail-type` champ est converti en`detailType`. Pour plus d'informations, reportez-vous à l'appel PutEvents d'API dans le manuel Amazon EventBridge API Reference.

Les exemples suivants présentent des événements pour CodePipeline. Dans la mesure du possible, chaque exemple montre le schéma d'un événement émis ainsi que le schéma d'un événement traité.

Rubriques

- Types de détails

- Pipeline-level événements
- Stage-level événements
- Action-level événements
- Création d'une règle qui envoie une notification concernant un événement de pipeline

Types de détails

Lorsque vous configurez des événements à surveiller, vous pouvez choisir le type de détail de l'événement.

Vous pouvez configurer des notifications à envoyer lorsque l'état passe à :

- Les pipelines spécifiés ou tous vos pipelines. Vous pouvez contrôler cet élément à l'aide de `"detail-type": "CodePipeline Pipeline Execution State Change"`
- Les étapes spécifiées ou toutes vos étapes, dans un pipeline spécifié ou tous vos pipelines. Vous pouvez contrôler cet élément à l'aide de `"detail-type": "CodePipeline Stage Execution State Change"`
- Les actions spécifiées ou toutes les actions, dans une étape spécifiées ou toutes les étapes, dans un pipeline spécifié ou tous vos pipelines. Vous pouvez contrôler cet élément à l'aide de `"detail-type": "CodePipeline Action Execution State Change"`

> **Note**
>
> Les événements émis par EventBridge contiennent le `detail-type` paramètre, qui est converti en `detailType` date de traitement des événements.

Type de détail	State	Description
CodePipeline Changement de l'état d'exécution du pipeline	ANNULÉE	L'exécution du pipeline a été annulée en raison de la mise à jour de la structure du pipeline.
	ÉCHEC	L'exécution de pipeline ne s'est pas terminée avec succès.

Type de détail	State	Description
	REPRISE	Une exécution de pipeline échouée été retentée en réponse à l'appel d'API `RetryStageExecution` .
	DÉMARRÉE	L'exécution du pipeline est en cours d'exécution.
	STOPPED	Le processus d'arrêt aboutit et l'exécution du pipeline est arrêtée.
	STOPPING	L'exécution du pipeline s'arrête en raison d'une demande d'arrêt et d'attente ou d'arrêt et d'abandon de l'exécution du pipeline.
	RÉUSSI	L'exécution de pipeline s'est terminée avec succès.
	REMPLACÉE	Tandis que l'exécution de ce pipeline était en attente de l'achèvement de l'étape suivante, une autre exécution de pipeline plus récente a avancé et a poursuivi via ce pipeline.
CodePipeline Changement d'état d'exécution de la phase	ANNULÉE	L'étape a été annulée en raison de la mise à jour de la structure du pipeline.
	ÉCHEC	L'étape ne s'est pas terminée avec succès.
	REPRISE	Une étape échouée été retentée en réponse à l'appel d'API `RetryStageExecution` .
	DÉMARRÉE	L'étape est en cours d'exécution.
	STOPPED	Le processus d'arrêt aboutit et l'exécution de la phase est arrêtée.
	STOPPING	L'exécution de la phase s'arrête en raison d'une demande d'arrêt et d'attente ou d'arrêt et d'abandon de l'exécution du pipeline.
	RÉUSSI	L'étape s'est terminée avec succès.

Type de détail	State	Description
CodePipeline Changement de l'état d'exécution de l'action	ABANDONNÉ	L'action est abandonnée en raison d'une demande d'arrêt et d'abandon de l'exécution du pipeline.
	ANNULÉE	L'action a été annulée en raison de la mise à jour de la structure du pipeline.
	ÉCHEC	Pour les action d'approbation, l'état FAILED signifie que l'action a été rejetée par le réviseur ou a échoué en raison d'une configuration d'action incorrecte.
	DÉMARRÉE	L'action est en cours d'exécution.
	RÉUSSI	L'action s'est terminée avec succès.

Pipeline-level événements

Pipeline-level des événements sont émis lors d'un changement d'état pour l'exécution d'un pipeline.

Rubriques

- Évènement Pipeline STARTED
- événement Pipeline STOPPING
- événement Pipeline SUCCEE
- Pipeline réussi (exemple avec des balises Git)
- événement Pipeline FAILED
- ÉCHEC DU PIPELINE (exemple avec des balises Git)

Évènement Pipeline STARTED

Lorsqu'une exécution de pipeline démarre, elle émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé `"myPipeline"` dans la `us-east-1` région. Le `id` champ représente l'ID de l'événement et le `account` champ représente l'ID du compte sur lequel le pipeline est créé.

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": {}
}
```

événement Pipeline STOPPING

Lorsqu'une exécution de pipeline s'arrête, elle émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé `myPipeline` dans la `us-west-2` région.

```
{
    "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"
    }
}
```

événement Pipeline SUCCEE

Lorsqu'une exécution de pipeline réussit, elle émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé `myPipeline` dans la `us-east-1` région.

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 réussi (exemple avec des balises Git)

Lorsqu'une étape de l'exécution d'un pipeline a été réessayée et a réussi, elle émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé `myPipeline` dans la `eu-central-1` région où `execution-trigger` est configuré pour les balises Git.

> **Note**
>
> Le `execution-trigger` champ comportera l'une `tag-name` ou l'autre des deux`branch-name`, selon le type d'événement qui a déclenché le 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
    }
}
```

événement Pipeline FAILED

Lorsqu'une exécution de pipeline échoue, il émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé "myPipeline" dans la us-west-2 région.

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"
    }
```

ÉCHEC DU PIPELINE (exemple avec des balises Git)

À moins qu'il n'échoue au stade source, pour un pipeline configuré avec des déclencheurs, il émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé `myPipeline` dans la `eu-central-1` région où `execution-trigger` est configuré pour les balises Git.

> **Note**
>
> Le `execution-trigger` champ comportera l'une `tag-name` ou l'autre des deux`branch-name`, selon le type d'événement qui a déclenché le 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 événements

Stage-level des événements sont émis lorsqu'il y a un changement d'état pour l'exécution d'une étape.

Rubriques

- Événement Stage STARTED
- Événement Stage STOPPING
- Événement Stage STOPLED
- Reprise de l'étape après une nouvelle tentative

Événement Stage STARTED

Lorsque l'exécution d'une étape démarre, elle émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé `"myPipeline"` dans la `us-east-1` région, pour l'étape`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"
                }
            }
        ]
    }
}
```

Événement Stage STOPPING

Lorsque l'exécution d'une étape s'arrête, elle émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé `myPipeline` dans la `us-west-2` région, pour l'étape`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
    }
}
```

Événement Stage STOPLED

Lorsque l'exécution d'une étape est arrêtée, elle émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé `myPipeline` dans la `us-west-2` région, pour l'étape`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
    }
}
```

Reprise de l'étape après une nouvelle tentative

Lorsque l'exécution d'une étape est reprise et qu'une étape a été réessayée, elle émet un événement qui envoie des notifications avec le contenu suivant.

Lorsqu'une étape a été réessayée, le `stage-last-retry-attempt-time` champ s'affiche, comme indiqué dans l'exemple. Le champ s'affiche sur tous les événements de la scène si une nouvelle tentative a été effectuée.

> **Note**
>
> Le `stage-last-retry-attempt-time` champ sera présent lors de toutes les étapes suivantes une fois qu'une étape aura été réessayée.

```
{
    "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 événements

Action-level des événements sont émis lorsqu'il y a un changement d'état pour l'exécution d'une action.

Rubriques

- Événement Action STARTED
- Événement Action SUCCEE
- Événement Action FAILED
- Événement Action ABANDONED

Événement Action STARTED

Lorsque l'exécution d'une action démarre, elle émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé `myPipeline` dans la `us-east-1` région, pour l'action de déploiement`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": {}
}
```

Événement Action SUCCEE

Lorsqu'une action est exécutée avec succès, elle émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé `"myPipeline"` dans la `us-west-2` région, pour l'action source`"Source"`. Pour ce type d'événement, il existe deux `region` champs différents. Le `region` champ d'événement spécifie la région pour l'événement de pipeline. Le `region` champ situé sous `detail` cette section indique la région de l'action.

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": {}
}
```

Événement Action FAILED

Lorsqu'une action échoue, elle émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé "`myPipeline`" dans la `us-west-2` région, pour l'action"`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"
    }
}
```

Événement Action ABANDONED

Lorsqu'une action est abandonnée, elle émet un événement qui envoie des notifications avec le contenu suivant. Cet exemple concerne le pipeline nommé "myPipeline" dans la us-west-2 région, pour l'action"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
    }
```

```
}
```

Création d'une règle qui envoie une notification concernant un événement de pipeline

Une règle surveille certains événements, puis les achemine vers les AWS cibles que vous choisissez. Vous pouvez créer une règle qui exécute une AWS action automatiquement lorsqu'une autre AWS action se produit, ou une règle qui exécute une AWS action régulièrement selon un calendrier défini.

Rubriques

- Envoyer une notification lorsque l'état du pipeline change (console)
- Envoyer une notification lorsque l'état du pipeline change (CLI)

Envoyer une notification lorsque l'état du pipeline change (console)

Ces étapes montrent comment utiliser la EventBridge console pour créer une règle afin d'envoyer des notifications de modifications apportées à CodePipeline.

Pour créer une EventBridge règle qui cible votre pipeline avec une source Amazon S3

1. Ouvrez la EventBridge console Amazon à l'adresse https://console.aws.amazon.com/events/.
2. Dans le panneau de navigation, choisissez Rules. Laissez le bus par défaut sélectionné ou choisissez un bus d'événements. Choisissez Créer une règle.
3. Dans Nom, entrez le nom de votre règle.
4. Sous Type de règle, choisissez Règle avec un modèle d'événement. Choisissez Suivant.
5. Sous Modèle d'événement, sélectionnez AWS les services.
6. Depuis la liste déroulante Type d'événement, choisissez le niveau de changement d'état pour la notification.

   - Pour une règle qui s'applique aux événements au niveau du pipeline, choisissez CodePipeline Pipeline Execution State Change.
   - Pour une règle qui s'applique aux événements au niveau de l'étape, choisissez CodePipeline Stage Execution State Change.
   - Pour une règle qui s'applique aux événements au niveau de l'action, choisissez CodePipeline Action Execution State Change.

7. Spécifiez les modifications d'état auxquelles la règle s'applique :

   - Pour une règle qui s'applique à tous les changements d'état, choisissez Tous les états.
   - Pour une règle qui s'applique à certaines modifications d'état uniquement, choisissez État(s) spécifique(s), puis sélectionnez une ou plusieurs valeurs d'état dans la liste.

8. Pour les modèles d'événements plus détaillés que ne le permettent les sélecteurs, vous pouvez également utiliser l'option Modifier le modèle dans la fenêtre Modèle d'événement pour désigner un modèle d'événement au format JSON.

   > **Note**
   >
   > Sauf indication contraire, le modèle d'événement est créé pour tous les pipelines/stages / actions et les états.

   Pour des modèles d'événements plus détaillés, vous pouvez copier et coller les exemples de modèles d'événements suivants dans la fenêtre Modèle d'événement.

   - Example

     Utilisez cet exemple de modèle d'événement pour capturer les actions de déploiement et de build échouées parmi tous les pipelines.

```
{
"source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Deploy", "Build"]
    }
  }
}
```

- Example

  Utilisez cet exemple de modèle d'événement pour capturer les actions rejetés ou en échec d'approbation parmi tous les pipelines.

```
{
 "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Approval"]
    }
  }
}
```

- Example

  Utilisez cet exemple de modèle d'événement pour capturer tous les événements depuis les pipelines spécifiés.

```
{
"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. Choisissez Suivant.

10. Dans Types de cibles, choisissez AWS service.
11. Dans Sélectionnez une cible, choisissez CodePipeline. Dans Pipeline ARN, entrez l'ARN du pipeline pour le pipeline à démarrer selon cette règle.

> **Note**
>
> Pour l'obtenir l'ARN de pipeline, exécutez la commande get-pipeline. L'ARN de pipeline apparaît dans la sortie. Il est créé dans ce format :
> arn:aws:codepipeline : : : *region account pipeline-name*
> Exemple d'ARN de pipeline :
> arn:aws:codepipeline:us-east- 2:80398 EXAMPLE:MyFirstPipeline

12. Pour créer ou spécifier un rôle de service IAM qui accorde EventBridge des autorisations pour appeler la cible associée à votre EventBridge règle (dans ce cas, la cible est CodePipeline) :
    - Choisissez Créer un nouveau rôle pour cette ressource spécifique afin de créer un rôle de service qui donne EventBridge les autorisations nécessaires pour démarrer les exécutions de votre pipeline.
    - Choisissez Utiliser le rôle existant pour saisir un rôle de service qui autorise EventBridge le démarrage des exécutions de votre pipeline.
13. Choisissez Suivant.
14. Sur la page Tags, choisissez Suivant.
15. Sur la page Réviser et créer, passez en revue la configuration des règles. Si la règle vous convient, choisissez Créer une règle.

Envoyer une notification lorsque l'état du pipeline change (CLI)

Ces étapes montrent comment utiliser l'interface de ligne de commande pour créer une règle d' CloudWatch événements afin d'envoyer des notifications de modifications dans CodePipeline.

Pour utiliser le AWS CLI pour créer une règle, appelez la put-rule commande en spécifiant :

- Un nom qui identifie de façon unique la règle que vous créez. Ce nom doit être unique pour tous les pipelines que vous créez et qui CodePipeline sont associés à votre AWS compte.
- Le modèle d'événement pour la source et les champs de détails utilisés par la règle. Pour plus d'informations, consultez Amazon EventBridge et Event Patterns.

Pour créer une EventBridge règle avec CodePipeline comme source d'événement

1. Appelez la commande put-rule pour créer une règle spécifiant le modèle d'événement. (Voir les tables précédentes pour obtenir les états valides.)

   L'exemple de commande suivant permet --event-pattern de créer une règle appelée `"MyPipelineStateChanges"` qui émet l' CloudWatch événement en cas d'échec de l'exécution d'un pipeline pour le pipeline nommé « 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. Appelez la put-targets commande et incluez les paramètres suivants :

   - Le paramètre `--rule` s'utilise avec le la règle `rule_name` que vous avez créée à l'aide de la commande put-rule.
   - Le `--targets` paramètre est utilisé avec la liste `Id` des cibles dans la liste des cibles et dans `ARN` la rubrique Amazon SNS.

   L'exemple de commande suivant spécifie que pour la règle appelée `MyPipelineStateChanges`, l'`Id` cible est composé du numéro un, ce qui indique qu'il s'agit de la règle 1 dans une liste de cibles pour la règle. L'exemple de commande spécifie également un exemple `ARN` pour la rubrique Amazon SNS.

   ```
   aws events put-targets --rule MyPipelineStateChanges --targets
    Id=1,Arn=arn:aws:sns:us-west-2:11111EXAMPLE:MyNotificationTopic
   ```

3. Ajoutez des autorisations EventBridge pour utiliser le service cible désigné afin d'appeler la notification. Pour plus d'informations, consultez la section Utilisation de politiques basées sur les ressources pour Amazon. EventBridge

Référence des compartiments d'espace réservé pour les événements

Cette section est fournie à des fins de référence uniquement. Pour obtenir des informations sur la création d'un pipeline avec des ressources de détection d'événements, veuillez consulter Actions à la source et méthodes de détection des modifications.

Créez des actions fournies par Amazon S3 et CodeCommit utilisez des ressources de détection des modifications basées sur des événements pour déclencher votre pipeline lorsqu'une modification est apportée au compartiment ou au référentiel source. Ces ressources sont les règles relatives aux CloudWatch événements qui sont configurées pour répondre aux événements de la source du pipeline, tels qu'une modification du code du CodeCommit référentiel. Lorsque vous utilisez CloudWatch Events pour une source Amazon S3, vous devez l'activer CloudTrail pour que les événements soient enregistrés. CloudTrail nécessite un compartiment S3 dans lequel il peut envoyer ses résumés. Vous pouvez accéder aux fichiers journaux de vos ressources d' CloudWatch événements à partir du compartiment personnalisé, mais vous ne pouvez pas accéder aux données du compartiment réservé.

- Si vous avez utilisé la CLI ou CloudFormation pour configurer les ressources CloudWatch Events, vous pouvez trouver vos CloudTrail fichiers dans le bucket que vous avez spécifié lors de la configuration de votre pipeline.
- Si vous avez utilisé la console pour configurer votre pipeline avec une source S3, la console utilise un compartiment CloudTrail réservé lorsqu'elle crée vos ressources d' CloudWatch événements pour vous. CloudTrail les résumés sont stockés dans le compartiment réservé dans Région AWS lequel le pipeline est créé.

Vous pouvez modifier la configuration si vous souhaitez utiliser un compartiment autre que le compartiment d'espace réservé.

> **Note**
>
> Les données enregistrées CloudTrail dans des compartiments réservés expirent automatiquement au bout d'un jour et ne sont pas conservées.

Pour plus d'informations sur la recherche et la gestion de vos fichiers CloudTrail journaux, consultez la section Obtenir et consulter vos fichiers CloudTrail journaux.

Rubriques

- Noms des compartiments d'espace réservé pour les événements par région

Noms des compartiments d'espace réservé pour les événements par région

Ce tableau répertorie les noms des compartiments réservés S3 qui contiennent des fichiers journaux permettant de suivre les événements de détection de modifications pour les pipelines comportant des actions source Amazon S3.

Nom de la région	Nom du compartiment d'espace réservé	Identifiant de région
USA Est (Ohio)	codepipeline-cloudtrail-placeholder-bucket-us-est-2	us-east-2
USA Est (Virginie du Nord)	codepipeline-cloudtrail-placeholder-bucket-us-est-1	us-east-1
USA Ouest (Californie du Nord)	codepipeline-cloudtrail-placeholder-bucket-us-ouest-1	us-west-1
USA Ouest (Oregon)	codepipeline-cloudtrail-placeholder-bucket-us-ouest-2	us-west-2
Canada (Centre)	codepipeline-cloudtrail-placeholder-bucket-ca-central-1	ca-central-1
Europe (Francfort)	codepipeline-cloudtrail-placeholder-bucket-eu-central-1	eu-central-1
Europe (Irlande)	codepipeline-cloudtrail-placeholder-bucket-eu-ouest-1	eu-west-1
Europe (Londres)	codepipeline-cloudtrail-placeholder-bucket-eu-ouest-2	eu-west-2
Europe (Paris)	codepipeline-cloudtrail-placeholder-bucket-eu-ouest-3	eu-west-3
Europe (Stockholm)	codepipeline-cloudtrail-placeholder-bucket-eu-nord-1	eu-north-1

Nom de la région	Nom du compartiment d'espace réservé	Identifiant de région
Asie-Pacifique (Hong Kong)	codepipeline-cloudtrail-placeholder-bucket-ap-est-1	ap-east-1
Asie-Pacifique (Hyderabad)	codepipeline-cloudtrail-placeholder-bucket-ap-sud-2	ap-south-2
Asie-Pacifique (Jakarta)	codepipeline-cloudtrail-placeholder-bucket-ap-southeast-3	ap-southeast-3
Asie-Pacifique (Melbourne)	codepipeline-cloudtrail-placeholder-bucket-ap-southeast-4	ap-southeast-4
Asie-Pacifique (Mumbai)	codepipeline-cloudtrail-placeholder-bucket-ap-sud-1	ap-south-1
Asie-Pacifique (Osaka)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-3-prod	ap-northeast-3
Asie-Pacifique (Tokyo)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-1	ap-northeast-1
Asie-Pacifique (Séoul)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-2	ap-northeast-2
Asie-Pacifique (Singapour)	codepipeline-cloudtrail-placeholder-bucket-ap-sud-est 1	ap-southeast-1
Asie-Pacifique (Sydney)	codepipeline-cloudtrail-placeholder-bucket-ap-sud-est 2	ap-southeast-2

Nom de la région	Nom du compartiment d'espace réservé	Identifiant de région
Asie-Pacifique (Tokyo)	codepipeline-cloudtrail-placeholder-bucket-ap-northeast-1	ap-northeast-1
Canada (Centre)	codepipeline-cloudtrail-placeholder-bucket-ca-central-1	ca-central-1
Europe (Francfort)	codepipeline-cloudtrail-placeholder-bucket-eu-central-1	eu-central-1
Europe (Irlande)	codepipeline-cloudtrail-placeholder-bucket-eu-ouest-1	eu-west-1
Europe (Londres)	codepipeline-cloudtrail-placeholder-bucket-eu-ouest-2	eu-west-2
Europe (Milan)	codepipeline-cloudtrail-placeholder-bucket-eu-sud-1	eu-south-1
Europe (Paris)	codepipeline-cloudtrail-placeholder-bucket-eu-ouest-3	eu-west-3
Europe (Espagne)	codepipeline-cloudtrail-placeholder-bucket-eu-sud-2	eu-south-2
Europe (Stockholm)	codepipeline-cloudtrail-placeholder-bucket-eu-nord-1	eu-north-1
Europe (Zurich) *	codepipeline-cloudtrail-placeholder-bucket-eu-central-2	eu-central-2
Israël (Tel Aviv)	codepipeline-cloudtrail-placeholder-bucket-il-central-1	il-central-1
Moyen-Orient (Bahreïn) *	codepipeline-cloudtrail-placeholder-bucket-moi-sud-1	me-south-1

Nom de la région	Nom du compartiment d'espace réservé	Identifiant de région
Moyen-Orient (EAU)	codepipeline-cloudtrail-placeholder-bucket-moi-central-1	me-central-1
Amérique du Sud (São Paulo)	codepipeline-cloudtrail-placeholder-bucket-sa-est-1	sa-east-1

Journalisation des appels d' CodePipeline API avec AWS CloudTrail

AWS CodePipeline est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un Service AWS utilisateur CodePipeline. CloudTrail capture tous les appels d'API en CodePipeline tant qu'événements. Les appels capturés incluent les appels depuis la CodePipeline console et les appels de code vers les opérations de l' CodePipeline API. Si vous créez un historique, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris des événements pour CodePipeline. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents sur la CloudTrail console dans l'historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite CodePipeline, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des informations supplémentaires.

Pour en savoir plus CloudTrail, consultez le Guide de AWS CloudTrail l'utilisateur.

CodePipeline informations dans CloudTrail

CloudTrail est activé Compte AWS lors de la création de votre compte. Lorsqu'une activité se produit CodePipeline, cette activité est enregistrée dans un CloudTrail événement avec d'autres Service AWS événements dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents sur votre AWS compte. Pour plus d'informations, consultez la section Affichage des événements avec l'historique des CloudTrail événements.

Pour un enregistrement continu des événements qui vous concernent Compte AWS, y compris ceux qui vous CodePipeline concernent, créez un historique. Un journal permet CloudTrail de transmettre des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un historique dans la console, celui-ci s'applique à toutes les AWS régions. Le journal enregistre les événements

de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez en configurer d'autres Services AWS pour analyser plus en profondeur les données d'événements collectées dans les CloudTrail journaux et y donner suite. Pour plus d'informations, consultez les ressources suivantes :

- Vue d'ensemble de la création d'un journal d'activité
- CloudTrail Services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux provenant de plusieurs régions et réception de fichiers CloudTrail journaux provenant de plusieurs comptes

Toutes les CodePipeline actions sont enregistrées CloudTrail et documentées dans la référence de l'CodePipeline API. Par exemple, les appels aux `CreatePipeline` `UpdatePipeline` actions `GetPipelineExecution` et les actions génèrent des entrées dans les fichiers CloudTrail journaux.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été faite avec des informations d'identification root ou Gestion des identités et des accès AWS (IAM).
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la requête a été effectuée par un autre Service AWS.

Pour plus d'informations, consultez la section Élément userIdentity CloudTrail .

Comprendre les entrées des fichiers CodePipeline journaux

Un journal est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne sont pas une pile ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal pour un événement de pipeline de mise à jour, dans lequel un pipeline nommé MyFirstPipeline a été modifié par l'utilisateur dont le

nom JaneDoe-CodePipeline de compte est 80398EXAMPLE. L'utilisateur a changé le nom de l'étape source d'un pipeline, de `Source` à `MySourceStage`. Les éléments `requestParameters` et `responseElements` du journal CloudTrail contiennent toute la structure du pipeline modifié, par conséquent ils ont été abrégés dans l'exemple suivant. L'accent a été mis sur la portion `requestParameters` du pipeline où la modification s'est produite, le numéro de version précédente du pipeline, et la partie `responseElements`, qui affiche le numéro de version incrémenté d'un niveau. Les parties modifiées sont marquées d'ellipses (...) pour mettre en évidence l'endroit où les données sont le plus nombreuses dans une entrée de journal réel.

```
{
 "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 métriques

Vous pouvez utiliser des métriques CloudWatch pour mesurer les CodePipeline pipelines. Vous pouvez utiliser les mesures suivantes pour mesurer la durée du pipeline et les échecs d'exécution du pipeline.

Vous pouvez surveiller vos pipelines à deux niveaux :

Niveau du pipeline

Ces indicateurs se situent au niveau du pipeline. Choisissez Par pipeline dans CloudWatch. Les pipelines disponibles seront répertoriés dans CloudWatch.

AWS niveau du compte

Ces statistiques concernent tous les pipelines d'un compte. Pour voir les statistiques au niveau du AWS compte, sélectionnez Mesures du compte dans CloudWatch.

Pour plus d'informations sur l'affichage des métriques, consultez la section Afficher les métriques disponibles dans le guide de CloudWatch l'utilisateur Amazon.

Rubriques

- PipelineDuration
- FailedPipelineExecutions

PipelineDuration

La `PipelineDuration` métrique mesure la durée d'exécution du pipeline. Cette métrique n'est disponible que pour le niveau du pipeline.

Unités : secondes

FailedPipelineExecutions

La `FailedPipelineExecutions` métrique mesure le nombre d'exécutions de pipeline qui ont échoué. Cette métrique est disponible au niveau du pipeline et au niveau du compte.

Cette métrique inclut un décompte distinct pour les exécutions de pipeline avec une nouvelle tentative d'une action ayant échoué. En d'autres termes, lorsqu'une action ayant échoué est réessayée dans un pipeline, cela sera considéré comme une métrique d'exécution de pipeline distincte. Par exemple, pour un pipeline avec une action source S3 où l'action source S3 échoue la première fois, cela effectue une exécution, puis l'action source est réessayée et échoue à nouveau. Pour cet exemple, la métrique sera la suivante :

```
FailedPipelineExecutionAttempts: 2
```

Unités : nombre

Résolution des problèmes CodePipeline

Les informations suivantes peuvent vous aider à résoudre les problèmes courants dans AWS CodePipeline.

Rubriques

- Erreur de pipeline : un pipeline configuré avec AWS Elastic Beanstalk renvoie un message d'erreur : « Le déploiement a échoué. Le rôle fourni ne dispose pas d'autorisations suffisantes : Service:AmazonElasticLoadBalancing »
- Erreur de déploiement : un pipeline configuré avec un AWS Elastic Beanstalk l'action de déploiement se bloque au lieu d'échouer si l'autorisation « DescribeEvents » est manquante
- Erreur de pipeline : une action source renvoie le message d'autorisations insuffisantes : « Impossible d'accéder au nom du CodeCommit référentiel du référentiel. Assurez-vous que le rôle IAM du pipeline dispose des autorisations suffisantes pour accéder au référentiel. »
- Erreur de pipeline : Une action Jenkins de génération ou de test s'exécute pendant une longue durée puis échoue, en raison d'informations d'identification ou d'autorisations insuffisantes
- Erreur de pipeline : un pipeline créé en un AWS Région utilisant un bucket créé dans un autre AWS La région renvoie un « InternalError » avec le code « JobFailed »
- Erreur de déploiement : un fichier ZIP contenant un fichier WAR a été déployé avec succès sur AWS Elastic Beanstalk, mais l'URL de l'application indique une erreur 404 introuvable
- Les noms des dossiers d'artefact du pipeline semblent tronqués
- Ajoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com
- Ajouter CodeBuild GitClone des autorisations pour les actions CodeCommit source
- Erreur de pipeline : un déploiement avec l' CodeDeployToECS action renvoie un message d'erreur : « Exception lors de la tentative de lecture du fichier d'artefact de définition de tâche depuis : nom de l'artefact < source » >
- GitHub (via l'application OAuth) action source : la liste des référentiels montre les différents référentiels
- GitHub Action source (via GitHub l'application) : impossible de terminer la connexion pour un référentiel
- Erreur Amazon S3 : l'accès au compartiment S3 > est refusé à l'<ARN du rôle de CodePipeline service ARN < BucketName >

- Les pipelines dotés d'un Amazon S3, d'Amazon ECR ou d' CodeCommit une source ne démarrent plus automatiquement
- Erreur de connexion lors de la connexion à GitHub : « Un problème est survenu, assurez-vous que les cookies sont activés dans votre navigateur » ou « Le propriétaire d'une organisation doit installer l' GitHub application »
- Les pipelines dont le mode d'exécution est passé en mode QUEUED ou PARALLEL échouent lorsque la limite d'exécution est atteinte
- Les pipelines en mode PARALLÈLE ont une définition de pipeline obsolète s'ils sont modifiés lors du passage en mode QUEUED ou SUPERSEDED
- Les pipelines passés du mode PARALLÈLE afficheront un mode d'exécution précédent
- Les pipelines avec des connexions qui utilisent le filtrage des déclencheurs par chemin de fichier peuvent ne pas démarrer lors de la création de la branche
- Les pipelines dont les connexions utilisent le filtrage des déclencheurs par chemin de fichier risquent de ne pas démarrer lorsque la limite de fichiers est atteinte
- CodeCommit ou les révisions de source S3 en mode PARALLÈLE peuvent ne pas correspondre à EventBridge l'événement
- L'action de déploiement EC2 échoue avec un message d'erreur Aucun fichier de ce type
- L'action EKS Deploy échoue avec un message d'erreur indiquant que le cluster est inaccessible
- Le pipeline ne se déclenche pas pour toutes les branches lorsque plusieurs actions de source font référence au même référentiel
- Besoin d'aide pour résoudre un autre problème ?

Erreur de pipeline : un pipeline configuré avec AWS Elastic Beanstalk renvoie un message d'erreur : « Le déploiement a échoué. Le rôle fourni ne dispose pas d'autorisations suffisantes : Service:AmazonElasticLoadBalancing »

Problème : le rôle de service pour CodePipeline ne dispose pas d'autorisations suffisantes pour AWS Elastic Beanstalk, notamment, mais sans s'y limiter, certaines opérations dans Elastic Load Balancing. Le rôle de service pour CodePipeline a été mis à jour le 6 août 2015 pour résoudre ce problème. Les clients ayant créé leur rôle de service avant cette date doivent modifier la déclaration de stratégie de leur rôle de service afin d'ajouter les autorisations requises.

Correctifs possibles : la solution la plus simple consiste à modifier la déclaration de stratégie pour votre rôle de service, comme indiqué dans Ajouter des autorisations au rôle CodePipeline de service.

Après avoir appliqué la politique modifiée, suivez les étapes décrites pour Lancement manuel d'un pipeline réexécuter manuellement tous les pipelines utilisant Elastic Beanstalk.

Vous pouvez modifier les autorisations par différents moyens, selon vos besoins en termes de sécurité.

Erreur de déploiement : un pipeline configuré avec un AWS Elastic Beanstalk l'action de déploiement se bloque au lieu d'échouer si l'autorisation « DescribeEvents » est manquante

Problème : le rôle de service pour CodePipeline doit inclure l'`"elasticbeanstalk:DescribeEvents"`action pour tous les pipelines qui l'utilisent AWS Elastic Beanstalk. Sans cette autorisation, les actions de AWS Elastic Beanstalk déploiement se bloquent sans échouer ni indiquer d'erreur. Si cette action est absente de votre rôle de service, cela CodePipeline signifie qu'il n'est pas autorisé à exécuter la phase de déploiement du pipeline AWS Elastic Beanstalk en votre nom.

Correctifs possibles : passez en revue votre rôle CodePipeline de service. Si l'action `"elasticbeanstalk:DescribeEvents"` n'en fait pas partie, utilisez la procédure indiquée dans Ajouter des autorisations au rôle CodePipeline de service pour l'ajouter à l'aide de la fonction Edit Policy (Modifier la stratégie) dans la console IAM.

Après avoir appliqué la politique modifiée, suivez les étapes décrites pour Lancement manuel d'un pipeline réexécuter manuellement tous les pipelines utilisant Elastic Beanstalk.

Erreur de pipeline : une action source renvoie le message d'autorisations insuffisantes : « Impossible d'accéder au nom du CodeCommit référentiel du **`référentiel`**. Assurez-vous que le rôle IAM du pipeline dispose des autorisations suffisantes pour accéder au référentiel. »

Problème : le rôle de service pour CodePipeline ne dispose pas d'autorisations suffisantes CodeCommit et a probablement été créé avant l'ajout de la prise en charge de l'utilisation

CodeCommit des référentiels le 18 avril 2016. Les clients ayant créé leur rôle de service avant cette date doivent modifier la déclaration de stratégie de leur rôle de service afin d'ajouter les autorisations requises.

Correctifs possibles : ajoutez les autorisations requises CodeCommit pour la politique CodePipeline de votre rôle de service. Pour plus d'informations, consultez Ajouter des autorisations au rôle CodePipeline de service.

Erreur de pipeline : Une action Jenkins de génération ou de test s'exécute pendant une longue durée puis échoue, en raison d'informations d'identification ou d'autorisations insuffisantes

Problème : si le serveur Jenkins est installé sur une instance Amazon EC2, l'instance n'a peut-être pas été créée avec un rôle d'instance disposant des autorisations requises pour. CodePipeline Si vous utilisez un utilisateur IAM sur un serveur Jenkins, une instance sur site ou une instance Amazon EC2 créée sans le rôle IAM requis, soit l'utilisateur IAM ne dispose pas des autorisations requises, soit le serveur Jenkins ne peut pas accéder à ces informations d'identification via le profil configuré sur le serveur.

Correctifs possibles : assurez-vous que le rôle d'instance Amazon EC2 ou l'utilisateur IAM est configuré avec la politique `AWSCodePipelineCustomActionAccess` gérée ou avec les autorisations équivalentes. Pour plus d'informations, consultez AWS politiques gérées pour AWS CodePipeline.

Si vous utilisez un utilisateur IAM, assurez-vous que le AWS profil configuré sur l'instance utilise l'utilisateur IAM configuré avec les autorisations appropriées. Vous devrez peut-être fournir les informations d'identification utilisateur IAM que vous avez configurées pour l'intégration entre Jenkins et CodePipeline directement dans l'interface utilisateur de Jenkins. Ce n'est pas recommandé. Si vous devez le faire, assurez-vous que le serveur Jenkins est sécurisé et utilise le protocole HTTPS au lieu de HTTP.

Erreur de pipeline : un pipeline créé en un AWS Région utilisant un bucket créé dans un autre AWS La région renvoie un « InternalError » avec le code « JobFailed »

Problème : le téléchargement d'un artefact stocké dans un compartiment Amazon S3 échouera si le pipeline et le compartiment sont créés dans des AWS régions différentes.

Corrections possibles : assurez-vous que le compartiment Amazon S3 dans lequel votre artefact est stocké se trouve dans la même AWS région que le pipeline que vous avez créé.

Erreur de déploiement : un fichier ZIP contenant un fichier WAR a été déployé avec succès sur AWS Elastic Beanstalk, mais l'URL de l'application indique une erreur 404 introuvable

Problème : un fichier WAR est déployé avec succès dans un environnement AWS Elastic Beanstalk , mais l'URL de l'application renvoie une erreur « 404 - Non trouvé ».

Corrections AWS Elastic Beanstalk possibles : possibilité de décompresser un fichier ZIP, mais pas un fichier WAR contenu dans un fichier ZIP. Au lieu de spécifier un fichier WAR dans votre fichier `buildspec.yml`, spécifiez un dossier contenant le contenu à déployer. Par exemple :

```
version: 0.2

phases:
  post_build:
    commands:
      - mvn package
      - mv target/my-web-app ./
artifacts:
  files:
    - my-web-app/**/*
 discard-paths: yes
```

Pour obtenir un exemple, consultez Exemple AWS Elastic Beanstalk pour CodeBuild.

Les noms des dossiers d'artefact du pipeline semblent tronqués

Problème : lorsque vous affichez les noms des artefacts du pipeline dans CodePipeline, ils semblent tronqués. Les noms peuvent sembler similaires ou ne plus contenir l'intégralité du nom du pipeline.

Explication : CodePipeline tronque les noms des artefacts pour garantir que le chemin complet d'Amazon S3 ne dépasse pas les limites de taille fixées par la politique lors de la génération d'informations d' CodePipeline identification temporaires pour les travailleurs.

Même si le nom de l'artefact semble tronqué, il est CodePipeline mappé vers le compartiment d'artefacts d'une manière qui n'est pas affectée par les artefacts dont le nom est tronqué. Le pipeline peut fonctionner normalement. Ce n'est pas un problème avec le dossier ou les artefacts. Les noms de pipeline sont limités à 100 caractères. Bien que le nom du dossier de l'artefact puisse apparaître raccourci, il est toujours unique à votre pipeline.

Ajoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com

Lorsque vous utilisez une action AWS CodeConnections dans une source et une CodeBuild action, l'artefact d'entrée peut être transmis au build de deux manières :

- Par défaut : l'action source produit un fichier zip contenant le code à CodeBuild télécharger.
- Clone complet : le code source peut être directement téléchargé dans l'environnement de construction.

  Le mode de clonage complet vous permet d'interagir avec le code source en tant que dépôt Git fonctionnel. Pour utiliser ce mode, vous devez autoriser votre CodeBuild environnement à utiliser la connexion.

Pour ajouter des autorisations à votre politique CodeBuild de rôle de service, vous créez une politique gérée par le client que vous associez à votre rôle de CodeBuild service. Les étapes suivantes créent une politique dans laquelle l'`UseConnection`autorisation est spécifiée dans le `action` champ, l'ARN de connexion est spécifié dans le `Resource` champ et l'ID du référentiel source est limité via`Condition`.

Pour utiliser la console pour ajouter les UseConnection autorisations

1. Pour trouver l'ARN de connexion et l'ID du référentiel source de votre pipeline, ouvrez votre pipeline, cliquez sur l'icône (i) sur votre action source et passez à l'onglet Entrée.

   Voici un exemple d'ARN de connexion :

   ```
   arn:aws:codeconnections:eu-central-1:123456789123:connection/
   sample-1908-4932-9ecc-2ddacee15095
   ```

   Exemple de l'ID du référentiel source :

   ```
   owner/test-app
   ```

   Vous ajoutez l'ARN de connexion et l'ID du référentiel à votre politique CodeBuild de rôle de service.
2. Pour trouver votre rôle CodeBuild de service, choisissez le projet de construction utilisé dans votre pipeline et accédez à l'onglet Détails de la construction.
3. Sélectionnez le lien Rôle de service. Cela ouvre la console IAM où vous pouvez ajouter une nouvelle stratégie qui accorde l'accès à votre connexion.
4. Dans la console IAM, choisissez Ajouter des autorisations, puis choisissez Create inline policy.

   Utilisez l'exemple de modèle de stratégie suivant. Ajoutez votre ARN de connexion dans le `Resource` champ et votre ID `codeconnections:FullRepositoryId` de référentiel dans le `Condition` champ, comme indiqué dans cet exemple :

   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"
   ```

```
                }
            }
        }
    ]
}
```

Utilisez le `Condition` champ pour réduire davantage les autorisations de votre politique en fonction des exigences de vos spécifications de construction (voir la documentation `CodeConnection` des conditions).

Sous l'onglet JSON, collez votre stratégie.

5. Choisissez Suivant. Entrez un nom pour la stratégie (par exemple, **`connection-permissions`**), puis choisissez Créer une stratégie.

   Vous verrez une **`connection-permissions`** politique associée à votre rôle, des politiques d'autorisations.

Ajouter CodeBuild GitClone des autorisations pour les actions CodeCommit source

Lorsque votre pipeline comporte une action CodeCommit source, vous pouvez transmettre l'artefact d'entrée au build de deux manières :

- Par défaut — L'action source produit un fichier zip contenant le code à CodeBuild télécharger.
- Clone complet — Le code source peut être directement téléchargé dans l'environnement de construction.

  L'option Full clone vous permet d'interagir avec le code source en tant que dépôt Git fonctionnel. Pour utiliser ce mode, vous devez ajouter des autorisations permettant à votre CodeBuild environnement d'extraire des données de votre référentiel.

Pour ajouter des autorisations à votre politique CodeBuild de rôle de service, vous créez une politique gérée par le client que vous associez à votre rôle de CodeBuild service. Les étapes suivantes créent une politique qui spécifie l'`codecommit:GitPull`autorisation dans le `action` champ.

Pour utiliser la console pour ajouter les GitPull autorisations

1. Pour trouver votre rôle CodeBuild de service, ouvrez le projet de construction utilisé dans votre pipeline et accédez à l'onglet Détails de la construction.
2. Sélectionnez le lien Rôle de service. Cela ouvre la console IAM dans laquelle vous pouvez ajouter une nouvelle politique qui accorde l'accès à votre référentiel.
3. Dans la console IAM, choisissez Attach policies (Attacher des stratégies), puis Créer une stratégie.
4. Dans l'onglet JSON, collez l'exemple de politique suivant.

```
{
    "Action": [
        "codecommit:GitPull"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
```

5. Sélectionnez Examiner une politique. Entrez un nom pour la stratégie (par exemple, **`codecommit-gitpull`**), puis choisissez Créer une stratégie.
6. Revenez à la page où vous attachiez les autorisations, actualisez la liste des stratégies et sélectionnez la stratégie que vous venez de créer. Choisissez Attacher des politiques.

Erreur de pipeline : un déploiement avec l' CodeDeployToECS action renvoie un message d'erreur : « Exception lors de la tentative de lecture du fichier d'artefact de définition de tâche depuis : nom de l'artefact < source » >

Problème :

Le fichier de définition de tâche est un artefact obligatoire pour l'action de CodePipeline déploiement sur Amazon ECS via CodeDeploy (l'`CodeDeployToECS`action). La taille maximale du fichier ZIP d'artefact dans l'action de `CodeDeployToECS` déploiement est de 3 Mo. Le message d'erreur suivant est renvoyé lorsque le fichier est introuvable ou que la taille de l'artefact dépasse 3 Mo :

Exception while trying to read the task definition artifact file from: <nom artefact source> (Exception lors de la tentative de lecture du fichier d'artefact de définition de tâche à partir de : <nom artefact source>)

Corrections possibles : Assurez-vous que le fichier de définition de tâche est inclus en tant qu'artefact. Si le fichier existe déjà, assurez-vous que la taille compressée est inférieure à 3 Mo.

GitHub (via l'application OAuth) action source : la liste des référentiels montre les différents référentiels

Problème :

Après une autorisation réussie pour une action GitHub (via l'application OAuth) dans la CodePipeline console, vous pouvez choisir parmi la liste de vos GitHub référentiels. Si la liste n'inclut pas les référentiels que vous vous attendiez à voir, vous pouvez résoudre les problèmes liés au compte utilisé pour l'autorisation.

Corrections possibles : La liste des référentiels fournie dans la CodePipeline console est basée sur l' GitHub organisation à laquelle appartient le compte autorisé. Vérifiez que le compte que vous utilisez pour l'autorisation GitHub est le compte associé à l' GitHub organisation dans laquelle votre référentiel est créé.

GitHub Action source (via GitHub l'application) : impossible de terminer la connexion pour un référentiel

Problème :

Étant donné qu'une connexion à un GitHub référentiel utilise le AWS connecteur pour GitHub, vous devez disposer des autorisations du propriétaire de l'organisation ou des autorisations d'administrateur sur le référentiel pour créer la connexion.

Corrections possibles : pour plus d'informations sur les niveaux d'autorisation pour un GitHub référentiel, consultez https://docs.github.com/en/free-pro-team@ latest/github teams/permission / setting-up-and-managing-organizations-and- -levels-for-an-organization.

Erreur Amazon S3 : l'accès au compartiment S3 > est refusé à l'<ARN du rôle de CodePipeline service ARN < BucketName >

Problème :

En cours d'exécution, l' CodeCommit action CodePipeline vérifie que le bucket d'artefacts du pipeline existe. Si l'action n'est pas autorisée à être vérifiée, une `AccessDenied` erreur se produit dans Amazon S3 et le message d'erreur suivant s'affiche dans CodePipeline :

CodePipeline le rôle de service « arn:aws:iam : : *`AccountID`* : role/service -role/ *`RoleID`* » se voit refuser l'accès S3 pour le compartiment S3. » *`BucketName`*

Les CloudTrail journaux de l'action enregistrent également l'`AccessDenied`erreur.

Corrections possibles : Procédez comme suit :

- Pour la politique associée à votre rôle CodePipeline de service, `s3:ListBucket` ajoutez-la à la liste des actions de votre stratégie. Pour obtenir des instructions sur la façon de consulter votre politique de rôle de service, consultezAfficher l'ARN du pipeline et l'ARN du rôle de service (console). Modifiez la déclaration de politique relative à votre rôle de service, comme indiqué dansAjouter des autorisations au rôle CodePipeline de service.
- Pour la politique basée sur les ressources attachée au compartiment d'artefacts Amazon S3 pour votre pipeline, également appelée politique de compartiment d'artefacts, ajoutez une déclaration autorisant l'utilisation de l'`s3:ListBucket`autorisation par votre rôle de service. CodePipeline

  Pour ajouter votre politique au compartiment d'artefacts

  1. Suivez les étapes décrites Afficher l'ARN du pipeline et l'ARN du rôle de service (console) pour choisir votre compartiment d'artefacts sur la page des paramètres du pipeline, puis consultez-le dans la console Amazon S3.
  2. Choisissez Permissions.
  3. Sous Politique de compartiment, choisissez Modifier.
  4. Dans le champ de texte Politique, entrez une nouvelle politique de compartiment ou modifiez la politique existante comme indiqué dans l'exemple suivant. La politique du bucket est un fichier JSON, vous devez donc saisir un JSON valide.

     L'exemple suivant montre une déclaration de politique de compartiment pour un compartiment d'artefacts où l'exemple d'ID de rôle pour le rôle de service est*`AROAEXAMPLEID`*.

```
{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::BucketName",
    "Condition": {
        "StringLike": {
            "aws:userid": "AROAEXAMPLEID:*"
        }
    }
}
```

L'exemple suivant montre la même déclaration de politique de compartiment après l'ajout de l'autorisation.

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
                }
            }
        }
    ]
}
```

Pour plus d'informations, suivez la procédure de la section https://aws.amazon.com/blogs/security/writing-iam-policies-how-to-grant-access-to-an-amazon-s3-bucket/.

5. Cliquez sur Enregistrer.

Après avoir appliqué la politique modifiée, suivez les étapes décrites Lancement manuel d'un pipeline pour réexécuter manuellement votre pipeline.

Les pipelines dotés d'un Amazon S3, d'Amazon ECR ou d' CodeCommit une source ne démarrent plus automatiquement

Problème :

Après avoir modifié les paramètres de configuration d'une action qui utilise des règles d'événements (EventBridgeou des CloudWatch événements) pour détecter les modifications, la console risque de ne pas détecter de modification lorsque les identificateurs de déclencheurs de la source sont similaires et comportent des caractères initiaux identiques. La nouvelle règle d'événement n'étant pas créée par la console, le pipeline ne démarre plus automatiquement.

Un exemple de modification mineure à la fin du nom du paramètre pour CodeCommit serait de changer le nom de votre CodeCommit branche `MyTestBranch-1` en`MyTestBranch-2`. Comme la modification se trouve à la fin du nom de la branche, il est possible que la règle d'événement pour

l'action source ne soit pas mise à jour ou ne crée pas de règle pour les nouveaux paramètres de source.

Cela s'applique aux actions source qui utilisent les événements CWE pour détecter les modifications, comme suit :

Action à la source	Paramètres/identifiants de déclenchement (console)
Amazon ECR	Nom du référentiel Balise d'image
Amazon S3	Compartiment Clé d'objet S3
CodeCommit	Nom du référentiel Nom de la succursale

Correctifs possibles :

Effectuez l'une des actions suivantes :

- Modifiez les paramètres CodeCommit/S3/ECR de configuration afin que des modifications soient apportées à la partie initiale de la valeur du paramètre.

  Exemple : remplacez le nom `release-branch` de votre succursale par`2nd-release-branch`. Évitez de modifier la fin du nom, par exemple`release-branch-2`.
- Modifiez les paramètres CodeCommit/S3/ECR de configuration pour chaque pipeline.

  Exemple : remplacez le nom `myRepo/myBranch` de votre succursale par`myDeployRepo/myDeployBranch`. Évitez de modifier la fin du nom, par exemple`myRepo/myBranch`*`2`*.
- Au lieu de la console, utilisez la CLI ou CloudFormation pour créer et mettre à jour vos règles relatives aux événements de détection des modifications. Pour obtenir des instructions sur la création de règles d'événement pour une action source S3, consultezConnexion à la source Amazon S3 : actions qui utilisent EventBridge et AWS CloudTrail. Pour obtenir des instructions sur la création de règles d'événement pour une action Amazon ECR, consultezActions et ressources

relatives aux sources Amazon ECR EventBridge. Pour obtenir des instructions sur la création de règles d'événement pour une CodeCommit action, consultezCodeCommit actions à la source et EventBridge.

Après avoir modifié la configuration de vos actions dans la console, acceptez les ressources de détection des modifications mises à jour créées par la console.

Erreur de connexion lors de la connexion à GitHub : « Un problème est survenu, assurez-vous que les cookies sont activés dans votre navigateur » ou « Le propriétaire d'une organisation doit installer l' GitHub application »

Problème :

Pour créer la connexion pour une action GitHub source dans CodePipeline, vous devez être le propriétaire de GitHub l'organisation. Pour les référentiels qui ne font pas partie d'une organisation, vous devez en être le propriétaire. Lorsqu'une connexion est créée par une personne autre que le propriétaire de l'organisation, une demande est créée pour le propriétaire de l'organisation et l'une des erreurs suivantes s'affiche :

Un problème est survenu, assurez-vous que les cookies sont activés dans votre navigateur

OU

Le propriétaire d'une organisation doit installer l' GitHub application

Correctifs possibles : pour les référentiels d'une GitHub organisation, le propriétaire de l'organisation doit créer la connexion au GitHub référentiel. Pour les référentiels qui ne font pas partie d'une organisation, vous devez être le propriétaire du référentiel.

Les pipelines dont le mode d'exécution est passé en mode QUEUED ou PARALLEL échouent lorsque la limite d'exécution est atteinte

Problème : le nombre maximum d'exécutions simultanées pour un pipeline en mode QUEUED est de 50 exécutions. Lorsque cette limite est atteinte, le pipeline échoue sans message d'état.

Corrections possibles : Lorsque vous modifiez la définition du pipeline pour le mode exécution, effectuez la modification séparément des autres actions de modification.

Pour plus d'informations sur le mode d'exécution QUEUED ou PARALLEL, consultez. CodePipeline concepts

Les pipelines en mode PARALLÈLE ont une définition de pipeline obsolète s'ils sont modifiés lors du passage en mode QUEUED ou SUPERSEDED

Problème : pour les pipelines en mode parallèle, lorsque vous modifiez le mode d'exécution du pipeline sur QUEUED ou SUPERSEDED, la définition du pipeline pour le mode PARALLEL ne sera pas mise à jour. La définition de pipeline mise à jour lors de la mise à jour du mode PARALLEL n'est pas utilisée en mode SUPERSEDED ou QUEUED.

Corrections possibles : pour les pipelines en mode parallèle, lorsque vous modifiez le mode d'exécution du pipeline sur QUEUED ou SUPERSEDED, évitez de mettre à jour la définition du pipeline en même temps.

Pour plus d'informations sur le mode d'exécution QUEUED ou PARALLEL, consultez. CodePipeline concepts

Les pipelines passés du mode PARALLÈLE afficheront un mode d'exécution précédent

Problème : pour les pipelines en mode PARALLÈLE, lorsque vous modifiez le mode d'exécution du pipeline sur QUEUED ou SUPERSEDED, l'état du pipeline n'affichera pas l'état mis à jour en tant que PARALLÈLE. Si le pipeline est passé de PARALLÈLE à QUEUED ou SUPERSEDED, l'état du pipeline en mode SUPERSEDED ou QUEUED sera le dernier état connu dans l'un ou l'autre de ces modes. Si le pipeline n'a jamais été exécuté dans ce mode auparavant, l'état sera vide.

Correctifs possibles : pour les pipelines en mode parallèle, lorsque vous modifiez le mode d'exécution du pipeline sur QUEUED ou SUPERSEDED, notez que l'affichage du mode d'exécution n'affichera pas l'état PARALLEL.

Pour plus d'informations sur le mode d'exécution QUEUED ou PARALLEL, consultez. CodePipeline concepts

Les pipelines avec des connexions qui utilisent le filtrage des déclencheurs par chemin de fichier peuvent ne pas démarrer lors de la création de la branche

Description : pour les pipelines dotés d'actions source qui utilisent des connexions, telles qu'une action BitBucket source, vous pouvez configurer un déclencheur avec une configuration Git qui vous permet de filtrer par chemin de fichier pour démarrer votre pipeline. Dans certains cas, pour les pipelines dont les déclencheurs sont filtrés sur les chemins de fichiers, le pipeline peut ne pas démarrer lorsqu'une branche avec un filtre de chemin de fichier est créée pour la première fois, car cela ne permet pas à la AWS CodeConnections connexion de résoudre les fichiers modifiés. Lorsque la configuration Git du déclencheur est configurée pour filtrer les chemins de fichiers, le pipeline ne démarre pas lorsque la branche contenant le filtre vient d'être créée dans le référentiel source. Pour plus d'informations sur le filtrage des chemins de fichiers, consultezAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ».

Résultat : Par exemple, les pipelines dotés CodePipeline d'un filtre de chemin de fichier sur une branche « B » ne seront pas déclenchés lors de la création de la branche « B ». S'il n'existe aucun filtre de chemin de fichier, le pipeline démarre quand même.

Les pipelines dont les connexions utilisent le filtrage des déclencheurs par chemin de fichier risquent de ne pas démarrer lorsque la limite de fichiers est atteinte

Description : pour les pipelines dotés d'actions source qui utilisent des connexions, telles qu'une action BitBucket source, vous pouvez configurer un déclencheur avec une configuration Git qui vous permet de filtrer par chemin de fichier pour démarrer votre pipeline. CodePipeline récupère jusqu'aux 100 premiers fichiers ; par conséquent, lorsque la configuration Git du déclencheur est configurée pour filtrer les chemins de fichiers, le pipeline peut ne pas démarrer s'il y a plus de 100 fichiers. Pour plus d'informations sur le filtrage des chemins de fichiers, consultezAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ».

Résultat : par exemple, si un diff contient 150 fichiers, CodePipeline examine les 100 premiers fichiers (sans ordre particulier) pour vérifier qu'ils correspondent au filtre de chemin de fichier spécifié. Si le fichier correspondant au filtre de chemin de fichier ne figure pas parmi les 100 fichiers récupérés par CodePipeline, le pipeline ne sera pas invoqué.

CodeCommit ou les révisions de source S3 en mode PARALLÈLE peuvent ne pas correspondre à EventBridge l'événement

Description : pour les exécutions de pipeline en mode PARALLÈLE, une exécution peut commencer par la modification la plus récente, telle que la validation du CodeCommit référentiel, qui peut être différente de la modification apportée à l' EventBridge événement. Dans certains cas, lorsqu'une fraction de seconde peut s'écouler entre les validations ou les balises d'image qui démarrent le pipeline, lorsqu'un autre commit ou une autre balise d'image a été envoyé CodePipeline (par exemple, l' CodeCommit action), le commit HEAD sera cloné à ce moment-là lors de la CodePipeline réception de l'événement et du démarrage de cette exécution.

Résultat : pour les pipelines en mode PARALLÈLE avec une source CodeCommit ou S3, quelle que soit la modification qui a déclenché l'exécution du pipeline, l'action source clonera toujours le HEAD au moment de son démarrage. Par exemple, pour un pipeline en mode PARALLÈLE, un commit est poussé, ce qui démarre le pipeline pour l'exécution 1, et la seconde exécution du pipeline utilise le second commit.

L'action de déploiement EC2 échoue avec un message d'erreur **`Aucun fichier de ce type`**

Description : une fois que l'action de déploiement EC2 a décompressé les artefacts du répertoire cible sur les instances, l'action exécute le script. Si le script se trouve dans le répertoire cible mais que l'action ne parvient pas à exécuter le script, l'action échoue sur cette instance et le déploiement des instances restantes échoue.

Une erreur similaire aux messages d'erreur suivants s'affiche dans les journaux d'un déploiement dans lequel se trouvent le répertoire cible `/home/ec2-user/deploy/` et le chemin du référentiel source`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
  ```

Résultat : L'action de déploiement échoue dans le pipeline.

Corrections possibles : Pour résoudre le problème, procédez comme suit.

1. Consultez les journaux pour vérifier quelle instance a provoqué l'échec du script.
2. Remplacez directory (`cd`) par le répertoire cible de votre instance. Testez le script sur l'instance.
3. Dans votre référentiel source, modifiez le fichier de script pour supprimer les commentaires ou les commandes susceptibles d'être à l'origine du problème.

L'action EKS Deploy échoue avec un message d'erreur indiquant que le **`cluster est inaccessible`**

Description : une fois l'action de déploiement EKS exécutée, l'action échoue avec un message `cluster unreachable` d'erreur. Le message indique un problème d'accès au cluster dû à des autorisations manquantes. En fonction du type de fichier (graphique Helm ou fichiers manifestes Kubernetes), le message d'erreur s'affiche comme suit.

- Pour une action de déploiement EKS utilisant un graphique Helm, un message d'erreur similaire au message d'erreur suivant s'affiche.

  ```
  error message:

  helm upgrade --install my-release test-chart --wait
  Error: Kubernetes cluster unreachable: the server has asked for the client to provide
   credentials
  ```

- Pour une action de déploiement EKS utilisant des fichiers manifestes Kubernetes, un message d'erreur similaire au message d'erreur suivant s'affiche.

  ```
  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
  ```
  ```

Résultat : L'action de déploiement échoue dans le pipeline.

Correctifs possibles : si vous utilisez un rôle existant, le rôle de CodePipeline service doit être mis à jour avec les autorisations requises pour utiliser l'action de déploiement EKS. En outre, pour autoriser le rôle de CodePipeline service à accéder à votre cluster, vous devez ajouter une entrée d'accès à votre cluster et spécifier le rôle de service pour cette entrée d'accès.

1. Vérifiez que le rôle CodePipeline de service dispose des autorisations requises pour l'action de déploiement d'EKS. Pour la référence des autorisations, voirAutorisations relatives aux politiques de rôle de service.
2. Ajoutez une entrée d'accès à votre cluster et spécifiez le rôle de CodePipeline service pour l'accès. Pour obtenir un exemple, consultez Étape 4 : Création d'une entrée d'accès pour le rôle CodePipeline de service.

Le pipeline ne se déclenche pas pour toutes les branches lorsque plusieurs actions de source font référence au même référentiel

Problème : un pipeline comporte plusieurs actions source qui utilisent des connexions (comme un Bitbucket ou une GitLab connexion), chaque action source pointant vers une branche différente du même référentiel. GitHub Une seule des branches déclenche le pipeline lorsque vous apportez des modifications. L'abonnement au webhook de la connexion est enregistré pour la combinaison du pipeline et du référentiel, et non pour chaque branche. Par conséquent, les actions sources multiples ciblant différentes branches d'un même référentiel au sein d'un même pipeline ne sont pas prises en charge.

Correctifs possibles : utilisez un pipeline distinct pour chaque branche que vous souhaitez déclencher indépendamment.

Besoin d'aide pour résoudre un autre problème ?

Essayez les ressources suivantes :

- Contactez AWS Support.
- Posez une question sur le Forum CodePipeline.
- Demandez une augmentation de quota. Pour plus d'informations, consultez Quotas dans AWS CodePipeline.

> **Note**
>
> Les demandes d'augmentation de quota sont traitées dans un délai de deux semaines maximum.

Sécurité dans AWS CodePipeline

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez de centres de données et d'architectures réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cela comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS Cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Third-party les auditeurs testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à AWS CodePipeline, voir AWS Services concernés par programme de conformitéAWS .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris la sensibilité de vos données, les exigences de votre entreprise, ainsi que la législation et la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de son utilisation CodePipeline. Les rubriques suivantes expliquent comment procéder à la configuration CodePipeline pour atteindre vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser vos CodePipeline ressources.

Rubriques

- Protection des données dans AWS CodePipeline
- Gestion des identités et des accès pour AWS CodePipeline
- Connexion et surveillance CodePipeline
- Validation de conformité pour AWS CodePipeline
- Résilience dans AWS CodePipeline
- Sécurité de l'infrastructure dans AWS CodePipeline
- Bonnes pratiques de sécurité

Protection des données dans AWS CodePipeline

Le modèle de responsabilité AWS partagée Le modèle de s'applique à la protection des données dans. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la FAQ sur la confidentialité des données et les . Pour plus d'informations sur la protection des données en Europe, consultez le Centre du règlement général sur la protection des données (RGPD).

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec ou d'autres Services AWS utilisateurs de la console, de l'API ou AWS des SDK. AWS CLI Toutes les données que vous entrez dans des balises

ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Les bonnes pratiques de sécurité suivantes s'appliquent également à la protection des données dans CodePipeline :

- Configurer le chiffrement côté serveur pour les artefacts stockés dans Amazon S3 pour CodePipeline
- Utilisation AWS Secrets Manager pour suivre les mots de passe de base de données ou les clés d'API tierces

Confidentialité du trafic inter-réseau

Amazon VPC est un système Service AWS que vous pouvez utiliser pour lancer AWS des ressources dans un réseau virtuel (cloud privé virtuel) que vous définissez. CodePipelineprend en charge les points de terminaison Amazon VPC basés sur une AWS technologie qui facilite la communication privée entre les Services AWS utilisateurs d'une interface Elastic Network avec des adresses IP privées. AWS PrivateLink Cela signifie que vous pouvez vous connecter directement CodePipeline via un point de terminaison privé dans votre VPC, en conservant tout le trafic à l'intérieur de votre VPC et du réseau. AWS Auparavant, les applications exécutées dans un VPC nécessitaient un accès Internet pour la connexion à CodePipeline. Avec un VPC, vous contrôlez vos paramètres réseau, tels que :

- plage d'adresses IP,
- Sous-réseaux,
- tables de routage, et
- Passerelles réseau

Pour connecter votre VPC à CodePipeline, vous définissez un point de terminaison VPC d'interface pour. CodePipeline Ce type de point de terminaison vous permet de connecter votre VPC à. Services AWS Le point de terminaison fournit une connectivité fiable et évolutive CodePipeline sans nécessiter de passerelle Internet, d'instance de traduction d'adresses réseau (NAT) ou de connexion VPN. Pour plus d'informations sur la configuration d'un VPC, consultez le Guide de l'utilisateur VPC.

Chiffrement au repos

Les données entrantes CodePipeline sont cryptées au repos à l'aide de AWS KMS keys. Les artefacts de code sont stockés dans un compartiment S3 appartenant au client et chiffrés à l'aide de la clé Clé gérée par AWS ou d'une clé gérée par le client. Pour plus d'informations, consultez Configurer le chiffrement côté serveur pour les artefacts stockés dans Amazon S3 pour CodePipeline.

Chiffrement en transit

Toutes les communications de service à service sont cryptées en transit à l'aide de. SSL/TLS

Gestion des clés de chiffrement

Si vous choisissez l'option par défaut pour chiffrer les artefacts de code, CodePipeline utilise le Clé gérée par AWS. Vous ne pouvez ni le modifier ni le supprimer Clé gérée par AWS. Si vous utilisez une clé gérée par le client AWS KMS pour chiffrer ou déchiffrer des artefacts dans le compartiment S3, vous pouvez modifier ou faire pivoter cette clé gérée par le client si nécessaire.

> **Important**
>
> CodePipeline ne prend en charge que les clés KMS symétriques. N'utilisez pas de clé KMS asymétrique pour chiffrer les données de votre compartiment S3.

Configurer le chiffrement côté serveur pour les artefacts stockés dans Amazon S3 pour CodePipeline

Il existe deux manières de configurer le chiffrement côté serveur pour les artefacts Amazon S3 :

- CodePipeline crée un compartiment d'artefacts S3 et un compartiment par défaut Clé gérée par AWS lorsque vous créez un pipeline à l'aide de l'assistant de création de pipeline. Clé gérée par AWS Il est crypté avec les données des objets et géré par AWS.
- Vous pouvez créer et gérer votre propre clé gérée par le client.

> **Important**
>
> CodePipeline ne prend en charge que les clés KMS symétriques. N'utilisez pas de clé KMS asymétrique pour chiffrer les données de votre compartiment S3.

Si vous utilisez la clé S3 par défaut, vous ne pouvez ni la modifier ni la supprimer Clé gérée par AWS. Si vous utilisez une clé gérée par le client AWS KMS pour chiffrer ou déchiffrer des artefacts dans le compartiment S3, vous pouvez modifier ou faire pivoter cette clé gérée par le client si nécessaire.

Amazon S3 prend en charge les politiques de compartiment que vous pouvez utiliser si vous exigez un chiffrement côté serveur pour tous les objets stockés dans le compartiment. Par exemple, la politique de compartiment suivante refuse l'autorisation de upload object (`s3:PutObject`) à tout le monde si la demande n'inclut pas l'`x-amz-server-side-encryption`en-tête demandant le chiffrement côté serveur avec. 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"
                }
            }
        }
    ]
}
```

Pour plus d'informations sur le chiffrement côté serveur AWS KMS, voir Protection des données à l'aide du Server-Side chiffrement et Protection des données à l'aide du chiffrement côté serveur avec des clés KMS stockées dans (). AWS Key Management Service SSE-KMS

Pour plus d'informations à ce sujet AWS KMS, consultez le guide du AWS Key Management Service développeur.

Rubriques

- Consultez votre Clé gérée par AWS
- Configurez le chiffrement côté serveur pour les compartiments S3 à l'aide de CloudFormation ou le AWS CLI

Consultez votre Clé gérée par AWS

Lorsque vous utilisez l'assistant Création d'un pipeline pour créer votre premier pipeline, un compartiment S3 est créé automatiquement dans la région où vous avez créé ce pipeline. Ce compartiment permet de stocker les artefacts du pipeline. Lorsqu'un pipeline s'exécute, les artefacts sont placés dans le compartiment S3 et en sont extraits. Par défaut, CodePipeline utilise le chiffrement côté serveur à AWS KMS l'aide de la `aws/s3` clé Clé gérée par AWS pour Amazon S3. Il Clé gérée par AWS est créé et enregistré dans votre AWS compte. Lorsque des artefacts sont extraits du compartiment S3, CodePipeline utilise le même SSE-KMS processus pour déchiffrer l'artefact.

Pour consulter les informations relatives à votre Clé gérée par AWS

1. Connectez-vous à la AWS KMS console Console de gestion AWS et ouvrez-la.
2. Si une page de bienvenue apparaît, choisissez Commencer maintenant.
3. Dans le volet de navigation du service, choisissez les clés AWS gérées.

4. Choisissez la région de votre pipeline. Par exemple, si le pipeline a été créé en `us-east-2`, assurez-vous que le filtre est défini sur US East (Ohio).

   Pour plus d'informations sur les régions et les points de terminaison disponibles CodePipeline, consultez la section AWS CodePipeline Points de terminaison et quotas.

5. Dans la liste, choisissez la clé avec l'alias utilisé pour votre pipeline (par défaut, aws/s3). Les informations de base sur la clé s'affichent.

Configurez le chiffrement côté serveur pour les compartiments S3 à l'aide de CloudFormation ou le AWS CLI

Lorsque vous utilisez CloudFormation ou AWS CLI pour créer un pipeline, vous devez configurer le chiffrement côté serveur manuellement. Utilisez l'exemple de politique de compartiment ci-dessus, puis créez votre propre clé gérée par le client. Vous pouvez également utiliser vos propres clés au lieu de Clé gérée par AWS. Voici quelques raisons de choisir votre propre clé :

- Vous souhaitez effectuer une rotation de la clé sur un planning afin de répondre aux exigences relatives aux activités et à la sécurité de votre organisation.
- Vous souhaitez créer un pipeline qui utilise des ressources associées à un autre compte AWS . Cette opération nécessite l'utilisation d'une clé gérée par le client. Pour plus d'informations, consultez Créez un pipeline CodePipeline qui utilise les ressources d'un autre AWS compte.

Les bonnes pratiques de chiffrement décourage la réutilisation étendue des clés de chiffrement. La bonne pratique consiste à effectuer régulièrement une rotation de votre clé. Pour créer un nouveau matériel cryptographique pour vos AWS KMS clés, vous pouvez créer une clé gérée par le client, puis modifier vos applications ou alias afin d'utiliser la nouvelle clé gérée par le client. Vous pouvez également activer la rotation automatique des clés pour une clé gérée par le client existante.

Pour faire pivoter votre clé gérée par le client, voir Rotation des clés.

> **Important**
>
> CodePipeline ne prend en charge que les clés KMS symétriques. N'utilisez pas de clé KMS asymétrique pour chiffrer les données de votre compartiment S3.

Utilisation AWS Secrets Manager pour suivre les mots de passe de base de données ou les clés d'API tierces

Nous vous recommandons de les utiliser AWS Secrets Manager pour alterner, gérer et récupérer les informations d'identification de base de données, les clés d'API et autres secrets tout au long de leur cycle de vie. Secrets Manager vous permet de remplacer les informations d'identification codées en dur dans votre code (y compris les mots de passe) par un appel d'API à Secrets Manager pour récupérer le secret par programmation. Pour plus d'informations, consultez Qu'est-ce que AWS Secrets Manager ? dans le Guide de l'utilisateur de AWS Secrets Manager.

Pour les pipelines dans lesquels vous transmettez des paramètres secrets (tels que les informations d'identification OAuth) dans un CloudFormation modèle, vous devez inclure des références dynamiques dans votre modèle qui accèdent aux secrets que vous avez stockés dans Secrets Manager. Pour le modèle d'ID de référence et des exemples, voir Secrets Manager Secrets dans le guide de AWS CloudFormation l'utilisateur. Pour un exemple qui utilise des références dynamiques dans un extrait de modèle pour un GitHub webhook dans un pipeline, voir Configuration des ressources du webhook.

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles pour la gestion des secrets.

- Secrets Manager peut automatiquement alterner les informations d'identification de base de données, par exemple pour la rotation des secrets Amazon RDS. Pour plus d'informations, consultez Rotating Your AWS Secrets Manager Secrets Manager dans le guide de l'utilisateur de AWS Secrets Manager.
- Pour voir les instructions permettant d'ajouter des références dynamiques Secrets Manager à vos modèles CloudFormation , veuillez consulter https://aws.amazon.com/blogs/security/how-to-create-and-retrieve-secrets-managed-in-aws-secrets-manager-using-aws-cloudformation-template/.

Gestion des identités et des accès pour AWS CodePipeline

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser CodePipeline les ressources. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- Public ciblé
- Authentification par des identités
- Gestion de l'accès à l'aide de politiques
- Comment ? AWS CodePipeline fonctionne avec IAM
- AWS CodePipeline exemples de politiques basées sur l'identité
- AWS CodePipeline exemples de politiques basées sur les ressources
- Résolution des problèmes AWS CodePipeline identité et accès
- référence des autorisations
- Gérer le rôle CodePipeline de service

Public ciblé

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction de votre rôle :

- Utilisateur du service : demandez des autorisations à votre administrateur si vous ne pouvez pas accéder aux fonctionnalités (voir Résolution des problèmes AWS CodePipeline identité et accès)
- Administrateur du service : déterminez l'accès des utilisateurs et soumettez les demandes d'autorisation (voir Comment ? AWS CodePipeline fonctionne avec IAM)
- Administrateur IAM : rédigez des politiques pour gérer l'accès (voir AWS CodePipeline exemples de politiques basées sur l'identité)

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification. Google/Facebook Pour plus d'informations sur la connexion, consultez Connexion à votre Compte AWS dans le Guide de l'utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez Signature AWS Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Utilisateur racine d'un compte AWS

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez Tâches qui requièrent les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.

Les groupes IAM spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en passant d'un rôle d'utilisateur à un rôle IAM (console) ou en appelant une opération d' AWS API AWS CLI ou d'API. Pour plus d'informations, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Gestion de l'accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique définit les autorisations lorsqu'elles sont associées à une identité

ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

Identity-based politiques

Identity-based les politiques sont des documents de politique d'autorisation JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Identity-based les politiques peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités). Pour découvrir comment choisir entre des politiques gérées et en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Resource-based politiques

Resource-based les politiques sont des documents de politique JSON que vous attachez à une ressource. Les exemples incluent les politiques de confiance de rôle IAM et les stratégies de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Vous devez spécifier un principal dans une politique basée sur les ressources.

Resource-based les politiques sont des politiques intégrées qui se trouvent dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Autres types de politique

AWS prend en charge des types de politiques supplémentaires qui peuvent définir les autorisations maximales accordées par les types de politiques les plus courants :

- Limites d'autorisations : une limite des autorisations définit le nombre maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM. Pour plus d'informations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCP) : spécifient les autorisations maximales pour une organisation ou une unité organisationnelle dans AWS Organizations. Pour plus d'informations, consultez Politiques de contrôle de service dans le Guide de l'utilisateur AWS Organizations .
- Politiques de contrôle des ressources (RCP) : définissent les autorisations maximales disponibles pour les ressources de votre organisation. Pour plus d'informations, consultez Politiques de contrôle des ressources (RCP) dans le Guide de l'utilisateur AWS Organizations .
- Politiques de session : politiques avancées que vous passez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Comment ? AWS CodePipeline fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à CodePipeline, vous devez connaître les fonctionnalités IAM disponibles. CodePipeline Pour obtenir une vue d'ensemble de la manière dont IAM fonctionne CodePipeline et Services AWS des autres fonctionnalités, consultez Services AWS la section relative à l'utilisation d'IAM dans le Guide de l'utilisateur d'IAM.

Rubriques

- CodePipeline politiques basées sur l'identité
- CodePipeline politiques basées sur les ressources
- Autorisation basée sur les CodePipeline tags
- CodePipeline Rôles IAM

CodePipeline politiques basées sur l'identité

Avec les politiques basées sur l'identité IAM, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. CodePipeline prend en charge des actions, ressources et clés de condition spécifiques. Pour en savoir plus sur tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Actions

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Les actions de politique en CodePipeline cours utilisent le préfixe suivant avant l'action :`codepipeline:`.

Par exemple, pour accorder à quelqu'un l'autorisation d'afficher les pipelines existants dans le compte, vous incluez l'action `codepipeline:GetPipeline` dans sa stratégie. Les déclarations de politique doivent inclure un `NotAction` élément `Action` ou. CodePipeline définit son propre ensemble d'actions décrivant les tâches que vous pouvez effectuer avec ce service.

Pour spécifier plusieurs actions dans une seule déclaration, séparez-les par des virgules comme suit :

```
"Action": [
      "codepipeline:action1",
      "codepipeline:action2"
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Get`, incluez l'action suivante :

```
"Action": "codepipeline:Get*"
```

Pour obtenir la liste des CodePipeline actions, reportez-vous à la section Actions définies par AWS CodePipeline dans le guide de l'utilisateur IAM.

Ressources

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

ressources et opérations

Dans , la principale ressource est le pipeline. Dans votre stratégie, vous utilisez un nom Amazon Resource Name (ARN) pour identifier la ressource à laquelle la stratégie s'applique. prend en charge d'autres ressources qui peuvent être utilisées avec la ressource principale, telles que les étapes, les actions et les actions personnalisées. Celles-ci sont appelées sous-ressources. Des noms Amazon Resource Name (ARN) sont associés à ces ressources et sous-ressources. Pour plus d'informations sur les ARN, consultez Amazon Resource Names (ARN) et les Service AWS espaces de noms dans le. Référence générale d'Amazon Web Services Pour obtenir l'ARN du pipeline associé à votre pipeline, vous pouvez le trouver sous Paramètres de la console. Pour plus d'informations, consultez Afficher l'ARN du pipeline et l'ARN du rôle de service (console).

Type de ressource	Format ARN
Pipeline	arn:aws:codepipeline : : *region account pipeline-name*
Étape	arn:aws:codepipeline : : :/*regionaccountpipeline-name stage-name*
Action	arn:aws:codepipeline : : :/*regionaccountpipeline-name stage-name action-name*
Action personnalisée	arn:aws:codepipeline : ::actiont ype :///*regionaccountownercategoryproviderversion*
Toutes les ressources	arn:aws:codepipeline : *
Toutes les ressources appartenant au compte spécifié dans la région indiquée	arn:aws:codepipeline : : : * *region account*

> **Note**
>
> La plupart des services AWS traitent deux points (:)) ou une barre oblique (/) comme le même caractère dans les ARN. Cependant, recherche une correspondance exacte dans les modèles et règles d'événements. Veillez à utiliser les caractères ARN corrects lors de la création de modèles d'événements afin qu'ils correspondent à la syntaxe ARN dans le pipeline que vous souhaitez faire correspondre.

Dans, certains appels d'API prennent en charge les autorisations au niveau des ressources. Resource-level les autorisations indiquent si un appel d'API peut spécifier un ARN de ressource ou si l'appel d'API peut uniquement spécifier toutes les ressources à l'aide du caractère générique. Consultez référence des autorisations pour une description détaillée des autorisations au niveau des ressources et une liste des appels d' CodePipeline API qui prennent en charge les autorisations au niveau des ressources.

Par exemple, vous pouvez indiquer un pipeline (*myPipeline*) spécifique dans votre instruction à l'aide de son ARN comme suit :

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:myPipeline"
```

Vous pouvez aussi spécifier tous les pipelines qui appartiennent à un compte spécifique à l'aide du caractère générique (*) comme suit :

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:*"
```

Pour spécifier toutes les ressources, ou si une action d'API spécifique ne prend pas en charge les ARN, utilisez le caractère générique (*) dans l'élément `Resource`, comme suit :

```
"Resource": "*"
```

> **Note**
>
> Lorsque vous créez des politiques IAM, suivez les conseils de sécurité standard qui consistent à accorder le moindre privilège, c'est-à-dire à n'accorder que les autorisations nécessaires à l'exécution d'une tâche. Si un appel d'API prend en charge les ARN, il prend

en charge les autorisations au niveau des ressources et vous n'avez pas besoin d'utiliser le caractère générique (*).

Certains appels d'API acceptent plusieurs ressources (par exemple,`GetPipeline`). Pour spécifier plusieurs ressources dans une seule déclaration, séparez leurs ARN par des virgules comme suit :

```
"Resource": ["arn1", "arn2"]
```

fournit un ensemble d'opérations permettant de travailler avec les ressources. Pour obtenir la liste des opérations disponibles, consultez référence des autorisations.

Clés de condition

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` indique à quel moment les instructions s'exécutent en fonction de critères définis. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

CodePipeline définit son propre ensemble de clés de condition et prend également en charge l'utilisation de certaines clés de condition globales. Pour voir toutes les clés de condition AWS globales, consultez la section Clés contextuelles de condition AWS globale dans le guide de l'utilisateur IAM.

Toutes les actions Amazon EC2 prennent en charge les clés de condition `aws:RequestedRegion` et `ec2:Region`. Pour plus d'informations, consultez Exemple : restriction de l'accès à une région spécifique.

Pour consulter la liste des clés de CodePipeline condition, reportez-vous à la section Clés de AWS CodePipeline condition du guide de l'utilisateur IAM. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, voir Actions définies par AWS CodePipeline.

Exemples

Pour consulter des exemples de politiques CodePipeline basées sur l'identité, consultez. AWS CodePipeline exemples de politiques basées sur l'identité

CodePipeline politiques basées sur les ressources

CodePipeline ne prend pas en charge les politiques basées sur les ressources. Cependant, un exemple de politique basée sur les ressources pour le service S3 associé CodePipeline est fourni.

Exemples

Pour consulter des exemples de politiques CodePipeline basées sur les ressources, voir AWS CodePipeline exemples de politiques basées sur les ressources

Autorisation basée sur les CodePipeline tags

Vous pouvez associer des balises aux CodePipeline ressources ou transmettre des balises dans une demande à CodePipeline. Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `codepipeline:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`. Pour plus d'informations sur le balisage des ressources CodePipeline , consultez Balisage des ressources.

Pour afficher un exemple de stratégie basée sur l'identité permettant de limiter l'accès à une ressource basée sur les balises de cette ressource, veuillez consulter Utilisation de balises pour contrôler l'accès aux CodePipeline ressources.

CodePipeline Rôles IAM

Un rôle IAM est une entité de votre AWS compte dotée d'autorisations spécifiques.

Utilisation d'informations d'identification temporaires avec CodePipeline

Vous pouvez utiliser des informations d'identification temporaires pour vous connecter à l'aide de la fédération, endosser un rôle IAM ou encore pour endosser un rôle intercompte. Vous obtenez des informations d'identification de sécurité temporaires en appelant des opérations d' AWS STS API telles que AssumeRoleou GetFederationToken.

CodePipeline prend en charge l'utilisation d'informations d'identification temporaires.

Rôles du service

CodePipeline permet à un service d'assumer un rôle de service en votre nom. Ce rôle autorise le service à accéder à des ressources d'autres services pour effectuer une action en votre nom. Les rôles de service s'affichent dans votre compte IAM et sont la propriété du compte. Cela signifie qu'un administrateur IAM peut modifier les autorisations associées à ce rôle. Toutefois, une telle action peut perturber le bon fonctionnement du service.

CodePipeline prend en charge les rôles de service.

AWS CodePipeline exemples de politiques basées sur l'identité

Par défaut, les utilisateurs et les rôles IAM ne sont pas autorisés à créer ou modifier les ressources CodePipeline . Ils ne peuvent pas non plus effectuer de tâches à l'aide de l' AWS API Console de gestion AWS AWS CLI, ou. Un administrateur IAM doit créer des politiques IAM autorisant les utilisateurs et les rôles à exécuter des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. Il doit ensuite attacher ces politiques aux utilisateurs ou aux groupes IAM ayant besoin de ces autorisations.

Pour savoir comment créer une stratégie IAM basée sur l'identité à l'aide de ces exemples de documents de stratégie JSON, veuillez consulter Création de stratégies dans l'onglet JSON dans le Guide de l'utilisateur IAM.

Pour savoir comment créer un pipeline utilisant les ressources d'un autre compte, et pour obtenir des exemples de politiques connexes, consultezCréez un pipeline CodePipeline qui utilise les ressources d'un autre AWS compte.

Rubriques

- Bonnes pratiques en matière de politiques
- Affichage des ressources dans la console
- Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations
- Identity-based exemples de politiques (IAM)
- Utilisation de balises pour contrôler l'accès aux CodePipeline ressources
- Autorisations requises pour utiliser la console
- Autorisations requises pour consulter les journaux de calcul dans la console
- AWS politiques gérées pour AWS CodePipeline

- Exemples de politiques gérées par le client

Bonnes pratiques en matière de politiques

Identity-based les politiques déterminent si quelqu'un peut créer, accéder ou supprimer CodePipeline des ressources dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.
- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politiques avec IAM Access Analyzer dans le Guide de l'utilisateur IAM.

- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Sécurisation de l'accès aux API avec MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Affichage des ressources dans la console

La console nécessite l'`ListRepositories`autorisation d'afficher la liste des référentiels pour votre AWS compte dans la AWS région où vous êtes connecté. La console comprend également une fonction Go to resource (Accéder aux ressources) qui permet d'effectuer rapidement une recherche de ressources sensible à la casse. Cette recherche est effectuée dans votre AWS compte dans la AWS région où vous êtes connecté. Les ressources suivantes sont affichées pour les services suivants :

- AWS CodeBuild : Projets de génération
- AWS CodeCommit : Référentiels
- AWS CodeDeploy : Applications
- AWS CodePipeline : Pipelines

Pour effectuer cette recherche pour les ressources dans tous les services, vous devez disposer des autorisations suivantes :

- AWS CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`
- CodePipeline: `ListPipelines`

Les résultats ne sont pas renvoyés pour les ressources d'un service si vous ne disposez pas d'autorisations pour ce service. Même si vous êtes autorisé à afficher des ressources, certaines ressources ne sont pas renvoyées si une valeur `Deny` explicite est définie pour l'affichage de ces ressources.

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or 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 exemples de politiques (IAM)

Vous pouvez attacher des politiques à des identités IAM. Par exemple, vous pouvez effectuer les opérations suivantes :

- Associer une politique d'autorisation à un utilisateur ou à un groupe de votre compte : pour autoriser un utilisateur à consulter les pipelines dans la console, vous pouvez associer une politique d'autorisations à un utilisateur ou à un groupe auquel l'utilisateur appartient.
- Attacher une politique d'autorisations à un rôle (accorder des autorisations entre comptes) : vous pouvez attacher une politique d'autorisation basée sur une identité à un rôle IAM afin d'accorder des autorisations entre comptes. Par exemple, l'administrateur du compte A peut créer un rôle pour accorder des autorisations entre comptes à un autre AWS compte (par exemple, le compte B) ou à un compte Service AWS comme suit :
  1. L'administrateur du Compte A crée un rôle IAM et attache une politique d'autorisation à ce rôle qui accorde des autorisations sur les ressources dans le Compte A.
  2. L'administrateur du Compte A attache une politique d'approbation au rôle identifiant le Compte B comme principal pouvant assumer ce rôle.
  3. L'administrateur du compte B peut ensuite déléguer les autorisations nécessaires pour assumer le rôle à n'importe quel utilisateur du compte B. Cela permet aux utilisateurs du compte B de créer ou d'accéder aux ressources du compte A. Le principal de la politique de confiance peut également être un Service AWS principal si vous souhaitez accorder l' Service AWS autorisation d'assumer le rôle.

  Pour en savoir plus sur l'utilisation d'IAM pour déléguer des autorisations, consultez Gestion des accès dans le Guide de l'utilisateur IAM.

Voici un exemple de politique d'autorisation qui accorde des autorisations pour désactiver et activer les transitions entre toutes les étapes du pipeline nommé `MyFirstPipeline` dans le `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/*"
      ]
    }
  ]
}
```

L'exemple suivant montre une politique du compte 111222333444 qui permet aux utilisateurs d'afficher, mais pas de modifier, le pipeline nommé `MyFirstPipeline` dans la console. Cette stratégie s'appuie sur la stratégie gérée `AWSCodePipeline_ReadOnlyAccess`, mais elle ne peut pas utiliser la stratégie gérée directement car elle est spécifique au pipeline `MyFirstPipeline`. Si vous ne souhaitez pas limiter la stratégie à un pipeline spécifique, il est conseillé d'utiliser l'une des stratégies gérées créées et stockées par . Pour plus d'informations, consultez Utilisation des politiques gérées. Vous devez associer cette politique à un rôle IAM que vous créez pour y accéder, par exemple un rôle nommé `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*"
        }
      }
    }
  ]
}
```

Après avoir créé cette politique, créez le rôle IAM dans le compte 111222333444 et associez la politique à ce rôle. Dans les relations de confiance du rôle, vous devez ajouter le AWS compte qui assumera ce rôle. L'exemple suivant montre une politique qui permet aux utilisateurs du *111111111111* AWS compte d'assumer les rôles définis dans le compte 111222333444 :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

L'exemple suivant montre une politique créée dans le *111111111111* AWS compte qui permet aux utilisateurs d'assumer le rôle nommé *CrossAccountPipelineViewers* dans le compte 111222333444 :

JSON

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

```
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource":
 "arn:aws:iam::111222333444:role/CrossAccountPipelineViewers"
        }
    ]
}
```

Vous pouvez créer des politiques IAM pour restreindre les appels et les ressources auxquels les utilisateurs de votre compte ont accès, puis associer ces politiques à votre utilisateur administratif. Pour plus d'informations sur la création de rôles IAM et pour découvrir des exemples de déclarations de politique IAM pour, voir. Exemples de politiques gérées par le client

Utilisation de balises pour contrôler l'accès aux CodePipeline ressources

Les conditions figurant dans les déclarations de politique IAM font partie de la syntaxe que vous utilisez pour spécifier les autorisations relatives aux ressources requises par CodePipeline les actions. L'utilisation des balises dans les conditions est un moyen de contrôler l'accès aux ressources et demandes. Pour plus d'informations sur le balisage CodePipeline des ressources, consultezBalisage des ressources. Cette rubrique explique le contrôle d'accès basé sur les balises.

Lorsque vous concevez des stratégies IAM, vous pouvez définir des autorisations granulaires en accordant l'accès à des ressources spécifiques. Au fur et à mesure que le nombre de ressources que vous gérez s'accroît, cette tâche devient plus difficile. Le balisage des ressources et l'utilisation de balises dans les déclarations de politique peuvent rendre cette tâche plus facile. Vous accordez l'accès en bloc à toute ressource avec une balise spécifique. Puis, vous appliquez cette balise à plusieurs reprises aux ressources correspondantes, lors de la création ou ultérieurement.

Les balises peuvent être attachées à la ressource ou transmises dans la demande aux services qui prennent en charge le balisage. Dans CodePipeline, les ressources peuvent avoir des balises, et certaines actions peuvent inclure des balises. Lorsque vous créez une politique IAM, vous pouvez utiliser des clés de condition de balise pour contrôler :

- quels utilisateurs peuvent effectuer des actions sur une ressource de pipeline, en fonction des balises que la ressource possède déjà ;
- quelles balises peuvent être transmises dans une demande d'action ;
- si des clés de balise spécifiques peuvent être utilisées dans une demande.

Les opérateurs de condition de chaîne permettent de créer des éléments `Condition` qui limitent l'accès après comparaison d'une clé à une valeur de chaîne. Vous pouvez ajouter `IfExists` à la fin de n'importe quel nom d'opérateur de condition, à l'exception de la condition Null. Ceci équivaut à spécifier que « Si la clé de politique est présente dans le contexte de la demande, la clé doit être traitée comme spécifié dans la politique. Si la clé n'est pas présente, la condition évalue l'élément de condition comme vrai. » Par exemple, vous pouvez utiliser `StringEqualsIfExists` pour restreindre par condition des clés qui peuvent ne pas être présentes sur d'autres types de ressources.

Pour connaître la syntaxe et la sémantique complètes des clés de condition des balises, consultez la section Contrôle de l'accès à l'aide de balises. Pour plus d'informations sur les clés de condition, consultez les ressources suivantes. Les exemples CodePipeline de politique présentés dans cette section s'alignent sur les informations suivantes sur les clés de condition et les développent avec des exemples de nuances, CodePipeline telles que l'imbrication des ressources.

- Opérateurs de condition de chaîne
- Services AWS qui fonctionnent avec IAM
- Syntaxe d'une stratégie de contrôle de service
- Éléments de politique JSON IAM : Condition
- lois : RequestTag/tag-key
- Clés de condition pour CodePipeline

Les exemples suivants montrent comment spécifier des conditions de balises dans les stratégies pour les utilisateurs CodePipeline .

Example 1 : Limiter les actions en fonction des balises dans la demande

La politique des utilisateurs `AWSCodePipeline_FullAccess` gérés donne aux utilisateurs des autorisations illimitées pour effectuer n'importe quelle CodePipeline action sur n'importe quelle ressource.

La politique suivante limite ce pouvoir et refuse aux utilisateurs non autorisés l'autorisation de créer des pipelines dans lesquels des balises spécifiques sont répertoriées dans la demande. Pour ce faire, elle refuse l'action `CreatePipeline` si la demande spécifie une balise nommée `Project` avec une valeur `ProjectA` ou `ProjectB`. (La clé de condition `aws:RequestTag` est utilisée pour contrôler les balises qui peuvent être transmises dans une demande IAM.)

Dans l'exemple suivant, le but de la politique est de refuser aux utilisateurs non autorisés l'autorisation de créer un pipeline avec les valeurs de balise spécifiées. Cependant, la création d'un pipeline nécessite d'accéder à des ressources en plus du pipeline lui-même (par exemple, les actions et les étapes du pipeline). Dans la mesure où la stratégie est `'Resource'` spécifiée`'*'`, la politique est évaluée par rapport à chaque ressource dotée d'un ARN et est créée lors de la création du pipeline. Ces ressources supplémentaires ne disposent pas de la clé de condition du tag, de sorte que la `StringEquals` vérification échoue et que l'utilisateur n'est pas autorisé à créer un pipeline. Pour éviter ce problème, utilisez l'opérateur de condition `StringEqualsIfExists` à la place. De cette façon, le test n'est effectué que si la clé de condition existe.

Vous pourriez lire ce qui suit : « Si la ressource vérifiée possède une clé de `"RequestTag/Project"` condition de balise, autorisez l'action uniquement si la valeur de la clé commence par`projectA`. Si la ressource en cours de vérification n'est pas dotée de cette clé de condition, peu importe. »

En outre, la politique empêche ces utilisateurs non autorisés d'altérer les ressources en utilisant la clé de `aws:TagKeys` condition pour empêcher les actions de modification des balises d'inclure ces mêmes valeurs de balise. L'administrateur d'un client doit associer cette politique IAM aux utilisateurs administratifs non autorisés, en plus de la politique des utilisateurs gérés.

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 : Limitez les actions de balisage en fonction des balises de ressources

La politique des utilisateurs `AWSCodePipeline_FullAccess` gérés donne aux utilisateurs des autorisations illimitées pour effectuer n'importe quelle CodePipeline action sur n'importe quelle ressource.

La stratégie suivante limite cette possibilité et interdit aux utilisateurs non autorisés d'effectuer des actions sur les pipelines de projets spécifiés. Pour ce faire, elle refuse certaines actions si la ressource possède une balise nommée `Project` avec une valeur `ProjectA` ou `ProjectB`. (La clé de condition `aws:ResourceTag` est utilisée pour contrôler l'accès aux ressources en fonction des balises sur ces ressources.) L'administrateur d'un client peut attacher cette stratégie IAM aux utilisateurs IAM non autorisés et à la stratégie d'utilisateur gérée.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": ["ProjectA", "ProjectB"]
        }
```

```
      }
    }
  ]
}
```

Example 3 : Limiter les actions en fonction des balises dans la demande

La stratégie suivante accorde aux utilisateurs l'autorisation de créer des pipelines de développement dans CodePipeline.

Pour ce faire, elle autorise les actions `CreatePipeline` et `TagResource` si la demande spécifie une balise nommée `Project` avec la valeur `ProjectA`. En d'autres termes, la seule clé de balise qui peut être spécifiée est`Project`, et sa valeur doit être`ProjectA`.

La clé de `aws:RequestTag` condition est utilisée pour contrôler les balises qui peuvent être transmises dans une demande IAM. La condition `aws:TagKeys` garantit que la clé de balise est sensible à la casse. Cette politique est utile pour les utilisateurs ou les rôles auxquels la politique d'utilisateur `AWSCodePipeline_FullAccess` géré n'est pas attachée. La politique gérée donne aux utilisateurs des autorisations illimitées pour effectuer n'importe quelle CodePipeline action sur n'importe quelle ressource.

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 : Limitez les actions de débalisage en fonction des balises de ressources

La politique des utilisateurs `AWSCodePipeline_FullAccess` gérés donne aux utilisateurs des autorisations illimitées pour effectuer n'importe quelle CodePipeline action sur n'importe quelle ressource.

La stratégie suivante limite cette possibilité et interdit aux utilisateurs non autorisés d'effectuer des actions sur les pipelines de projets spécifiés. Pour ce faire, elle refuse certaines actions si la ressource possède une balise nommée `Project` avec une valeur `ProjectA` ou `ProjectB`.

La politique empêche également ces utilisateurs non autorisés d'altérer les ressources en utilisant la clé de `aws:TagKeys` condition pour empêcher les actions de modification des balises de supprimer complètement la `Project` balise. L'administrateur d'un client doit associer cette politique IAM à des utilisateurs ou à des rôles non autorisés, en plus de la politique d'utilisateur géré.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

Autorisations requises pour utiliser la console

Pour utiliser dans la console , vous devez disposer d'un ensemble minimal d'autorisations provenant des services suivants :

- Gestion des identités et des accès AWS
- Amazon Simple Storage Service

Ces autorisations vous permettent de décrire d'autres AWS ressources pour votre AWS compte.

Selon les autres services que vous intégrez dans vos pipelines, vous devrez peut-être demander des autorisations à un ou plusieurs des services suivants :

- AWS CodeCommit
- AWS CodeBuild
- CloudFormation
- AWS CodeDeploy
- AWS Elastic Beanstalk
- AWS Lambda
- AWS OpsWorks

Si vous créez une politique IAM plus restrictive que les autorisations minimales requises, la console ne fonctionnera pas comme prévu pour les utilisateurs dotés de cette politique IAM. Pour garantir que ces utilisateurs puissent continuer à utiliser la console, attachez également la stratégie gérée `AWSCodePipeline_ReadOnlyAccess` à l'utilisateur, comme décrit dans AWS politiques gérées pour AWS CodePipeline.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent l'API AWS CLI ou l'API.

Autorisations requises pour consulter les journaux de calcul dans la console

Pour consulter les journaux dans le cadre de l'action Commandes sur la CodePipeline console, le rôle de console doit disposer d'autorisations. Pour afficher les journaux dans la console, ajoutez les `logs:GetLogEvents` autorisations au rôle de console.

Dans la déclaration de politique relative au rôle de console, limitez les autorisations au niveau du pipeline, comme indiqué dans l'exemple suivant.

```
{
    "Effect": "Allow",
    "Action": [
        "Action": "logs:GetLogEvents"
    ],
    "Resource": "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
}
```

AWS politiques gérées pour AWS CodePipeline

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

> **Important**
>
> Les politiques `AWSCodePipelineFullAccess` gérées par AWS `AWSCodePipelineReadOnlyAccess` ont été remplacées. Utilisez les `AWSCodePipeline_ReadOnlyAccess` politiques `AWSCodePipeline_FullAccess` et.

AWS politique gérée : **AWSCodePipeline_ FullAccess**

Il s'agit d'une politique qui accorde un accès complet à CodePipeline. Pour consulter le document de politique JSON dans la console IAM, consultez AWSCodePipeline_ FullAccess.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

- `codepipeline`— Accorde des autorisations à CodePipeline.
- `chatbot`— Accorde des autorisations permettant aux principaux de gérer les ressources dans Amazon Q Developer dans des applications de chat.
- `cloudformation`— Accorde des autorisations permettant aux principaux de gérer les piles de ressources dans. CloudFormation
- `cloudtrail`— Accorde des autorisations permettant aux principaux de gérer les ressources de connexion. CloudTrail
- `codebuild`— Accorde des autorisations pour permettre aux principaux d'accéder aux ressources de construction. CodeBuild
- `codecommit`— Accorde des autorisations permettant aux principaux d'accéder aux ressources sources dans CodeCommit.
- `codedeploy`— Accorde des autorisations permettant aux principaux d'accéder aux ressources de déploiement dans CodeDeploy.
- `codestar-notifications`— Accorde des autorisations permettant aux principaux d'accéder aux ressources dans AWS CodeStar les notifications.
- `ec2`— Accorde des autorisations pour autoriser les déploiements CodeCatalyst afin de gérer l'équilibrage de charge élastique dans Amazon EC2.
- `ecr`— Accorde des autorisations pour autoriser l'accès aux ressources dans Amazon ECR.
- `elasticbeanstalk`— Accorde des autorisations permettant aux principaux d'accéder aux ressources dans Elastic Beanstalk.
- `iam`— Accorde des autorisations permettant aux principaux de gérer les rôles et les politiques dans IAM.

- `lambda`— Accorde des autorisations permettant aux principaux de gérer les ressources dans Lambda.
- `events`— Accorde des autorisations permettant aux principaux de gérer les ressources dans CloudWatch Events.
- `opsworks`— Accorde des autorisations permettant aux principaux de gérer les ressources dans AWS OpsWorks.
- `s3`— Accorde des autorisations permettant aux principaux de gérer les ressources dans Amazon S3.
- `sns`— Accorde des autorisations permettant aux principaux de gérer les ressources de notification dans Amazon SNS.
- `states`— Accorde des autorisations permettant aux principaux de visualiser les machines d'état dans AWS Step Functions lesquelles ils se trouvent. Une machine à états consiste en un ensemble d'états qui gèrent les tâches et la transition entre les états.

Pour la politique, voir AWSCodePipeline_ FullAccess.

AWS politique gérée : **`AWSCodePipeline_ ReadOnlyAccess`**

Il s'agit d'une politique qui accorde un accès en lecture seule à. CodePipeline Pour consulter le document de politique JSON dans la console IAM, consultez AWSCodePipeline_ ReadOnlyAccess.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

- `codepipeline`— Accorde des autorisations aux actions dans CodePipeline.
- `codestar-notifications`— Accorde des autorisations permettant aux principaux d'accéder aux ressources dans AWS CodeStar les notifications.
- `s3`— Accorde des autorisations permettant aux principaux de gérer les ressources dans Amazon S3.
- `sns`— Accorde des autorisations permettant aux principaux de gérer les ressources de notification dans Amazon SNS.

Pour la politique, voir AWSCodePipeline_ ReadOnlyAccess.

AWS politique gérée : **`AWSCodePipelineApproverAccess`**

Il s'agit d'une politique qui autorise l'approbation ou le rejet d'une action d'approbation manuelle. Pour consulter le document de politique JSON dans la console IAM, consultez AWSCodePipelineApproverAccess.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

- `codepipeline`— Accorde des autorisations aux actions dans CodePipeline.

Pour la politique, voir AWSCodePipelineApproverAccess.

AWS politique gérée : **`AWSCodePipelineCustomActionAccess`**

Il s'agit d'une politique qui autorise la création d'actions personnalisées CodePipeline ou l'intégration des ressources Jenkins pour les actions de création ou de test. Pour consulter le document de politique JSON dans la console IAM, consultez AWSCodePipelineCustomActionAccess.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

- `codepipeline`— Accorde des autorisations aux actions dans CodePipeline.

Pour la politique, voir AWSCodePipelineCustomActionAccess.

CodePipeline politiques et notifications gérées

CodePipeline prend en charge les notifications, qui peuvent informer les utilisateurs des modifications importantes apportées aux pipelines. Les politiques gérées CodePipeline incluent des déclarations

de politique relatives à la fonctionnalité de notification. Pour plus d'informations, consultez En quoi consistent les notifications ?

Autorisations liées aux notifications dans les stratégies gérées d'accès complet

Cette politique gérée accorde des autorisations CodePipeline ainsi que les services CodeCommit, CodeBuild CodeDeploy, et AWS CodeStar notifications associés. La politique accorde également les autorisations dont vous avez besoin pour travailler avec d'autres services intégrés à vos pipelines, tels qu'Amazon S3, Elastic CloudTrail Beanstalk, Amazon EC2 et. CloudFormation Les utilisateurs auxquels cette politique gérée est appliquée peuvent également créer et gérer des sujets Amazon SNS pour les notifications, abonner et désinscrire des utilisateurs à des sujets, répertorier les sujets à choisir comme cibles pour les règles de notification et répertorier Amazon Q Developer dans les applications de chat clientes configurées pour Slack.

La stratégie gérée `AWSCodePipeline_FullAccess` inclut les déclarations suivantes pour permettre un accès complet aux notifications.

```
    {
        "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": "*"
    }
```

Autorisations liées aux notifications dans les stratégies gérées en lecture seule

La stratégie gérée `AWSCodePipeline_ReadOnlyAccess` inclut les déclarations suivantes pour autoriser l'accès en lecture seule aux notifications. Les utilisateurs auxquels s'applique cette stratégie peuvent voir des notifications pour les ressources, mais ne peuvent pas les créer, les gérer ni s'y abonner.

```
   {
        "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": "*"
    }
```

Pour plus d'informations sur l'IAM et les notifications, consultez Identity and Access Management for AWS CodeStar Notifications.

AWS CodePipeline mises à jour de AWS stratégies gérées

Consultez les détails des mises à jour des politiques AWS gérées CodePipeline depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la CodePipeline Page historique du document https://docs.aws.amazon.com/codepipeline/latest/userguide/history.html.

Modifier	Description	Date
AWSCodePipeline_ FullAccess — Mises à jour de la politique existante	CodePipeline a ajouté une autorisation à cette politique pour la prise `ListStacks` en charge dans CloudFormation.	15 mars 2024
AWSCodePipeline_ FullAccess — Mises à jour de la politique existante	Cette politique a été mise à jour afin d'ajouter des autorisations pour Amazon Q Developer dans les applicati	21 juin 2023

Modifier	Description	Date
	ons de chat. Pour plus d'informations, consultez CodePipeline politiques et notifications gérées.	
AWSCodePipeline_ FullAccess et AWSCodePipeline_ politiques ReadOnlyAccess gérées — Mises à jour de la politique existante	CodePipeline a ajouté une autorisation à ces politiques pour prendre en charge un type de notification supplémentaire utilisant Amazon Q Developer dans les applications de chat,`chatbot:ListMicrosoftTeamsChannelConfigurations` .	16 mai 2023
AWSCodePipelineFullAccess : obsolète	Cette stratégie a été remplacée par `AWSCodePipeline_FullAccess` . Après le 17 novembre 2022, cette politique ne pourra plus être associée à de nouveaux utilisateurs, groupes ou rôles. Pour plus d'informations, consultez AWS politiques gérées pour AWS CodePipeline.	17 novembre 2022

Modifier	Description	Date
AWSCodePipelineReadOnlyAccess : obsolète	Cette stratégie a été remplacée par `AWSCodePipeline_ReadOnlyAccess` . Après le 17 novembre 2022, cette politique ne pourra plus être associée à de nouveaux utilisateurs, groupes ou rôles. Pour plus d'informations, consultez AWS politiques gérées pour AWS CodePipeline.	17 novembre 2022
CodePipeline a commencé à suivre les modifications	CodePipeline a commencé à suivre les modifications apportées AWS à ses politiques gérées.	12 mars 2021

Exemples de politiques gérées par le client

Dans cette section, vous trouverez des exemples de politiques utilisateur qui accordent des autorisations pour diverses actions . Ces politiques fonctionnent lorsque vous utilisez l'API, AWS les SDK ou le AWS CLI. Lorsque vous utilisez la console, vous devez accorder des autorisations supplémentaires spécifiques à la console. Pour plus d'informations, consultez Autorisations requises pour utiliser la console.

> **Note**
>
> Tous les exemples utilisent la région USA Ouest (Oregon) (`us-west-2`) et contiennent des ID de compte fictifs.

Exemples

- Exemple 1 : Octroi d'autorisations pour obtenir l'état d'un pipeline

- Exemple 2 : Octroi d'autorisations pour activer et désactiver les transitions entre les étapes
- Exemple 3 : Octroi d'autorisations pour obtenir la liste de tous les types d'action disponibles
- Exemple 4 : Octroi d'autorisations pour approuver ou rejeter des actions d'approbation manuelle
- Exemple 5 : Octroi d'autorisations pour interroger les tâches d'une action personnalisée
- Exemple 6 : joindre ou modifier une politique pour l'intégration de Jenkins avec AWS CodePipeline
- Exemple 7 : Configuration d'un accès entre comptes à un pipeline

Exemple 1 : Octroi d'autorisations pour obtenir l'état d'un pipeline

L'exemple suivant accorde des autorisations pour obtenir l'état du pipeline nommé `MyFirstPipeline` :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:GetPipelineState"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline"
        }
    ]
}
```

Exemple 2 : Octroi d'autorisations pour activer et désactiver les transitions entre les étapes

L'exemple suivant accorde des autorisations pour activer et désactiver les transitions entre toutes les étapes du pipeline nommé `MyFirstPipeline` :

JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:DisableStageTransition",
                "codepipeline:EnableStageTransition"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline/*"
        }
    ]
}
```

Pour permettre à l'utilisateur d'activer et de désactiver les transitions pour une seule étape d'un pipeline, vous devez spécifier l'étape. Par exemple, pour permettre à l'utilisateur d'activer et de désactiver les transitions pour une étape nommée `Staging` d'un pipeline nommé `MyFirstPipeline` :

```
"Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/Staging"
```

Exemple 3 : Octroi d'autorisations pour obtenir la liste de tous les types d'action disponibles

L'exemple suivant accorde les autorisations nécessaires pour obtenir une liste de tous les types d'action disponibles pour les pipelines dans la région `us-west-2` :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:ListActionTypes"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:actiontype:*"
        }
    ]
}
```

Exemple 4 : Octroi d'autorisations pour approuver ou rejeter des actions d'approbation manuelle

L'exemple suivant accorde les autorisations nécessaires pour approuver ou rejeter des actions d'approbation manuelle lors d'une étape nommée `Staging` dans un pipeline nommé `MyFirstPipeline` :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PutApprovalResult"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline/Staging/*"
        }
    ]
}
```

Exemple 5 : Octroi d'autorisations pour interroger les tâches d'une action personnalisée

L'exemple suivant accorde les autorisations nécessaires pour rechercher, dans tous les pipelines, les tâches de l'action personnalisée nommée `TestProvider`, qui est une action de type `Test` dans sa première version :

> **Note**
>
> Le job worker chargé d'une action personnalisée peut être configuré sous un autre AWS compte ou nécessiter un rôle IAM spécifique pour fonctionner.

JSON

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

```
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PollForJobs"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111222333444:actionType:Custom/Test/TestProvider/1"
            ]
        }
    ]
}
```

Exemple 6 : joindre ou modifier une politique pour l'intégration de Jenkins avec AWS CodePipeline

Si vous configurez un pipeline pour utiliser Jenkins à des fins de génération ou de test, créez une identité distincte pour cette intégration et associez une politique IAM dotée des autorisations minimales requises pour l'intégration entre Jenkins et. Cette stratégie est similaire à la stratégie gérée `AWSCodePipelineCustomActionAccess`. L'exemple suivant montre une politique d'intégration de Jenkins :

JSON

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

    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:AcknowledgeJob",
                "codepipeline:GetJobDetails",
                "codepipeline:PollForJobs",
                "codepipeline:PutJobFailureResult",
                "codepipeline:PutJobSuccessResult"
            ],
            "Resource": "*"
        }
    ]
}
```

Exemple 7 : Configuration d'un accès entre comptes à un pipeline

Vous pouvez configurer l'accès aux pipelines pour les utilisateurs et groupes d'un autre compte AWS . La méthode recommandée consiste à créer un rôle dans le compte où le pipeline a été créé. Le rôle doit permettre aux utilisateurs de l'autre AWS compte d'assumer ce rôle et d'accéder au pipeline. Pour plus d'informations, voir Procédure pas à pas : Cross-Account accès à l'aide de rôles.

L'exemple suivant montre une politique du compte 80398EXAMPLE qui permet aux utilisateurs de consulter, mais pas de modifier, le pipeline nommé `MyFirstPipeline` dans la console. Cette stratégie s'appuie sur la stratégie gérée `AWSCodePipeline_ReadOnlyAccess`, mais elle ne peut pas utiliser la stratégie gérée directement car elle est spécifique au pipeline `MyFirstPipeline`. Si vous ne souhaitez pas limiter la stratégie à un pipeline spécifique, il est conseillé d'utiliser l'une des stratégies gérées créées et stockées par . Pour plus d'informations, consultez Utilisation des politiques gérées. Vous devez associer cette politique à un rôle IAM que vous créez pour y accéder, par exemple un rôle nommé `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"
        }
    ]
}
```

Après avoir créé cette politique, créez le rôle IAM dans le compte 80398EXAMPLE et associez la politique à ce rôle. Dans les relations de confiance du rôle, vous devez ajouter le AWS compte qui assume ce rôle.

L'exemple suivant montre une politique créée dans le *111111111111* AWS compte qui permet aux utilisateurs d'assumer le rôle nommé `CrossAccountPipelineViewers` dans le compte 80398EXAMPLE :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource":
 "arn:aws:iam::111122223333:role/CrossAccountPipelineViewers"
        }
    ]
}
```

AWS CodePipeline exemples de politiques basées sur les ressources

D'autres services, tels qu'Amazon S3, prennent également en charge les politiques d'autorisation basées sur une ressource. Par exemple, vous pouvez attacher une politique à un compartiment S3 pour gérer les autorisations d'accès à ce compartiment. Bien qu'il CodePipeline ne prenne pas en charge les politiques basées sur les ressources, il stocke les artefacts à utiliser dans des pipelines dans des compartiments S3 versionnés.

Example Pour créer une politique pour un compartiment S3 à utiliser comme magasin d'artefacts pour CodePipeline

Vous pouvez utiliser n'importe quel compartiment S3 versionné comme magasin d'artefacts pour. CodePipeline Si vous utilisez l'assistant Création d'un pipeline pour créer votre premier pipeline, ce compartiment S3 est créé automatiquement pour garantir que tous les objets chargés dans le magasin d'artefacts sont chiffrés et que les connexions au compartiment sont sécurisées. Si vous créez votre propre compartiment S3, pensez à ajouter la stratégie suivante ou ses éléments au compartiment, à titre de bonne pratique. Dans cette stratégie, l'ARN du compartiment S3 est `codepipeline-us-east-2-1234567890`. Remplacez cet ARN par l'ARN de votre compartiment 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
                }
            }
```

```
        }
    ]
}
```

Résolution des problèmes AWS CodePipeline identité et accès

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec CodePipeline IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans CodePipeline
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je suis administrateur et je souhaite autoriser d'autres personnes à accéder CodePipeline
- Je souhaite autoriser des personnes extérieures à mon AWS compte pour accéder à mes CodePipeline ressources

Je ne suis pas autorisé à effectuer une action dans CodePipeline

S'il vous Console de gestion AWS indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni votre nom d'utilisateur et votre mot de passe.

L'exemple d'erreur suivant se produit lorsque l'utilisateur `mateojackson` IAM essaie d'utiliser la console pour afficher les détails d'un pipeline, mais ne dispose pas des `codepipeline:GetPipeline` autorisations nécessaires.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 codepipeline:GetPipeline on resource: my-pipeline
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource `my-pipeline` à l'aide de l'action `codepipeline:GetPipeline`.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à exécuter l'action `iam:PassRole`, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni votre nom d'utilisateur et votre mot de passe.

Demandez à cette personne de mettre à jour vos politiques pour vous permettre de transmettre un rôle à CodePipeline.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service, au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, vous devez disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour exécuter une action dans CodePipeline. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction du service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, Mary demande à son administrateur de mettre à jour ses politiques pour lui permettre d'exécuter l'action `iam:PassRole`.

Je suis administrateur et je souhaite autoriser d'autres personnes à accéder CodePipeline

Pour autoriser d'autres personnes à y accéder CodePipeline, vous devez accorder l'autorisation aux personnes ou aux applications qui ont besoin d'y accéder. Si vous utilisez AWS IAM Identity Center pour gérer des personnes et des applications, vous attribuez des ensembles d'autorisations aux utilisateurs ou aux groupes afin de définir leur niveau d'accès. Les ensembles d'autorisations créent et attribuent automatiquement des politiques IAM aux rôles IAM associés à la personne ou à l'application. Pour plus d'informations, consultez la section Ensembles d'autorisations dans le guide de AWS IAM Identity Center l'utilisateur.

Si vous n'utilisez pas IAM Identity Center, vous devez créer des entités IAM (utilisateurs ou rôles) pour les personnes ou les applications qui ont besoin d'un accès. Vous devez ensuite associer une politique à l'entité qui leur accorde les autorisations appropriées dans CodePipeline. Une fois les autorisations accordées, fournissez les informations d'identification à l'utilisateur ou au développeur de l'application. Ils utiliseront ces informations d'identification pour y accéder AWS. Pour en savoir plus sur la création d'utilisateurs, de groupes, de politiques et d'autorisations IAM, consultez la section Identités, politiques et autorisations IAM dans le guide de l'utilisateur IAM.

Je souhaite autoriser des personnes extérieures à mon AWS compte pour accéder à mes CodePipeline ressources

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si ces fonctionnalités sont prises CodePipeline en charge, consultezComment ? AWS CodePipeline fonctionne avec IAM.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

référence des autorisations

Utilisez le tableau suivant comme référence lorsque vous configurez le contrôle d'accès et que vous rédigez des politiques d'autorisation que vous pouvez associer à une identité IAM (politiques basées sur l'identité). Ce tableau répertorie chaque opération d'API et les actions correspondantes pour lesquelles vous pouvez accorder des autorisations. Pour les opérations qui prennent en charge les autorisations au niveau des ressources, le tableau répertorie les AWS ressources pour lesquelles vous pouvez accorder les autorisations. Vous spécifiez les actions dans le champ `Action` de la politique.

Resource-level les autorisations sont celles qui vous permettent de spécifier les ressources sur lesquelles les utilisateurs sont autorisés à effectuer des actions. AWS CodePipeline fournit une prise en charge partielle des autorisations au niveau des ressources. Cela signifie que pour certains appels d' AWS CodePipeline API, vous pouvez contrôler le moment où les utilisateurs sont autorisés à utiliser ces actions en fonction des conditions qui doivent être remplies ou des ressources que les utilisateurs sont autorisés à utiliser. Par exemple, vous pouvez accorder aux utilisateurs l'autorisation d'afficher des informations d'exécution de pipeline, mais uniquement pour un ou plusieurs pipelines spécifiques.

> **Note**
>
> La colonne Ressources répertorie la ressource requise pour les appels d'API qui prennent en charge les autorisations de niveau ressource. Pour les appels d'API qui ne prennent pas en charge les autorisations au niveau des ressources, vous pouvez accorder aux utilisateurs l'autorisation de l'utiliser, mais vous devez spécifier un caractère générique (*) pour l'élément ressource de votre déclaration de stratégie.

Opérations d'API et autorisations requises pour les actions

AcknowledgeJob

Action : `codepipeline:AcknowledgeJob`

Requise pour afficher les informations relatives à une tâche spécifiée et déterminer si cette tâche a été reçue par l'exécutant de tâches. Utilisée pour les actions personnalisées uniquement.

Ressources : Prend en charge uniquement un caractère générique (*) dans l'élément `Resource` de la stratégie.

AcknowledgeThirdPartyJob

Action : `codepipeline:AcknowledgeThirdPartyJob`

Requise pour vérifier qu'un exécutant de tâches a reçu la tâche spécifiée. Utilisée pour les actions partenaires uniquement.

Ressources : Prend en charge uniquement un caractère générique (*) dans l'élément `Resource` de la stratégie.

CreateCustomActionType

Action : `codepipeline:CreateCustomActionType`

Nécessaire pour créer une nouvelle action personnalisée pouvant être utilisée dans tous les pipelines associés au AWS compte. Utilisée pour les actions personnalisées uniquement.

Ressources:

Type d'action

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:actiontype:`*`owner`*`/`*`category`*`/`*`provider`*`/`*`versio`*

CreatePipeline

Action : `codepipeline:CreatePipeline`

Requise pour créer un pipeline.

Ressources:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

DeleteCustomActionType

Action : `codepipeline:DeleteCustomActionType`

Requise pour marquer une action personnalisée comme supprimée. `PollForJobs` pour l'action personnalisée échoue une fois que l'action est marquée pour suppression. Utilisée pour les actions personnalisées uniquement.

Ressources:

Type d'action

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:actiontype:`*`owner`*`/`*`category`*`/`*`provider`*`/`*`versio`*

DeletePipeline

Action : `codepipeline:DeletePipeline`

Requise pour supprimer un pipeline.

Ressources:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

DeleteWebhook

Action : `codepipeline:DeleteWebhook`

Requise pour supprimer un webhook.

Ressources:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

DeregisterWebhookWithThirdParty

Action : `codepipeline:DeregisterWebhookWithThirdParty`

Avant de supprimer un webhook, il est nécessaire de supprimer la connexion entre le webhook créé par CodePipeline et l'outil externe dont les événements doivent être détectés. Actuellement pris en charge uniquement pour les webhooks qui ciblent un type d'action de GitHub.

Ressources:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

DisableStageTransition

Action : `codepipeline:DisableStageTransition`

Requise pour empêcher les artefacts d'un pipeline de passer à l'étape suivante du pipeline.

Ressources:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

EnableStageTransition

Action : `codepipeline:EnableStageTransition`

Requise pour permettre aux artefacts d'un pipeline de passer à une étape d'un pipeline.

Ressources:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

GetJobDetails

Action : `codepipeline:GetJobDetails`

Requise pour récupérer des informations sur une tâche. Utilisée uniquement pour les actions personnalisées.

Ressources : Aucune ressource n'est requise.

GetPipeline

Action : `codepipeline:GetPipeline`

Requis pour récupérer la structure, les étapes, les actions et les métadonnées d'un pipeline, notamment l'ARN du pipeline.

Ressources:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

GetPipelineExecution

Action : `codepipeline:GetPipelineExecution`

Requise pour récupérer des informations sur une exécution d'un pipeline, ainsi que des informations sur les artefacts, l'ID d'exécution du pipeline, et le nom, la version et l'état du pipeline.

Ressources:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

GetPipelineState

Action : `codepipeline:GetPipelineState`

Requise pour récupérer des informations sur l'état d'un pipeline, notamment les étapes et les actions.

Ressources:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

GetThirdPartyJobDetails

Action : `codepipeline:GetThirdPartyJobDetails`

Requise pour demander les détails d'une tâche pour une action tierce. Utilisée pour les actions partenaires uniquement.

Ressources : Prend en charge uniquement un caractère générique (*) dans l'élément `Resource` de la stratégie.

ListActionTypes

Action : `codepipeline:ListActionTypes`

Nécessaire pour générer un résumé de tous les types CodePipeline d'actions associés à votre compte.

Ressources:

Type d'action

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versior`

ListPipelineExecutions

Action : `codepipeline:ListPipelineExecutions`

Requise pour générer un résumé de exécutions les plus récentes pour un pipeline.

Ressources:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

ListPipelines

Action : `codepipeline:ListPipelines`

Requise pour générer un résumé de tous les pipelines associés à votre compte.

Ressources:

ARN du pipeline avec caractère générique (les autorisations au niveau des ressources au niveau du nom du pipeline ne sont pas prises en charge)

`arn:aws:codepipeline:region:account:*`

ListTagsForResource

Action : `codepipeline:ListTagsForResource`

Requise pour répertorier les balises pour une ressource spécifiée.

Ressources:

Type d'action

`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

Action : `codepipeline:ListWebhooks`

Requise pour répertorier tous les webhooks du compte pour cette région.

Ressources:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

PollForJobs

Action(s) : `codepipeline:PollForJobs`

Nécessaire pour récupérer des informations sur les tâches CodePipeline à exécuter.

Ressources:

Type d'action

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

PollForThirdPartyJobs

Action : `codepipeline:PollForThirdPartyJobs`

Requise pour déterminer la présence de tâches tierces sur lesquelles un exécutant de tâches doit agir. Utilisée pour les actions partenaires uniquement.

Ressources : Prend en charge uniquement un caractère générique (*) dans l'élément `Resource` de la stratégie.

PutActionRevision

Action : `codepipeline:PutActionRevision`

Obligatoire pour communiquer des informations CodePipeline sur les nouvelles révisions apportées à une source.

Ressources:

Action

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

PutApprovalResult

Action : `codepipeline:PutApprovalResult`

Requise pour signaler à CodePipeline la réponse à une demande d'approbation manuelle. Les réponses valides sont `Approved` et `Rejected`.

Ressources:

Action

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

> **Note**
>
> Cet appel d'API prend en charge les autorisations au niveau des ressources. Cependant, vous pouvez rencontrer une erreur si vous utilisez la console IAM ou le Générateur de

stratégies pour créer des stratégies avec `"codepipeline:PutApprovalResult"` qui spécifient un ARN de ressource. Si vous rencontrez une erreur, vous pouvez utiliser l'onglet JSON dans la console IAM ou l'interface de ligne de commande pour créer une stratégie.

PutJobFailureResult

Action : `codepipeline:PutJobFailureResult`

Requise pour signaler une tâche qui a échoué et qui est renvoyée dans le pipeline par un exécutant de tâches. Utilisée pour les actions personnalisées uniquement.

Ressources : Prend en charge uniquement un caractère générique (*) dans l'élément `Resource` de la stratégie.

PutJobSuccessResult

Action : `codepipeline:PutJobSuccessResult`

Requise pour signaler une tâche qui a réussi et qui est renvoyée dans le pipeline par un exécutant de tâches. Utilisée pour les actions personnalisées uniquement.

Ressources : Prend en charge uniquement un caractère générique (*) dans l'élément `Resource` de la stratégie.

PutThirdPartyJobFailureResult

Action : `codepipeline:PutThirdPartyJobFailureResult`

Requise pour signaler l'échec d'une tâche tierce lors de son renvoi au pipeline par un exécutant de tâches. Utilisée pour les actions partenaires uniquement.

Ressources : Prend en charge uniquement un caractère générique (*) dans l'élément `Resource` de la stratégie.

PutThirdPartyJobSuccessResult

Action : `codepipeline:PutThirdPartyJobSuccessResult`

Requise pour signaler la réussite d'une tâche tierce lors de son renvoi au pipeline par un exécutant de tâches. Utilisée pour les actions partenaires uniquement.

Ressources : Prend en charge uniquement un caractère générique (*) dans l'élément `Resource` de la stratégie.

PutWebhook

Action : `codepipeline:PutWebhook`

Requise pour créer un webhook.

Ressources:

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RegisterWebhookWithThirdParty

Action : `codepipeline:RegisterWebhookWithThirdParty`

Ressources:

Après la création d'un webhook, cette API est requise pour configurer les tiers pris en charge pour appeler l'URL de webhook générée.

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RetryStageExecution

Action : `codepipeline:RetryStageExecution`

Requise pour reprendre l'exécution d'un pipeline en réessayant les dernières actions d'une étape qui ont échoué.

Ressources:

Pipeline

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:`*`pipeline-name`*

StartPipelineExecution

Action : `codepipeline:StartPipelineExecution`

Requise pour démarrer le pipeline spécifié (en particulier, pour démarrer le traitement de la dernière validation à l'emplacement source spécifié dans le cadre du pipeline).

Ressources:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

TagResource

Action : `codepipeline:TagResource`

Requise pour baliser la ressource spécifiée.

Ressources:

Type d'action

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UntagResource

Action : `codepipeline:UntagResource`

Requise pour supprimer les balises de la ressource spécifiée.

Ressources:

Type d'action

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UpdatePipeline

Action : `codepipeline:UpdatePipeline`

Requise pour mettre à jour un pipeline spécifié en lui apportant des modifications ou des changements de structure.

Ressources:

Pipeline

`arn:aws:codepipeline:region:account:pipeline-name`

Gérer le rôle CodePipeline de service

Le rôle de service est configuré avec une ou plusieurs politiques qui contrôlent l'accès aux AWS ressources utilisées par le pipeline. Vous souhaiterez peut-être associer d'autres politiques à ce rôle, modifier la politique attachée au rôle ou configurer des politiques pour d'autres rôles de service dans AWS. Vous pouvez également attacher une stratégie à un rôle lorsque vous configurez l'accès entre comptes à votre pipeline.

Important

Le fait de modifier une déclaration de stratégie ou d'associer une autre stratégie au rôle peut nuire au fonctionnement de vos pipelines. Assurez-vous de bien tenir compte de tout ce qu'implique une modification du rôle de service pour . Veillez à tester vos pipelines après avoir modifié le rôle de service.

Note

Dans la console, les rôles de service créés avant septembre 2018 sont créés avec le nom `oneClick_AWS-CodePipeline-Service_ID-Number`.
Les rôles de service créés après septembre 2018 utilisent le format de nom de rôle de service `AWSCodePipelineServiceRole-Region-Pipeline_Name`. Par exemple, pour un pipeline nommé `MyFirstPipeline` dans`eu-west-2`, la console nomme le rôle et la politique`AWSCodePipelineServiceRole-eu-west-2-MyFirstPipeline`.

CodePipeline politique relative aux rôles de service

La déclaration de politique relative aux rôles de CodePipeline service contient les autorisations minimales pour la gestion des pipelines. Vous pouvez modifier l'énoncé du rôle de service pour supprimer ou ajouter l'accès aux ressources que vous n'utilisez pas. Consultez la référence d'action appropriée pour connaître les autorisations minimales requises CodePipeline utilisées pour chaque action.

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**
>
> Dans la politique, les autorisations suivantes sont requises lorsque les objets S3 de votre compartiment source contiennent des balises :
>
> ```
> s3:PutObjectTagging
> s3:GetObjectTagging
> s3:GetObjectVersionTagging
> ```

Supprimer les autorisations associées au rôle CodePipeline de service

Vous pouvez modifier la déclaration du rôle de service pour supprimer l'accès aux ressources que vous n'utilisez pas. Par exemple, si aucun de vos pipelines n'inclut Elastic Beanstalk, vous pouvez modifier la déclaration de politique afin de supprimer la section qui autorise l'accès aux ressources Elastic Beanstalk.

De même, si aucun de vos pipelines ne l'inclut CodeDeploy, vous pouvez modifier la déclaration de politique pour supprimer la section qui accorde l'accès aux CodeDeploy ressources :

```
{
"Action": [
    "codedeploy:CreateDeployment",
    "codedeploy:GetApplicationRevision",
    "codedeploy:GetDeployment",
    "codedeploy:GetDeploymentConfig",
    "codedeploy:RegisterApplicationRevision"
],
```

```
    "Resource": "*",
    "Effect": "Allow"
},
```

Ajouter des autorisations au rôle CodePipeline de service

Vous devez mettre à jour votre déclaration de politique de rôle de service avec des autorisations pour une déclaration de politique de rôle de service qui n'est Service AWS pas déjà incluse dans la déclaration de politique de rôle de service par défaut avant de pouvoir l'utiliser dans vos pipelines.

Cela est particulièrement important si le rôle de service que vous utilisez pour vos pipelines a été créé avant que le support ne soit ajouté à un Service AWS.

Le tableau suivant indique à quel moment le support a été ajouté pour les autres Services AWS.

Service AWS	CodePipeline date de support
CodePipeline le support d'action invoque a été ajouté. Consultez Autorisations relatives à la politique de rôle de service pour l'action d' CodePipeline invocation.	14 mars 2025
`EC2`support d'action ajouté. Consultez Autorisations relatives à la politique de rôle de service pour l'action de déploiement d'EC2.	21 février 2025
`EKS`support d'action ajouté. Consultez Autorisations relatives aux politiques de rôle de service.	20 février 2025
La prise en charge des `ECRBuildAndPublish` actions Amazon Elastic Container Registry a été ajoutée. Consultez Autorisations relatives aux rôles de service : `ECRBuildAndPublish` action.	22 novembre 2024
La prise en charge des `InspectorScan` actions Amazon Inspector a été ajoutée. Consultez Autorisations relatives aux rôles de service : `InspectorScan` action.	22 novembre 2024

Service AWS	CodePipeline date de support
Ajout du support d'action des commandes. Consultez Autorisations relatives aux rôles de service : action des commandes.	03 octobre 2024
CloudFormation support d'action ajouté. Consultez Autorisations relatives aux rôles de service : `CloudFormationStackSet` action et Autorisations relatives aux rôles de service : `CloudFormationStackInstances` action.	30 décembre 2020
CodeCommit support d'action complet au format d'artefact de sortie par clone ajouté. Consultez Autorisations relatives aux rôles de service : CodeCommit action.	11 novembre 2020
AWS CodeBuild Ajout d'un support d'action pour les builds par lots. Consultez Autorisations relatives aux rôles de service : CodeCommit action.	30 juillet 2020
AWS AppConfig support d'action ajouté. Consultez Autorisations relatives aux rôles de service : `AppConfig` action.	22 juin 2020
AWS Step Functions support d'action ajouté. Consultez Autorisations relatives aux rôles de service : `StepFunctions` action.	27 mai 2020
AWS CodeStar Ajout de la prise en charge des actions de connexion. Consultez Autorisations relatives aux rôles de service : AWS CodeConnections action.	18 décembre 2019

Service AWS	CodePipeline date de support
Ajout de la prise en charge des actions de déploiement dans S3. Consultez Autorisations relatives aux rôles de service : action de déploiement S3.	16 janvier 2019
Le support `CodeDeployToECS` d'action a été ajouté. Consultez Autorisations relatives aux rôles de service : `CodeDeployToECS` action.	27 novembre 2018
Le support d'action Amazon ECR a été ajouté. Consultez Autorisations relatives aux rôles de service : action Amazon ECR.	27 novembre 2018
La prise en charge des actions du Service Catalog a été ajoutée. Consultez Autorisations relatives aux rôles de service : action Service Catalog.	16 octobre 2018
AWS Device Farm support d'action ajouté. Consultez Autorisations relatives aux rôles de service : AWS Device Farm action.	19 juillet 2018
Le support d'action Amazon ECS a été ajouté. Consultez Autorisations relatives aux rôles de service : action standard Amazon ECS.	12 décembre 2017/ Mise à jour concernant l'acceptation de l'autorisation de marquage le 21 juillet 2017
CodeCommit support d'action ajouté. Consultez Autorisations relatives aux rôles de service : CodeCommit action.	18 avril 2016
AWS OpsWorks support d'action ajouté. Consultez Autorisations relatives aux rôles de service : AWS OpsWorks action.	2 juin 2016
CloudFormation support d'action ajouté. Consultez Autorisations relatives aux rôles de service : CloudFormation action.	3 novembre 2016

Service AWS	CodePipeline date de support
AWS CodeBuild support d'action ajouté. Consultez Autorisations relatives aux rôles de service : CodeBuild action.	1er décembre 2016
Le support d'action Elastic Beanstalk a été ajouté. Consultez Autorisations relatives aux rôles de service : action de `ElasticBeanstalk` déploiement.	Lancement initial du service
CodeDeploy support d'action ajouté. Consultez Autorisations relatives aux rôles de service : AWS CodeDeploy action.	Lancement initial du service
Ajout de la prise en charge des actions source S3. Consultez Autorisations relatives aux rôles de service : action source S3.	Lancement initial du service

Procédez comme suit pour ajouter des autorisations pour un service pris en charge :

1. Connectez-vous à la console IAM Console de gestion AWS et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le volet de navigation de la console IAM, sélectionnez Rôles, puis choisissez votre `AWS-CodePipeline-Service` rôle dans la liste des rôles.
3. Dans l'onglet Autorisations, dans Politiques intégrées, dans la ligne correspondant à votre politique de rôle de service, choisissez Modifier la politique.
4. Ajoutez les autorisations requises dans la zone du document de politique.

> **Note**
>
> Lorsque vous créez des politiques IAM, suivez les conseils de sécurité standard qui consistent à accorder le moindre privilège, c'est-à-dire à n'accorder que les autorisations nécessaires à l'exécution d'une tâche. Certains appels d'API prennent en charge les autorisations basées sur les ressources et autorisent la limitation d'accès. Par exemple, dans ce cas, pour limiter les autorisations lors de l'appel de `DescribeTasks`

et de `ListTasks`, vous pouvez remplacer le caractère générique (*) par un ARN de ressource ou par un ARN de ressource contenant un caractère générique (*). Pour plus d'informations sur la création d'une politique accordant un accès avec le moindre privilège, consultez. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege

5. Choisissez Examiner une stratégie afin de vérifier que la stratégie ne contient aucune erreur. Lorsque la politique est exempte d'erreurs, choisissez Appliquer la politique.

Connexion et surveillance CodePipeline

Vous pouvez utiliser les fonctionnalités de connexion AWS pour déterminer les actions effectuées par les utilisateurs sur votre compte et les ressources utilisées. Les fichiers journaux affichent :

- La date et l'heure des actions
- L'adresse IP source d'une action
- Les actions qui ont échoué en raison d'autorisations inadaptées

Les fonctionnalités de journalisation sont disponibles dans les catégories suivantes Services AWS :

- AWS CloudTrail peut être utilisé pour enregistrer les appels AWS d'API et les événements connexes effectués par ou pour le compte d'un Compte AWS. Pour plus d'informations, consultez Journalisation des appels d' CodePipeline API avec AWS CloudTrail.
- Amazon CloudWatch Events peut être utilisé pour surveiller vos AWS Cloud ressources et les applications que vous utilisez AWS. Vous pouvez créer des alertes dans Amazon CloudWatch Events en fonction des métriques que vous définissez. Pour plus d'informations, consultez Surveillance des CodePipeline événements.

Validation de conformité pour AWS CodePipeline

Notre nouvelle expérience AWS d'inscription n'est pas conçue pour les charges de travail réglementées. Si vous utilisez notre nouvelle expérience d' AWS inscription, mais que vous souhaitez l'utiliser AWS pour des charges de travail réglementées, vous pouvez vous inscrire à des fonctionnalités AWS (avancées) ou activer des fonctionnalités avancées pour votre AWS environnement.

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide d'un artefact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. Pour plus d'informations sur votre responsabilité en matière de conformité lors de l'utilisation Services AWS, consultez AWS la documentation de sécurité.

Résilience dans AWS CodePipeline

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

Sécurité de l'infrastructure dans AWS CodePipeline

En tant que service géré, AWS CodePipeline il est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder CodePipeline via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.

- Des suites de chiffrement dotées d'un secret de transmission parfait (PFS), telles que DHE (Ephemeral) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

Bonnes pratiques de sécurité

CodePipeline fournit un certain nombre de fonctionnalités de sécurité à prendre en compte lors de l'élaboration et de la mise en œuvre de vos propres politiques de sécurité. Les bonnes pratiques suivantes doivent être considérées comme des instructions générales et ne représentent pas une solution de sécurité complète. Étant donné que ces bonnes pratiques peuvent ne pas être appropriées ou suffisantes pour votre environnement, considérez-les comme des remarques utiles plutôt que comme des recommandations.

Vous utilisez des processus de chiffrement et d'authentification pour les référentiels source qui se connectent à vos pipelines. Voici les CodePipeline meilleures pratiques en matière de sécurité :

- Si vous créez une configuration de pipeline ou d'action qui doit inclure des secrets, tels que des jetons ou des mots de passe, n'entrez pas de secrets directement dans la configuration de l'action, ni les valeurs par défaut des variables définies au niveau du pipeline ou de la CloudFormation configuration, car les informations s'afficheront dans les journaux. Utilisez Secrets Manager pour configurer et stocker les secrets, puis utilisez le secret référencé dans le pipeline et la configuration des actions, comme décrit dansUtilisation AWS Secrets Manager pour suivre les mots de passe de base de données ou les clés d'API tierces.
- Si vous créez un pipeline qui utilise un compartiment source S3, configurez le chiffrement côté serveur pour les artefacts stockés dans Amazon S3 en les CodePipeline gérant AWS KMS keys, comme décrit dans. Configurer le chiffrement côté serveur pour les artefacts stockés dans Amazon S3 pour CodePipeline
- Si vous utilisez le fournisseur d'action Jenkins, lorsque vous utilisez un fournisseur de génération Jenkins pour l'action de génération ou de test de votre pipeline, installez Jenkins sur une instance EC2 et configurez un profil d'instance EC2 distinct. Assurez-vous que le profil d'instance accorde à Jenkins uniquement les AWS autorisations nécessaires pour effectuer des tâches relatives à votre projet, telles que la récupération de fichiers depuis Amazon S3. Pour apprendre à créer le rôle pour votre profil d'instance Jenkins, consultez les étapes de Création d'un rôle IAM à utiliser pour l'intégration de Jenkins.

CodePipeline référence de la structure du pipeline

Vous pouvez l'utiliser CodePipeline pour structurer un CI/CD pipeline d'étapes automatisées qui accomplissent des tâches de création, de test et de déploiement du code source de votre application. Cette section de référence fournit des détails sur la structure et les paramètres JSON de votre pipeline. Pour une liste détaillée des concepts qui décrivent la façon dont les pipelines sont utilisés, voirCodePipeline concepts.

- Lorsque vous créez un pipeline, vous choisissez une action source et un fournisseur disponibles, tels qu'un bucket S3, un CodeCommit référentiel, un référentiel Bitbucket ou un GitHub référentiel qui contient votre code source et démarre votre pipeline lorsque vous validez une modification de code source. Cette section de référence fournit des informations de référence sur les sources disponibles pour votre pipeline. Pour plus d'informations sur l'utilisation des actions source, consultezDémarrez un pipeline dans CodePipeline.
- Vous pouvez choisir les actions et les fournisseurs de test, de génération et de déploiement que vous souhaitez inclure automatiquement lors de l'exécution de votre pipeline. Cette section de référence fournit des informations de référence sur les actions disponibles et leur intégration dans le JSON de votre pipeline.
- Votre pipeline terminé comprendra une étape source ainsi que des étapes supplémentaires au cours desquelles vous configurerez des actions pour déployer et tester votre application. Pour un exemple conceptuel de DevOps pipeline qui déploie votre application, consultezDevOps exemple de pipeline.

Par défaut, tout pipeline que vous créez avec succès AWS CodePipeline possède une structure valide. Toutefois, si vous créez ou modifiez manuellement un fichier JSON pour créer un pipeline ou mettre à jour un pipeline à partir du AWS CLI, vous risquez de créer par inadvertance une structure non valide. La référence suivante vous aidera à mieux comprendre les exigences relatives à la structure de votre pipeline et à remédier à d'éventuels problèmes. Examinez les contraintes dans Quotas dans AWS CodePipeline, qui s'appliquent à tous les pipelines.

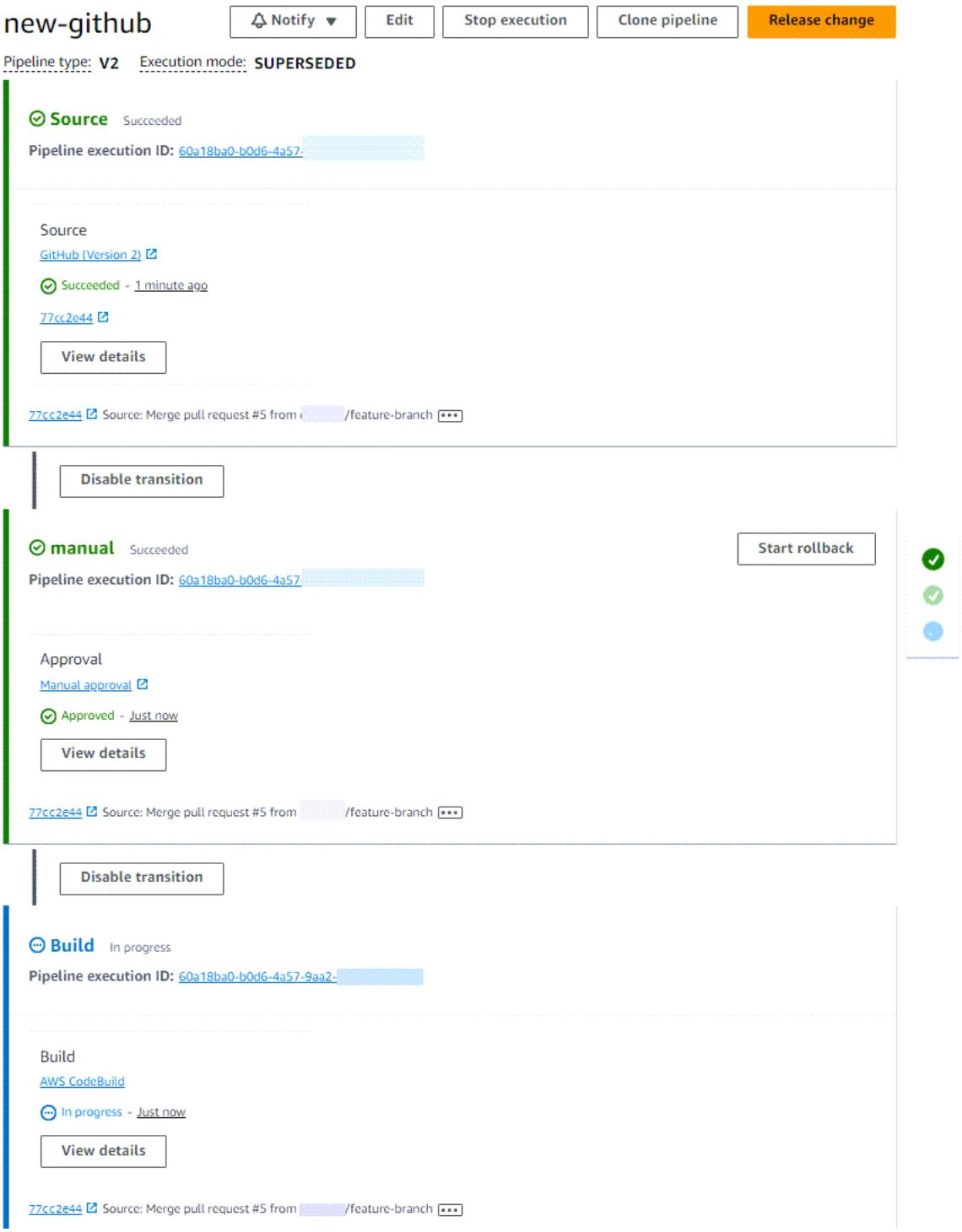
Les sections suivantes fournissent les paramètres de haut niveau et leur position dans la structure du pipeline. Les exigences relatives à la structure des pipelines sont détaillées dans chaque section pour les types de composants de pipeline suivants :

- Champ de référence pour Déclaration sur le pipeline

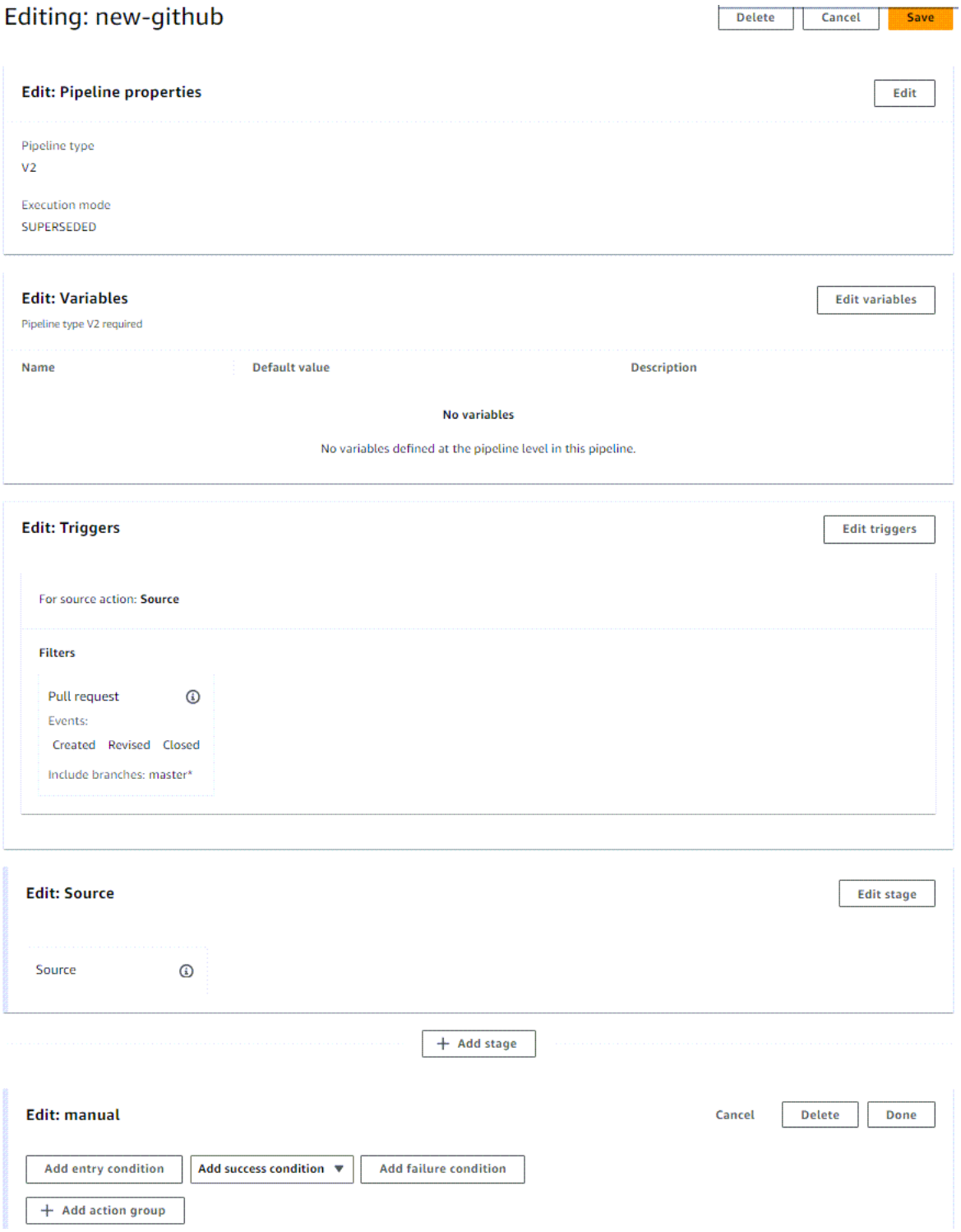
- Champ de référence pour Déclaration d'étape
- Champ de référence pour Déclaration d'action
- Liste Fournisseurs d'actions valides dans CodePipeline par type d'action
- Référence pour Réglages valides pour le `PollForSourceChanges` paramètre
- Référence pour Artefacts d'entrée et de sortie valides pour chaque type d'action
- Liste de liens pour Paramètres de configuration valides pour chaque type de fournisseur

Pour plus d'informations, consultez l'PipelineDeclarationobjet dans le Guide de CodePipeline l'API.

L'exemple de vue de console de pipeline suivant montre le pipeline nommé new-github, les étapes nomméES`Source`, et `manualBuild`, et les actions effectuées GitHub (via GitHub App), l'approbation manuelle et les fournisseurs d' CodeBuild actions.



Le mode d'édition du pipeline, lorsqu'il est affiché dans le schéma de la console, vous permet de modifier les remplacements, les déclencheurs et les actions de source, comme indiqué dans l'exemple suivant.



Rubriques

- Déclaration sur le pipeline
- Déclaration d'étape
- Déclaration d'action

- Fournisseurs d'actions valides dans CodePipeline
- Réglages valides pour le PollForSourceChanges paramètre
- Artefacts d'entrée et de sortie valides pour chaque type d'action
- Paramètres de configuration valides pour chaque type de fournisseur

Déclaration sur le pipeline

Le pipeline et le niveau de métadonnées d'un pipeline ont une structure de base qui inclut les paramètres et la syntaxe suivants. Le paramètre du pipeline représente la structure des actions et des étapes à exécuter dans le pipeline.

Pour plus d'informations, consultez l'PipelineDeclarationobjet dans le Guide de CodePipeline l'API.

L'exemple suivant montre le pipeline et le niveau de métadonnées de la structure du pipeline à la fois en JSON et en YAML pour un pipeline de type 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

Nom du pipeline. Lorsque vous modifiez ou mettez à jour un pipeline, le nom du pipeline ne peut pas être modifié.

> **Note**
>
> Si vous souhaitez renommer un pipeline existant, vous pouvez utiliser la commande CLI `get-pipeline` pour générer un fichier JSON contenant la structure de votre pipeline. Par la suite, vous pouvez utiliser la commande CLI `create-pipeline` pour créer un pipeline avec cette structure et lui attribuer un nouveau nom.

`roleArn`

L'ARN IAM pour le rôle de CodePipeline service, tel que arn:aws:iam : :80398 _. EXAMPLE:role/ CodePipeline Service_Role

Pour utiliser la console afin d'afficher l'ARN du rôle de service de pipeline au lieu de la structure JSON, choisissez votre pipeline dans la console, puis choisissez Paramètres. Sous l'onglet Général, le champ ARN du rôle de service s'affiche.

`ArtifactStore` OU `ArtifactStores`

Le `artifactStore` champ contient le type de compartiment d'artefacts et l'emplacement d'un pipeline avec toutes les actions dans la même AWS région. Si vous ajoutez des actions dans une région différente de celle de votre pipeline, le `artifactStores` mappage est utilisé pour répertorier le compartiment d'artefacts pour chaque AWS région dans laquelle les actions sont exécutées. Lorsque vous créez ou modifiez un pipeline, vous devez avoir un compartiment d'artefact dans le pipeline Région, puis vous devez disposer d'un compartiment d'artefact par région dans laquelle vous prévoyez d'exécuter une action.

> **Note**
>
> Dans la structure du pipeline, vous devez inclure l'un `artifactStore` ou l'autre `artifactStores` dans votre pipeline, mais vous ne pouvez pas utiliser les deux. Si vous créez une action entre régions dans votre pipeline, vous devez utiliser `artifactStores`.

L'exemple suivant illustre la structure de base d'un pipeline avec des actions inter-régions qui utilise le paramètre `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`

Type d'emplacement pour le compartiment d'artefacts, spécifié comme Amazon S3.

`location`

Le nom du compartiment Amazon S3 généré automatiquement pour vous la première fois que vous créez un pipeline à l'aide de la console, tel que codepipeline-us-east-2-1234567890, ou tout compartiment Amazon S3 que vous provisionnez à cette fin

`étapes`

Ce paramètre contient le nom de chaque étape du pipeline. Pour plus d'informations sur les paramètres et la syntaxe au niveau de l'étape de la structure du pipeline, consultez l'StageDeclarationobjet du Guide de l' CodePipeline API.

La structure du pipeline pour les étapes répond aux exigences suivantes :

- Un pipeline doit contenir au moins deux étapes.
- La première étape d'un pipeline doit contenir au moins une action source. Elle peut contenir des actions source uniquement.
- Seule la première étape d'un pipeline peut contenir des actions source.
- Au moins une étape dans chaque pipeline doit comporter une action autre qu'une action source.

- Tous les noms d'étapes dans un pipeline doivent être uniques.
- Les noms des scènes ne peuvent pas être modifiés dans la CodePipeline console. Si vous modifiez le nom d'un stage à l'aide de AWS CLI, et que le stage contient une action avec un ou plusieurs paramètres secrets (comme un jeton OAuth), la valeur de ces paramètres secrets n'est pas préservée. Vous devez entrer manuellement les valeurs des paramètres (qui sont masquées par quatre astérisques dans le JSON renvoyé par l'interface AWS CLI) et les inclure dans la structure JSON.

> **Important**
>
> Les sondages seront désactivés pour les pipelines qui sont inactifs pendant plus de 30 jours. Pour plus d'informations, consultez la section « sondage » DisabledAt dans la référence sur la structure du pipeline. Pour connaître les étapes à suivre pour migrer votre pipeline de la détection des modifications par sondage vers la détection des modifications basée sur les événements, consultez la section Méthodes de détection des modifications.

`version`

Le numéro de version d'un pipeline est automatiquement généré et mis à jour chaque fois que vous mettez à jour le pipeline.

`executionMode`

Vous pouvez définir le mode d'exécution du pipeline afin de pouvoir spécifier le comportement du pipeline pour des exécutions consécutives, telles que la mise en file d'attente, le remplacement ou l'exécution en mode parallèle. Pour de plus amples informations, veuillez consulter Définir ou modifier le mode d'exécution du pipeline.

> **Important**
>
> Pour les pipelines en mode PARALLÈLE, la restauration des étapes n'est pas disponible. De même, les conditions de défaillance avec un type de résultat d'annulation ne peuvent pas être ajoutées à un pipeline en mode PARALLÈLE.

Type de pipeline

Le type de pipeline spécifie la structure et les fonctionnalités disponibles dans le pipeline, par exemple pour un pipeline de type V2. Pour de plus amples informations, veuillez consulter Types de canalisations.

variables

Les variables au niveau du pipeline sont définies lorsque le pipeline est créé et résolues au moment de l'exécution du pipeline. Pour de plus amples informations, veuillez consulter Référence des variables. Pour un didacticiel avec une variable au niveau du pipeline qui est transmise au moment de l'exécution du pipeline, consultez. Tutoriel : Utiliser des variables au niveau du pipeline

déclencheurs

Les déclencheurs vous permettent de configurer votre pipeline pour qu'il démarre sur un type d'événement particulier ou un type d'événement filtré, par exemple lorsqu'une modification sur une branche particulière ou une pull request est détectée. Les déclencheurs sont configurables pour les actions source avec des connexions qui utilisent l'`CodeStarSourceConnection`action dans CodePipeline, par exemple GitHub, Bitbucket et GitLab. Pour plus d'informations sur les actions source qui utilisent des connexions, consultezAjoutez des fournisseurs de sources tiers aux pipelines à l'aide de AWS CodeConnections.

Pour plus d'informations et des exemples plus détaillés, consultezAutomatisez le démarrage des pipelines en utilisant des déclencheurs et des filtres.

Pour le filtrage, les modèles d'expressions régulières au format glob sont pris en charge, comme indiqué dansUtilisation de modèles globulaires dans la syntaxe.

> **Note**
>
> Les actions source CodeCommit et S3 nécessitent soit une ressource de détection des modifications configurée (une EventBridge règle), soit l'utilisation de l'option permettant d'interroger le référentiel pour détecter les modifications apportées à la source. Pour les pipelines comportant une action source Bitbucket ou GitHub Enterprise Server, il n'est pas nécessaire de configurer un webhook ou de lancer un sondage par défaut. GitHub L'action Connections gère la détection des modifications pour vous.

> **Important**
>
> Les sondages seront désactivés pour les pipelines qui sont inactifs pendant plus de 30 jours. Pour plus d'informations, consultez la section « sondage » DisabledAt dans la référence sur la structure du pipeline. Pour connaître les étapes à suivre pour migrer votre pipeline de la détection des modifications par sondage vers la détection des modifications basée sur les événements, consultez la section Méthodes de détection des modifications.

Champs de configuration Git

La configuration Git du déclencheur, y compris les types d'événements et tous les paramètres de filtrage par branches, chemins de fichiers, balises ou événements de pull request.

Les champs de la structure JSON sont définis comme suit :

- `sourceActionName`: nom de l'action source du pipeline avec la configuration Git.
- `push`: événements push avec filtrage. Ces événements utilisent une opération OR entre différents filtres push et une opération ET à l'intérieur des filtres.
  - `branches`: Les branches sur lesquelles filtrer. Les branches utilisent une opération ET entre les inclusions et les exclusions.
    - `includes`: modèles à filtrer pour les branches qui seront incluses. Inclut l'utilisation d'une opération OR.
    - `excludes`: modèles à filtrer pour les branches qui seront exclues. Exclut l'utilisation d'une opération OR.
  - `filePaths`: les noms des chemins de fichier sur lesquels filtrer.
    - `includes`: modèles à utiliser pour filtrer les chemins de fichiers qui seront inclus. Inclut l'utilisation d'une opération OR.
    - `excludes`: modèles à utiliser pour filtrer les chemins de fichiers qui seront exclus. Exclut l'utilisation d'une opération OR.
  - `tags`: Les noms de balises sur lesquels filtrer.
    - `includes`: modèles à filtrer pour les balises qui seront incluses. Inclut l'utilisation d'une opération OR.
    - `excludes`: modèles sur lesquels filtrer les balises qui seront exclues. Exclut l'utilisation d'une opération OR.

- `pullRequest`: événements de demande d'extraction avec filtrage sur les événements de demande d'extraction et filtres de demande d'extraction.
  - `events`: filtre les événements de pull request ouverts, mis à jour ou fermés, comme spécifié.
  - `branches`: Les branches sur lesquelles filtrer. Les branches utilisent une opération ET entre les inclusions et les exclusions.
    - `includes`: modèles à filtrer pour les branches qui seront incluses. Inclut l'utilisation d'une opération OR.
    - `excludes`: modèles à filtrer pour les branches qui seront exclues. Exclut l'utilisation d'une opération OR.
  - `filePaths`: les noms des chemins de fichier sur lesquels filtrer.
    - `includes`: modèles à utiliser pour filtrer les chemins de fichiers qui seront inclus. Inclut l'utilisation d'une opération OR.
    - `excludes`: modèles à utiliser pour filtrer les chemins de fichiers qui seront exclus. Exclut l'utilisation d'une opération OR.

Voici un exemple de configuration des déclencheurs pour les types d'événements de type push et 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"
                        ]
                    }
                }
            ]
```

```
                }
            }
        ],
```

Champs push de type d'événement pour inclure et exclure

Le comportement d'inclusion et d'exclusion pour les niveaux des champs de configuration Git pour les types d'événements push est illustré dans la liste suivante :

```
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)
```

Champs de demande d'extraction de type d'événement pour inclure et exclure

Le comportement d'inclusion et d'exclusion pour les niveaux des champs de configuration Git pour les types d'événements de pull request est illustré dans la liste suivante :

```
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

Les champs des métadonnées du pipeline sont différentes de la structure du pipeline et ne peuvent pas être modifiés. Lorsque vous mettez à jour un pipeline, la date du champ des métadonnées `updated` change automatiquement.

pipelineArn

Le nom de ressource Amazon (ARN) du pipeline.

Pour utiliser la console afin d'afficher l'ARN du pipeline au lieu de la structure JSON, choisissez votre pipeline dans la console, puis choisissez Paramètres. Sous l'onglet Général, le champ Pipeline ARN s'affiche.

créé

Date et heure de création du pipeline.

updated

Date et heure de la dernière mise à jour du pipeline.

sondage DisabledAt

Date et heure auxquelles, pour un pipeline configuré pour le sondage pour la détection des modifications, le sondage a été désactivé.

Les sondages seront désactivés pour les pipelines qui sont inactifs pendant plus de 30 jours.

- Les sondages seront désactivés pour les pipelines inactifs après 30 jours sans exécution.
- Les pipelines utilisant EventBridge des CodeStar connexions ou des webhooks ne seront pas affectés.
- Les pipelines actifs ne seront pas affectés.

Pour plus d'informations, consultez le `pollingDisabledAt` paramètre sous PipelineMetadata object dans le Guide de l' CodePipeline API. Pour connaître les étapes à suivre pour migrer votre

pipeline de la détection des modifications par sondage vers la détection des modifications basée sur les événements, consultez la section Méthodes de détection des modifications.

Déclaration d'étape

Le niveau d'étape d'un pipeline possède une structure de base qui inclut les paramètres et la syntaxe suivants. Pour plus d'informations, consultez l'StageDeclarationobjet dans le Guide de CodePipeline l'API.

L'exemple suivant montre le niveau d'étape de la structure du pipeline à la fois en JSON et en YAML. L'exemple montre deux étapes nommées `Source` et`Build`. L'exemple contient deux conditions, l'une pour `onSuccess` et l'autre pour`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

Nom de l'étape .

actions

Le niveau d'action d'un pipeline possède une structure de base qui inclut les paramètres et la syntaxe suivants. Pour consulter les paramètres et des exemples, consultezDéclaration d'action.

`conditions`

Les conditions contiennent une ou plusieurs règles qui sont disponibles dans une liste de règles dans CodePipeline. Si toutes les règles d'une condition réussissent, la condition est remplie. Vous pouvez configurer les conditions de sorte que lorsque les critères ne sont pas remplis, le résultat spécifié soit activé.

Vous pouvez configurer les types de conditions suivants :

- `beforeEntry`
- `onFailure`
- `onSuccess`

Pour plus d'informations et d'exemples, consultez Configuration des conditions pour une étape.

`rules`

Chaque condition possède un ensemble de règles qui est un ensemble ordonné de règles évaluées ensemble. Par conséquent, si une règle échoue dans la condition, la condition échoue. Vous pouvez modifier les conditions des règles lors de l'exécution du pipeline.

Les règles disponibles sont fournies dans la référence des règles. Pour plus d'informations, consultez la référence sur la structure des règles à l'adresseRéférence de structure de règles.

Déclaration d'action

Le niveau d'action d'un pipeline possède une structure de base qui inclut les paramètres et la syntaxe suivants. Pour plus d'informations, consultez l'ActionDeclarationobjet dans le Guide de CodePipeline l'API.

L'exemple suivant montre le niveau d'action de la structure du pipeline à la fois en JSON et en YAML.

YAML

```
. . .

  stages:
    - name: Source
      actions:
```

```
        - name: Source
          actionTypeId:
            category: Source
            owner: AWS
            provider: S3
            version: '1'
          runOrder: 1
          configuration:
            PollForSourceChanges: 'false'
            S3Bucket: amzn-s3-demo-bucket
            S3ObjectKey: codedeploy_linux.zip
          outputArtifacts:
            - name: SourceArtifact
          inputArtifacts: []
          region: us-west-2
          namespace: SourceVariables
    - name: Build
      actions:
        - name: Build
          actionTypeId:
            category: Build
            owner: AWS
            provider: CodeBuild
            version: '1'
          runOrder: 1
          configuration:
            EnvironmentVariables: >-
              [{"name":"ETag","value":"#{SourceVariables.ETag}","type":"PLAINTEXT"}]
            ProjectName: my-project
          outputArtifacts:
            - name: BuildArtifact
          inputArtifacts:
            - name: SourceArtifact
          region: us-west-2
          namespace: BuildVariables
          runOrder: 1
          configuration:
            CustomData: >-
              Here are the exported variables from the build action: S3 ETAG:
              #{BuildVariables.ETag}
          outputArtifacts: []
          inputArtifacts: []
          region: us-west-2
```

JSON

```
. . .

        "stages": [
            {
                "name": "Source",
                "actions": [
                    {
                        "name": "Source",
                        "actionTypeId": {
                            "category": "Source",
                            "owner": "AWS",
                            "provider": "S3",
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "PollForSourceChanges": "false",
                            "S3Bucket": "amzn-s3-demo-bucket",
                            "S3ObjectKey": "aws-codepipeline-s3-aws-
codedeploy_linux.zip"
                        },
                        "outputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "inputArtifacts": [],
                        "region": "us-west-2",
                        "namespace": "SourceVariables"
                    }
                ]
            },
            {
                "name": "Build",
                "actions": [
                    {
                        "name": "Build",
                        "actionTypeId": {
                            "category": "Build",
                            "owner": "AWS",
                            "provider": "CodeBuild",
```

```
                            "version": "1"
                        },
                        "runOrder": 1,
                        "configuration": {
                            "EnvironmentVariables": "[{\"name\":\"ETag\",\"value\":
\"#{SourceVariables.ETag}\",\"type\":\"PLAINTEXT\"}]",
                            "ProjectName": "my-build-project"
                        },
                        "outputArtifacts": [
                            {
                                "name": "BuildArtifact"
                            }
                        ],
                        "inputArtifacts": [
                            {
                                "name": "SourceArtifact"
                            }
                        ],
                        "region": "us-west-2",
                        "namespace": "BuildVariables"
                    }
                ]

. . .
```

Pour obtenir la liste des exemples des détails de `configuration` correspondant au type de fournisseur, consultez Paramètres de configuration valides pour chaque type de fournisseur.

La structure de l'action impose les critères suivants :

- Tous les noms des actions d'une étape doivent être uniques.
- Une action à la source est requise pour chaque pipeline.
- Les actions de source qui n'utilisent pas de connexion peuvent être configurées pour détecter les modifications ou pour désactiver la détection des modifications. Voir Méthodes de détection des modifications.
- Cela s'applique à toutes les actions, qu'elles se trouvent dans la même étape ou dans les étapes suivantes, mais l'artefact d'entrée ne doit pas nécessairement être l'action suivant immédiatement celle ayant fourni l'artefact de sortie. Les actions en parallèle peuvent déclarer différents lots d'artefacts de sortie, qui sont ensuite utilisés par différentes actions subséquentes.

- Lorsque vous utilisez un compartiment Amazon S3 comme lieu de déploiement, vous spécifiez également une clé d'objet. Une clé d'objet peut être un nom de fichier (objet) ou une combinaison d'un préfixe (chemin d'accès à un dossier) et d'un nom de fichier. Vous pouvez utiliser des variables pour spécifier le nom de l'emplacement que vous souhaitez que le pipeline utilise. Les actions de déploiement Amazon S3 prennent en charge l'utilisation des variables suivantes dans les clés d'objet Amazon S3.

  Utilisation de variables dans Amazon S3

Variable	Exemple d'entrée de console	Output
datetime	js-application/{datetime}.zip	Horodatage UTC dans ce format : <AAAA>-<MM>-JJ>_<HH>-<MM>-<SS> Exemple : js- application/2019 -01-10_07-39-57.zip
uuid	js-application/{uuid}.zip	L'UUID est un identifiant unique international qui est garanti différent de n'importe quel autre identifiant. L'UUID est au format suivant (tous les chiffres au format hexadécimal) : <8 chiffres>-<4 chiffres>-4 chiffres>-<4 chiffres>-<12 chiffres> Exemple : js- application/54a60075 -b96a-4bf 3-9013-Db3a9 EXAMPLE.zip

`name`

Le nom de l'action.

region

Pour les actions où le fournisseur est un Service AWS, le Région AWS nom de la ressource.

Cross-Region les actions utilisent le `Region` champ pour indiquer l' Région AWS endroit où les actions doivent être créées. Les AWS ressources créées pour cette action doivent être créées dans la même région que celle prévue `region` sur le terrain. Vous ne pouvez pas créer d'actions inter-régions pour les types d'action suivants :

- Actions source
- Actions par des fournisseurs tiers
- Actions par des fournisseurs personnalisés

roleArn

ARN du rôle de service IAM qui effectue l'action déclarée. Cela est supposé par le biais du ROLearn qui est spécifié au niveau du pipeline.

espace de nom

Les actions peuvent être configurées avec des variables. Vous utilisez le champ `namespace` pour définir l'espace de noms et les informations de variable pour les variables d'exécution. Pour plus d'informations sur les variables d'exécution et les variables de sortie d'action, consultez Référence des variables.

> **Note**
>
> Pour Amazon ECR, Amazon S3 ou les CodeCommit sources, vous pouvez également créer un remplacement de source à l'aide d'une entrée de transformation en entrée EventBridge pour utiliser l'`revisionValue`entrée de votre événement de pipeline, où le `revisionValue` est dérivé de la variable d'événement source pour votre clé d'objet, votre commit ou votre ID d'image. Pour plus d'informations, consultez l'étape facultative pour la saisie de la transformation d'entrée incluse dans les procédures sous Actions et ressources relatives aux sources Amazon ECR EventBridgeConnexion aux actions de source Amazon S3 avec une source activée pour les événements, ouCodeCommit actions à la source et EventBridge.

action TypeId

L'ID du type d'action est identifié comme une combinaison des quatre champs suivants.

category

Type d'action, ou étape, du pipeline, telle qu'une action source. Chaque type d'action possède un ensemble spécifique de fournisseurs d'actions valides. Pour obtenir la liste des fournisseurs valides par type d'action, consultez leRéférence sur la structure des actions.

Voici les `actionTypeId` catégories valides (types d'actions) pour CodePipeline :

- `Source`
- `Build`
- `Approval`
- `Deploy`
- `Test`
- `Invoke`
- `Compute`

owner

Pour tous les types d'actions actuellement pris en charge, la seule chaîne propriétaire valide est `AWSThirdParty`, ou`Custom`. Pour connaître la chaîne propriétaire valide pour une action spécifique, consultez leRéférence sur la structure des actions.

Pour plus d'informations, consultez la Référence des API CodePipeline .

version

Version de l'action.

provider

Le fournisseur de l'action, tel que CodeBuild.

- Les types de fournisseur valides pour une catégorie d'action dépendent de la catégorie. Par exemple, pour une catégorie d'action source, un type de fournisseur valide est

`S3`CodeStarSourceConnection,`CodeCommit`, ou`Amazon ECR`. Cet exemple illustre la structure pour une action source avec un fournisseur `S3` :

```
"actionTypeId": {
  "category": "Source",
  "owner": "AWS",
  "version": "1",
  "provider": "S3"},
```

InputArtifacts

Ce champ contient la structure d'artefact en entrée, si elle est prise en charge pour la catégorie d'action. L'artefact d'entrée d'une action doit correspondre exactement à l'artefact de sortie déclaré lors d'une action précédente. Par exemple, si une action précédente comprend la déclaration suivante :

```
"outputArtifacts": [
    {
    "MyApp"
    }
],
```

et il n'y a pas d'autres artefacts de sortie, alors l'artefact d'entrée d'une action suivante doit être :

```
"inputArtifacts": [
    {
    "MyApp"
    }
],
```

Par exemple, une action source ne peut pas comporter d'artefacts d'entrée car il s'agit de la première action du pipeline. Cependant, une action source contiendra toujours des artefacts de sortie qui sont traités par l'action suivante. Les artefacts de sortie d'une action source sont les fichiers d'application du référentiel source, compressés et fournis via le compartiment d'artefacts, qui sont traités par l'action suivante, telle qu'une CodeBuild action agissant sur les fichiers de l'application à l'aide de commandes de génération.

À titre d'exemple d'actions qui ne peuvent pas comporter d'artefacts de sortie, les actions de déploiement ne comportent pas d'artefacts de sortie car il s'agit généralement de la dernière action d'un pipeline.

name

Le nom de l'artefact pour les artefacts d'entrée de l'action.

Artefacts de sortie

Les noms des artefacts de sortie doivent être uniques dans un pipeline. Par exemple, un pipeline peut inclure une action dotée d'un artefact de sortie nommé `"MyApp"` et une autre action dotée d'un artefact de sortie nommé `"MyBuiltApp"`. Cependant, un pipeline ne peut pas contenir deux actions contenant toutes deux un artefact de sortie nommé `"MyApp"`.

Ce champ contient la structure de l'artefact en sortie, si elle est prise en charge pour la catégorie d'action. L'artefact de sortie d'une action doit correspondre exactement à l'artefact de sortie déclaré lors d'une action précédente. Par exemple, si une action précédente comprend la déclaration suivante :

```
"outputArtifacts": [
    {
    "MyApp"
    }
],
```

et il n'y a pas d'autres artefacts de sortie, alors l'artefact d'entrée d'une action suivante doit être :

```
"inputArtifacts": [
    {
    "MyApp"
    }
],
```

Par exemple, une action source ne peut pas comporter d'artefacts d'entrée car il s'agit de la première action du pipeline. Cependant, une action source contiendra toujours des artefacts de sortie qui sont traités par l'action suivante. Les artefacts de sortie d'une action source sont les fichiers d'application du référentiel source, compressés et fournis via le compartiment d'artefacts, qui sont traités par l'action suivante, telle qu'une CodeBuild action agissant sur les fichiers de l'application à l'aide de commandes de génération.

À titre d'exemple d'actions qui ne peuvent pas comporter d'artefacts de sortie, les actions de déploiement ne comportent pas d'artefacts de sortie car il s'agit généralement de la dernière action d'un pipeline.

`name`

Le nom de l'artefact pour les artefacts de sortie de l'action.

`configuration` (par fournisseur d'actions)

La configuration des actions contient des détails et des paramètres adaptés au type de fournisseur. Dans la section ci-dessous, les exemples de paramètres de configuration d'action sont spécifiques à l'action source S3.

La configuration des actions et les limites d' input/output artefacts peuvent varier en fonction du fournisseur d'actions. Pour obtenir une liste d'exemples de configuration d'actions par fournisseur d'actions, consultez Référence sur la structure des actions et le tableau dansParamètres de configuration valides pour chaque type de fournisseur. Le tableau fournit un lien vers la référence des actions pour chaque type de fournisseur, qui répertorie en détail les paramètres de configuration de chaque action. Pour un tableau présentant les limites d'artefacts en entrée et en sortie pour chaque fournisseur d'actions, consultezArtefacts d'entrée et de sortie valides pour chaque type d'action.

Les considérations suivantes s'appliquent à l'utilisation d'actions :

- Les actions source n'ont pas d'artefacts d'entrée et les actions de déploiement n'ont pas d'artefacts de sortie.
- Pour les fournisseurs d'actions source qui n'utilisent pas de connexion, tels que S3, vous devez utiliser le `PollForSourceChanges` paramètre pour spécifier si vous souhaitez que votre pipeline démarre automatiquement lorsqu'une modification est détectée. Consultez Réglages valides pour le `PollForSourceChanges` paramètre.
- Pour configurer la détection automatique des modifications afin de démarrer votre pipeline ou pour désactiver la détection des modifications, consultezActions à la source et méthodes de détection des modifications.
- Pour configurer des déclencheurs avec filtrage, utilisez l'action source pour les connexions, puis consultezAutomatisez le démarrage des pipelines en utilisant des déclencheurs et des filtres.
- Pour les variables de sortie pour chaque action, consultezRéférence des variables.

> **Note**
>
> Pour Amazon ECR, Amazon S3 ou les CodeCommit sources, vous pouvez également créer un remplacement de source à l'aide d'une entrée de transformation en entrée EventBridge pour utiliser l'`revisionValue`entrée de votre événement de pipeline, où le `revisionValue` est dérivé de la variable d'événement source pour votre clé d'objet, votre commit ou votre ID d'image. Pour plus d'informations, consultez l'étape facultative pour la saisie de la transformation d'entrée incluse dans les procédures sous Actions et ressources relatives aux sources Amazon ECR EventBridgeConnexion aux actions de source Amazon S3 avec une source activée pour les événements, ouCodeCommit actions à la source et EventBridge.

> **Important**
>
> Les sondages seront désactivés pour les pipelines qui sont inactifs pendant plus de 30 jours. Pour plus d'informations, consultez la section « sondage » DisabledAt dans la référence sur la structure du pipeline. Pour connaître les étapes à suivre pour migrer votre pipeline de la détection des modifications par sondage vers la détection des modifications basée sur les événements, consultez la section Méthodes de détection des modifications.

> **Note**
>
> Les actions source CodeCommit et S3 nécessitent soit une ressource de détection des modifications configurée (une EventBridge règle), soit l'utilisation de l'option permettant d'interroger le référentiel pour détecter les modifications apportées à la source. Pour les pipelines comportant une action source Bitbucket ou GitHub Enterprise Server, il n'est pas nécessaire de configurer un webhook ou de lancer un sondage par défaut. GitHub L'action Connections gère la détection des modifications pour vous.

`Ordre d'exécution`

Nombre entier positif qui indique l'ordre d'exécution de l'action au cours de l'étape. Les actions parallèles de l'étape sont affichées comme ayant le même entier. Par exemple, deux actions avec un ordre d'exécution de deux seront exécutées en parallèle après la première action de la phase.

La valeur par défaut `runOrder` pour une action est 1. La valeur doit être un nombre entier positif (nombre naturel). Vous ne pouvez pas utiliser des fractions, des nombres décimaux, négatifs ou zéro. Pour spécifier une séquence d'actions en série, utilisez le plus petit nombre pour la première action et les plus grands nombres pour chacune des autres actions de la séquence. Pour spécifier des actions parallèles, utilisez le même nombre entier pour chaque action que vous souhaitez exécuter en parallèle. Dans la console, vous pouvez spécifier une séquence en série pour une action en choisissant Ajouter un groupe d'actions au niveau de l'étape où vous souhaitez qu'elle s'exécute, ou vous pouvez spécifier une séquence parallèle en choisissant Ajouter une action. Le groupe d'actions fait référence à l'ordre d'exécution d'une ou de plusieurs actions au même niveau.

Par exemple, si vous souhaitez que trois actions s'exécutent en séquence dans une étape, vous devez attribuer à la première action `runOrder` la valeur 1, à la seconde action `runOrder` la valeur 2 et à la troisième `runOrder` la valeur 3. Toutefois, si vous souhaitez que les deuxième et troisième actions s'exécutent en parallèle, vous devez attribuer à la première action la `runOrder` valeur 1 et aux deuxième et troisième actions `runOrder` la valeur 2.

> **Note**
>
> La numérotation des actions en série ne doit pas nécessairement être dans un ordre strict. Par exemple, si vous avez trois actions dans une séquence et que vous décidez de supprimer la seconde action, vous n'avez pas besoin de réordonner la `runOrder` valeur de la troisième action. Comme la valeur `runOrder` de cette action (3) est supérieure à la valeur `runOrder` de la première action (1), elle s'exécute en série après la première action dans cette étape.

Fournisseurs d'actions valides dans CodePipeline

Le format de la structure du pipeline est utilisé pour générer les actions et les étapes d'un pipeline. Un type d'action est constitué d'une catégorie d'action et d'un type de fournisseur.

Chaque catégorie d'action possède une liste valide de fournisseurs d'actions. Pour référencer les fournisseurs d'actions valides pour chaque catégorie d'action, consultez leRéférence sur la structure des actions.

Chaque catégorie d'action possède un ensemble désigné de fournisseurs. Chaque fournisseur d'actions, tel qu'Amazon S3, possède un nom de fournisseur, par exemple`S3`, qui doit être utilisé dans le `Provider` champ de la catégorie d'action de votre structure de pipeline.

Il existe trois valeurs valides pour le champ `Owner` dans la section de catégorie d'action de votre structure de pipeline : `AWS`, `ThirdParty`, et `Custom`.

Pour trouver le nom du fournisseur et les informations sur le propriétaire de votre fournisseur d'actions, consultez le Référence sur la structure des actions ouArtefacts d'entrée et de sortie valides pour chaque type d'action.

Ce tableau répertorie les fournisseurs valides par type d'action.

> **Note**
>
> Pour les actions Bitbucket ou GitHub Enterprise Server, reportez-vous à la CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées rubrique de référence des actions. GitHub

Fournisseurs d'actions valides par type d'action

Catégorie d'actions	Fournisseurs d'actions valides	Type de pipeline pris en charge	Référence sur les actions
Source	Amazon S3	V1 ET V2	Référence des actions source d'Amazon S3
	Amazon ECR	V1 ET V2	Référence des actions source Amazon ECR
	CodeCommit	V1 ET V2	CodeCommit référence de l'action source
	CodeStarSourceConnection (pour les actions Bitbucket	V1 ET V2	CodeStarSourceConnection pour Bitbucket

Catégorie d'actions	Fournisseurs d'actions valides	Type de pipeline pris en charge	Référence sur les actions
	GitHub, GitHub Enterprise Server)		Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées
Génération	Action Amazon ECR `ECRBuildAndPublish`	V2 uniquement	`ECRBuildAndPublish` créer une référence d'action
	CodeBuild	V1 ET V2	AWS CodeBuild créer et tester une référence d'action
	Action des commandes (voir Calculer)	V2 uniquement	
	Personnalisé CloudBees	V1 ET V2	Artefacts d'entrée et de sortie valides pour chaque type d'action
	Custom Jenkins	V1 ET V2	Artefacts d'entrée et de sortie valides pour chaque type d'action
	Personnalisé TeamCity	V1 ET V2	Artefacts d'entrée et de sortie valides pour chaque type d'action
test	CodeBuild	V1 ET V2	AWS CodeBuild créer et tester une référence d'action
	AWS Device Farm	V1 ET V2	Artefacts d'entrée et de sortie valides pour chaque type d'action

Catégorie d'actions	Fournisseurs d'actions valides	Type de pipeline pris en charge	Référence sur les actions
	Personnalisé BlazeMeter	V1 ET V2	Artefacts d'entrée et de sortie valides pour chaque type d'action
	ThirdParty GhostInspector		Artefacts d'entrée et de sortie valides pour chaque type d'action
	Custom Jenkins		Artefacts d'entrée et de sortie valides pour chaque type d'action
	ThirdParty Micro Focus StormRunner Load		Artefacts d'entrée et de sortie valides pour chaque type d'action
	ThirdParty Nouvola		Artefacts d'entrée et de sortie valides pour chaque type d'action
	ThirdParty Runscope		Artefacts d'entrée et de sortie valides pour chaque type d'action
Déploiement	Amazon S3		Référence des actions de déploiement d'Amazon S3
	CloudFormation		CloudFormation référence d'action de déploiement
	CodeDeploy		Artefacts d'entrée et de sortie valides pour chaque type d'action

Catégorie d'actions	Fournisseurs d'actions valides	Type de pipeline pris en charge	Référence sur les actions
	Action de déploiement EC2	V2 uniquement	Référence des actions Amazon EC2
	Amazon ECS		Artefacts d'entrée et de sortie valides pour chaque type d'action
	Amazon ECS (Blue/Green) (c'est l'`CodeDeployToECS` action)		Artefacts d'entrée et de sortie valides pour chaque type d'action
	Action Amazon EKS	V2 uniquement	Référence des actions de déploiement d'Amazon Elastic Kubernetes Service EKS
	Haricot élastique		Artefacts d'entrée et de sortie valides pour chaque type d'action
	AWS AppConfig		AWS AppConfig référence d'action de déploiement
	OpsWorks		Artefacts d'entrée et de sortie valides pour chaque type d'action
	Service Catalog		Artefacts d'entrée et de sortie valides pour chaque type d'action
	Amazon Alexa		Artefacts d'entrée et de sortie valides pour chaque type d'action

Catégorie d'actions	Fournisseurs d'actions valides	Type de pipeline pris en charge	Référence sur les actions
	Personnalisé XebiaLabs		Artefacts d'entrée et de sortie valides pour chaque type d'action
Approbation	Manuelle		Artefacts d'entrée et de sortie valides pour chaque type d'action
Invocation	CodePipelineInvoquer une action		AWS CodePipeline invoquer la référence de l'action
	AWS Lambda		AWS Lambda invoquer la référence de l'action
	AWS Step Functions		AWS Step Functions invoquer une référence d'action
	InspectorScan		Amazon Inspector `InspectorScan` invoque une référence d'action
Calcul	Action des commandes		Référence des actions des commandes

Certains types d'actions CodePipeline ne sont disponibles que dans certaines AWS régions. Il est possible qu'un type d'action soit disponible dans une AWS région, mais aucun AWS fournisseur pour ce type d'action n'est disponible.

Pour plus d'informations sur chaque fournisseur d'actions, consultez Intégrations avec les types CodePipeline d'action.

Réglages valides pour le **PollForSourceChanges** paramètre

La valeur par défaut du paramètre `PollForSourceChanges` est déterminée par la méthode utilisée pour créer le pipeline, telle qu'indiquée dans le tableau ci-dessous. Dans de nombreux cas, le paramètre `PollForSourceChanges` prend par défaut la valeur Vrai et doit être désactivé.

Lorsque le paramètre `PollForSourceChanges` prend par défaut la valeur Vrai, procédez comme suit :

- Ajoutez le paramètre `PollForSourceChanges` au fichier JSON ou au modèle CloudFormation .
- Créez des ressources de détection des modifications (règle relative aux CloudWatch événements, le cas échéant).
- Définissez le paramètre `PollForSourceChanges` sur False.

> **Note**
>
> Si vous créez une règle d' CloudWatch événements ou un webhook, vous devez définir le paramètre sur false pour éviter de déclencher le pipeline plusieurs fois.

Le `PollForSourceChanges` paramètre n'est pas utilisé pour les actions relatives aux sources Amazon ECR.

- **`PollForSourceChanges`**paramètres par défaut

Source	Méthode de création	Exemple de sortie de structure JSON de « configuration »
CodeCommit	Le pipeline est créé à l'aide de la console (et les ressources de détection des modifications sont créées par la console). Le paramètre s'affiche dans la sortie de la structure du pipeline et prend la valeur par défaut `false`.	`BranchName": "main", "PollForSourceChanges": "false", "RepositoryName": "my-repo"`
	Le pipeline est créé à l'aide de la CLI ou CloudFormation, et le `PollForSourceChanges` paramètre n'est pas affiché	`BranchName": "main",`

Source	Méthode de création	Exemple de sortie de structure JSON de « configuration »
	dans la sortie JSON, mais il est défini sur `true` .[2]	`"RepositoryName": "my-repo"`
Amazon S3	Le pipeline est créé à l'aide de la console (et les ressources de détection des modifications sont créées par la console). Le paramètre s'affiche dans la sortie de la structure du pipeline et prend la valeur par défaut `false`.	`"S3Bucket": "my-bucket", "S3ObjectKey": "object.zip", "PollForSourceChanges": "false"`
	Le pipeline est créé à l'aide de la CLI ou CloudFormation, et le `PollForSourceChanges` paramètre n'est pas affiché dans la sortie JSON, mais il est défini sur `true` .[2]	`"S3Bucket": "my-bucket", "S3ObjectKey": "object.zip"`
GitHub	Le pipeline est créé à l'aide de la console (et les ressources de détection des modifications sont créées par la console). Le paramètre s'affiche dans la sortie de la structure du pipeline et prend la valeur par défaut `false`.	`"Owner": "MyGitHubAccountName", "Repo": "MyGitHubRepositoryName" "PollForSourceChanges": "false", "Branch": "main" "OAuthToken": "****"`
	Le pipeline est créé à l'aide de la CLI ou CloudFormation, et le `PollForSourceChanges` paramètre n'est pas affiché dans la sortie JSON, mais il est défini sur `true` .[2]	`"Owner": "MyGitHubAccountName", "Repo": "MyGitHubRepositoryName", "Branch": "main", "OAuthToken": "****"`

Source	Méthode de création	Exemple de sortie de structure JSON de « configuration »
	² S'il `PollForSourceChanges` a été ajouté à tout moment à la structure JSON ou au CloudFormation modèle, il s'affiche comme indiqué : `"PollForSourceChanges": "true",` ³ Pour plus d'informations sur les ressources de détection des modifications qui s'appliquent à chaque fournisseur source, consultez la section Méthodes de détection des modifications.	

Artefacts d'entrée et de sortie valides pour chaque type d'action

Selon le type d'action et le fournisseur, vous pouvez avoir le nombre suivant d'artefacts d'entrée et de sortie.

Contraintes relatives au type d'action pour les artefacts

Propriétaire	Type d'action	Fournisseur	Nombre valide d'artefacts d'entrée	Nombre valide d'artefacts de sortie
`AWS`	Source	`S3`	0	1
`AWS`	Source	`CodeCommit`	0	1
`AWS`	Source	`ECR`	0	1
`ThirdParty`	Source	`CodeStarSourceConnection`	0	1
`AWS`	Génération	`CodeBuild`	1 à 5	0 à 5
`AWS`	test	`CodeBuild`	1 à 5	0 à 5

Propriétaire	Type d'action	Fournisseur	Nombre valide d'artefacts d'entrée	Nombre valide d'artefacts de sortie
AWS	test	DeviceFarm	1	0
AWS	Approbation	ThirdParty	0	0
AWS	Déploiement	S3	1	0
AWS	Déploiement	CloudFormation	0 à 10	0 à 1
AWS	Déploiement	CodeDeploy	1	0
AWS	Déploiement	ElasticBeanstalk	1	0
AWS	Déploiement	OpsWorks	1	0
AWS	Déploiement	ECS	1	0
AWS	Déploiement	ServiceCatalog	1	0
AWS	Invocation	Lambda	0 à 5	0 à 5
ThirdParty	Déploiement	AlexaSkillsKit	1 à 2	0
Custom	Génération	Jenkins	0 à 5	0 à 5
Custom	test	Jenkins	0 à 5	0 à 5
Custom	Toutes les catégories prises en charge	Comme indiqué dans l'action personnalisée	0 à 5	0 à 5

Paramètres de configuration valides pour chaque type de fournisseur

Cette section répertorie les paramètres `configuration` valides pour chaque fournisseur d'action.

Chaque action doit être dotée d'une configuration d'action valide, laquelle dépend du type de fournisseur pour cette action. Le tableau suivant répertorie les éléments de configuration d'action requis pour chaque type de fournisseur valide :

Propriétés de configuration d'actions pour les types de fournisseur

Nom du fournisseur	Nom du fournisseur dans le type d'action	Propriétés de configuration	Required/Optional
Amazon S3 (fournisseur d'actions de déploiement)	Pour plus d'informations, y compris des exemples relatifs aux paramètres d'action Amazon S3 Deploy, consultezRéférence des actions de déploiement d'Amazon S3.		
Amazon S3 (fournisseur d'actions source)	Pour plus d'informations, y compris des exemples relatifs aux paramètres d'action source d'Amazon S3, consultezRéférence des actions source d'Amazon S3.		
Amazon ECR	Pour plus d'informations, y compris des exemples relatifs aux paramètres Amazon ECR, consultezRéférence des actions source Amazon ECR.		
CodeCommit	Pour plus d'informations, y compris des exemples relatifs aux CodeCommit paramètres, consultezCodeCommit référence de l'action source.		
CodeStarSourceConnection action pour Bitbucket GitHub (via GitHub l'applica	Pour plus d'informations, y compris des exemples de configuration des actions, consultezParamètres de configuration.		

Nom du fournisseur	Nom du fournisseur dans le type d'action	Propriétés de configuration	Required/Optional
tion), GHES et GitLab			
GitHub (via l'application OAuth)	Pour plus d'informations, y compris des exemples relatifs aux GitHub paramètres, consultezGitHub (via l'application OAuth) référence de l'action source. Il s'agit de l'GitHub action de la version 1.		
AWS CloudFormation	Pour plus d'informations, y compris des exemples relatifs aux AWS CloudFormation paramètres, consultezCloudFormation référence d'action de déploiement.		
CodeBuild	Pour plus de description et d'exemples relatifs aux CodeBuild paramètres, consultezAWS CodeBuild créer et tester une référence d'action.		
CodeDeploy	Pour plus de description et d'exemples relatifs aux CodeDeploy paramètres, consultezAWS CodeDeploy référence d'action de déploiement.		
AWS Device Farm	Pour plus de description et d'exemples relatifs aux AWS Device Farm paramètres, consultezAWS Device Farm référence de l'action de test.		
AWS Elastic Beanstalk	`ElasticBeanstalk`	`ApplicationName`	Obligatoire
		`EnvironmentName`	Obligatoire
AWS Lambda	Pour plus d'informations, y compris des exemples relatifs aux AWS Lambda paramètres, consultezAWS Lambda invoquer la référence de l'action.		
AWS OpsWorks Stacks	`OpsWorks`	`Stack`	Obligatoire
		`Layer`	Facultatif
		`App`	Obligatoire
Amazon ECS	Pour plus de description et d'exemples relatifs aux paramètres Amazon ECS, consultezRéférence des actions de déploiement d'Amazon Elastic Container Service.		

Nom du fournisseur	Nom du fournisseur dans le type d'action	Propriétés de configuration	Required/Optional
Amazon ECS et CodeDeploy (Blue/Green)	Pour plus de description et d'exemples relatifs à Amazon ECS et à CodeDeploy blue/green ses paramètres, consultezAmazon Elastic Container Service et référence des actions de déploiement CodeDeploy bleu-vert.		
Service Catalog	`ServiceCatalog`	`TemplateFilePath`	Obligatoire
		`ProductVersionName`	Obligatoire
		`ProductType`	Obligatoire
		`ProductVersionDescription`	Facultatif
		`ProductId`	Obligatoire
Kit Alexa Skills	`AlexaSkillsKit`	`ClientId`	Obligatoire
		`ClientSecret`	Obligatoire
		`RefreshToken`	Obligatoire
		`SkillId`	Obligatoire
Jenkins	Le nom de l'action que vous avez fournie dans le CodePipeline plug-in pour Jenkins (par exemple,*`MyJenkinsProviderName`*)	`ProjectName`	Obligatoire
Approbation manuelle	`Manual`	`CustomData`	Facultatif
		`ExternalEntityLink`	Facultatif
		`NotificationArn`	Facultatif

L'exemple suivant illustre une configuration valide pour une action de déploiement utilisant un kit Alexa Skills :

```
"configuration": {
  "ClientId": "amzn1.application-oa2-client.aadEXAMPLE",
  "ClientSecret": "****",
  "RefreshToken": "****",
  "SkillId": "amzn1.ask.skill.22649d8f-0451-4b4b-9ed9-bfb6cEXAMPLE"
}
```

L'exemple suivant montre une configuration valide pour une approbation manuelle :

```
"configuration": {
  "CustomData": "Comments on the manual approval",
  "ExternalEntityLink": "http://my-url.com",
  "NotificationArn": "arn:aws:sns:us-west-2:12345EXAMPLE:Notification"
}
```

Référence sur la structure des actions

Cette section est une référence pour la configuration des actions uniquement. Pour obtenir une présentation conceptuelle de la structure du pipeline, veuillez consulter CodePipeline référence de la structure du pipeline.

Chaque fournisseur d'actions CodePipeline utilise un ensemble de champs de configuration obligatoires et facultatifs dans la structure du pipeline. Cette section fournit les informations de référence suivantes par fournisseur d'action :

- Valeurs valides pour les champs `ActionType` inclus dans le bloc d'action de la structure du pipeline, telles que `Owner` et `Provider`.
- Descriptions et autres informations de référence pour les paramètres `Configuration` (obligatoires et facultatifs) inclus dans la section Action de la structure du pipeline.
- Exemples de champs d'action JSON et YAML valides.

Cette section est mise à jour régulièrement avec d'autres fournisseurs d'action. Les informations de référence sont actuellement disponibles pour les fournisseurs d'action suivants :

Rubriques

- Référence des actions Amazon EC2
- Référence des actions source Amazon ECR
- ECRBuildAndPublishcréer une référence d'action
- Amazon Elastic Container Service et référence des actions de déploiement CodeDeploy bleu-vert
- Référence des actions de déploiement d'Amazon Elastic Container Service
- Référence des actions de déploiement d'Amazon Elastic Kubernetes Service EKS
- AWS Lambda référence des actions de déploiement
- Référence des actions de déploiement d'Amazon S3
- Référence des actions source d'Amazon S3
- AWS AppConfig référence d'action de déploiement
- CloudFormation référence d'action de déploiement
- CloudFormation StackSets référence des actions de déploiement
- AWS CodeBuild créer et tester une référence d'action

- AWS CodePipeline invoquer la référence de l'action
- CodeCommit référence de l'action source
- AWS CodeDeploy référence d'action de déploiement
- CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées
- Référence des actions des commandes
- AWS Device Farm référence de l'action de test
- Référence des actions de déploiement d'Elastic Beanstalk
- Amazon Inspector InspectorScan invoque une référence d'action
- AWS Lambda invoquer la référence de l'action
- AWS OpsWorks référence des actions de déploiement
- AWS Service Catalog référence des actions de déploiement
- AWS Step Functions invoquer une référence d'action

Référence des actions Amazon EC2

Vous utilisez une EC2 action Amazon EC2 pour déployer le code d'application sur votre parc de déploiement. Votre parc de déploiement peut être composé d'instances Linux Amazon EC2 ou de SSM-managed nœuds Linux. L'agent SSM doit être installé sur vos instances.

> **Note**
>
> Cette action ne prend en charge que les types d'instances Linux. La taille maximale de la flotte prise en charge est de 500 instances.

L'action choisira un certain nombre d'instances en fonction d'un maximum spécifié. Les instances ayant échoué des instances précédentes seront sélectionnées en premier. L'action ignorera le déploiement sur certaines instances si l'instance a déjà reçu le déploiement du même artefact d'entrée, par exemple si l'action a échoué précédemment.

> **Note**
>
> Cette action n'est prise en charge que pour les pipelines de type V2.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Autorisations relatives à la politique de rôle de service pour l'action de déploiement d'EC2
- Référence du fichier de spécifications du déploiement
- Déclaration d'action
- Déclaration d'action avec exemple de spécification Deploy
- Consultez aussi

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `EC2`
- Version : `1`

Paramètres de configuration

InstanceTagKey

Obligatoire : oui

La clé de balise des instances que vous avez créées dans Amazon EC2, telles que`Name`.

InstanceTagValue

Obligatoire : non

La valeur de balise des instances que vous avez créées dans Amazon EC2, telles que`my-instances`.

Lorsque cette valeur n'est pas spécifiée, toutes les instances avec InstanceTagKey seront mises en correspondance.

InstanceType

Obligatoire : oui

Type d'instances ou de nœuds SSM créés dans Amazon EC2. Les valeurs valides sont `EC2` et `SSM_MANAGED_NODE`.

Vous devez déjà avoir créé, balisé et installé l'agent SSM sur toutes les instances.

> **Note**
>
> Lorsque vous créez l'instance, vous créez ou utilisez un rôle d'instance EC2 existant. Pour éviter les `Access Denied` erreurs, vous devez ajouter des autorisations de compartiment S3 au rôle d'instance afin de donner à l'instance des autorisations d'accès au compartiment d' CodePipeline artefacts. Créez un rôle par défaut ou mettez à jour votre rôle existant avec l'`s3:GetObject`autorisation limitée au compartiment d'artefacts pour la région de votre pipeline.

TargetDirectory

Obligatoire : Oui (si le script est spécifié)

Le répertoire à utiliser sur votre instance Amazon EC2 pour exécuter des scripts.

DeploySpec

Obligatoire : Oui (si la spécification de déploiement est spécifiée)

Fichier à utiliser pour configurer les événements de déploiement, d'installation et de cycle de vie. Pour obtenir des descriptions et des informations sur les champs des spécifications de déploiement, consultezRéférence du fichier de spécifications du déploiement. Pour afficher une configuration d'action avec le fichier de spécifications de déploiement spécifié, consultez l'exemple dansDéclaration d'action avec exemple de spécification Deploy.

MaxBatch

Obligatoire : non

Le nombre maximum d'instances autorisées à être déployées en parallèle.

MaxError

Obligatoire : non

Le nombre maximum d'erreurs d'instance autorisées pendant le déploiement.

TargetGroupNameList

Obligatoire : non

La liste des noms des groupes cibles pour le déploiement. Vous devez déjà avoir créé les groupes cibles.

Les groupes cibles fournissent un ensemble d'instances permettant de traiter des demandes spécifiques. Si le groupe cible est spécifié, les instances seront supprimées du groupe cible avant le déploiement et rajoutées au groupe cible après le déploiement.

PreScript

Obligatoire : non

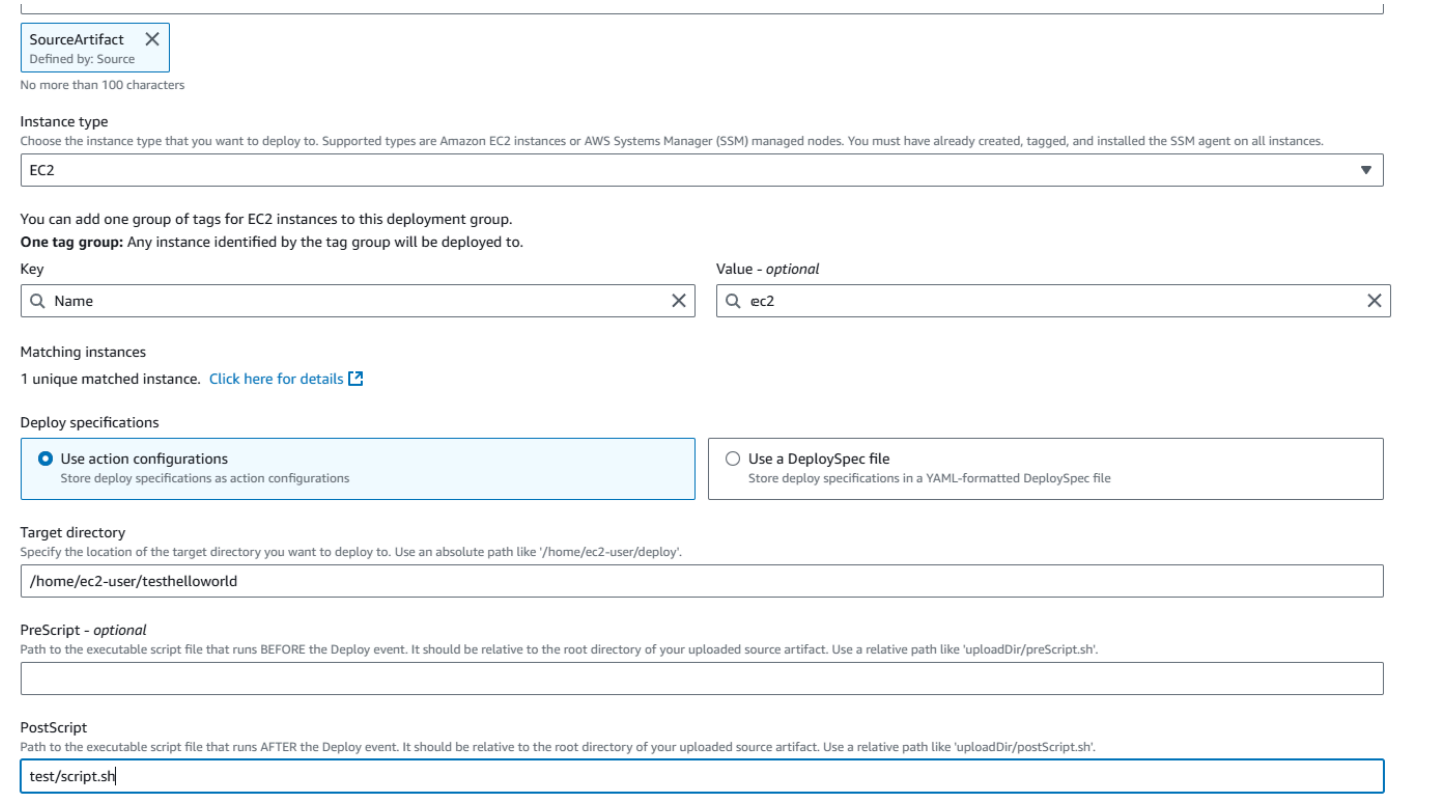
Le script à exécuter avant la phase de déploiement de l'action.

PostScript

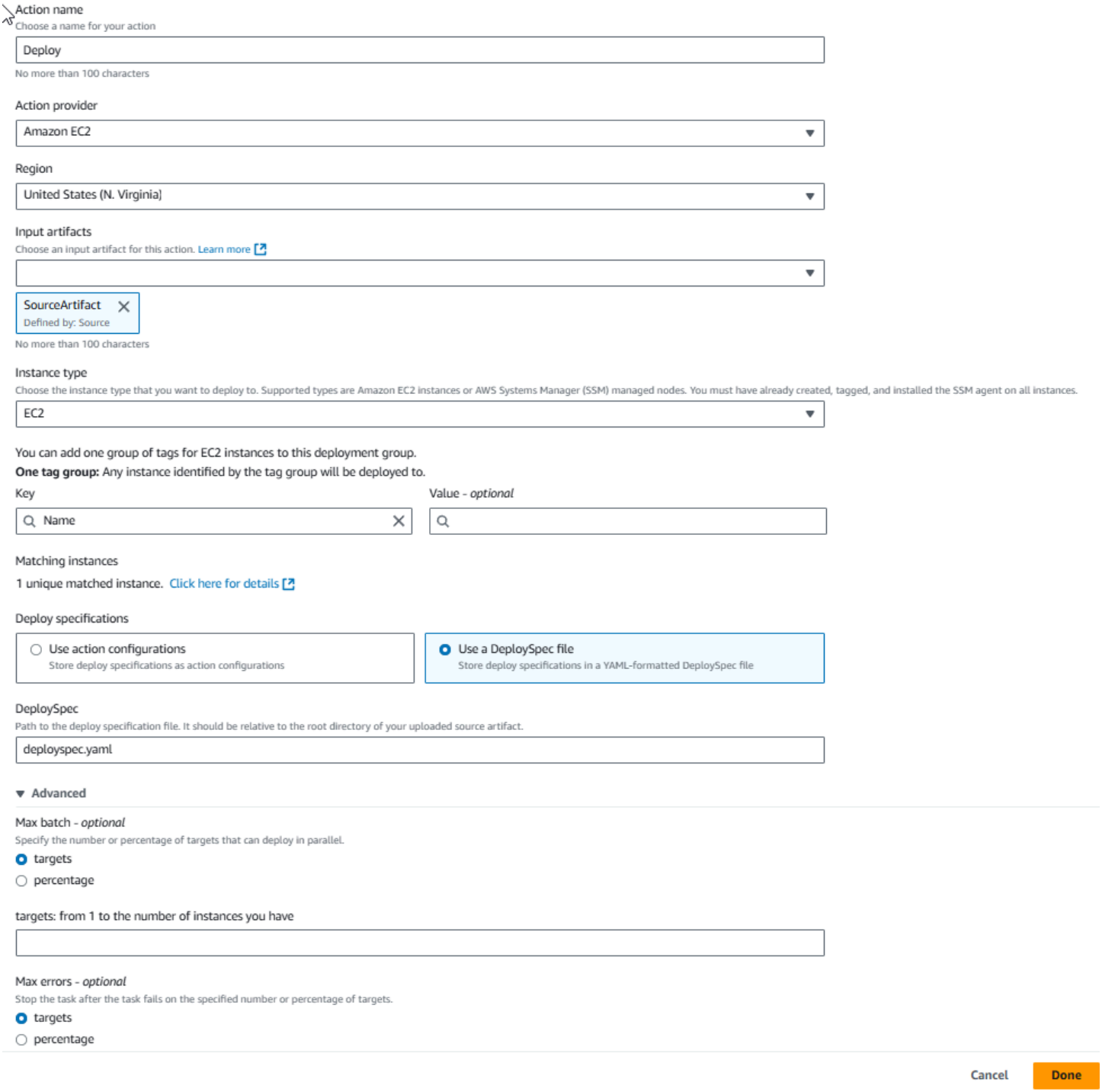
Obligatoire : oui

Le script à exécuter après la phase de déploiement de l'action.

L'image suivante montre un exemple de la page Modifier pour l'action dans laquelle l'option Utiliser les configurations d'action est sélectionnée.



L'image suivante montre un exemple de la page Modifier pour l'action dans laquelle l'option Utiliser un DeploySpec fichier est sélectionnée.



Artefacts d'entrée

- Nombre d'objets : 1
- Description : fichiers fournis, le cas échéant, pour prendre en charge les actions de script pendant le déploiement.

Artefacts de sortie

- Nombre d'objets : `0`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Autorisations relatives à la politique de rôle de service pour l'action de déploiement d'EC2

Lors de l' CodePipeline exécution de l'action, le rôle de CodePipeline service nécessite les autorisations suivantes, définies de manière appropriée pour un accès avec le moins de privilèges.

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"
            ]
        }
    ]
```

```
}
```

Consignez les groupes pour votre pipeline dans CloudWatch les journaux

Lorsque l'action est CodePipeline exécutée, CodePipeline crée un groupe de journaux en utilisant le nom du pipeline comme suit. Cela vous permet de limiter les autorisations d'enregistrement des ressources à l'aide du nom du pipeline.

```
/aws/codepipeline/MyPipelineName
```

Les autorisations de journalisation suivantes sont incluses dans les mises à jour ci-dessus pour le rôle de service.

- journaux : CreateLogGroup
- journaux : CreateLogStream
- journaux : PutLogEvents

Pour afficher les journaux dans la console à l'aide de la page de dialogue des détails des actions, l'autorisation d'affichage des journaux doit être ajoutée au rôle de console. Pour plus d'informations, consultez l'exemple de politique d'autorisations de console dansAutorisations requises pour consulter les journaux de calcul dans la console.

Autorisations relatives à la politique de rôle de service pour les CloudWatch journaux

Lorsque l'action est CodePipeline exécutée, CodePipeline crée un groupe de journaux en utilisant le nom du pipeline comme suit. Cela vous permet de limiter les autorisations d'enregistrement des ressources à l'aide du nom du pipeline.

```
/aws/codepipeline/MyPipelineName
```

Pour afficher les journaux dans la console à l'aide de la page de dialogue des détails des actions, l'autorisation d'affichage des journaux doit être ajoutée au rôle de console. Pour plus d'informations, consultez l'exemple de politique d'autorisations de console dansAutorisations requises pour consulter les journaux de calcul dans la console.

Référence du fichier de spécifications du déploiement

Lorsque l'action est CodePipeline exécutée, vous pouvez spécifier un fichier de spécifications pour configurer le déploiement sur vos instances. Le fichier de spécifications de déploiement spécifie les éléments à installer et les hooks d'événements du cycle de vie à exécuter en réponse aux événements du cycle de vie du déploiement. Le fichier de spécifications de déploiement est toujours YAML-formatted. Le fichier de spécifications de déploiement est utilisé pour :

- mapper les fichiers sources figurant dans votre révision d'application vers leurs destinations sur l'instance ;
- spécifier des autorisations personnalisées pour les fichiers déployés ;
- spécifier les scripts à exécuter sur chaque instance à diverses étapes du processus de déploiement.

Le fichier de spécifications de déploiement prend en charge les paramètres de configuration de déploiement spécifiques pris CodeDeploy en charge par le AppSpec fichier. Vous pouvez utiliser directement votre AppSpec fichier existant, et tous les paramètres non pris en charge seront ignorés. Pour plus d'informations sur le AppSpec fichier dans CodeDeploy, reportez-vous à la référence du fichier de spécification de l'application dans le Guide de CodeDeploy l'utilisateur.

Les paramètres de déploiement des fichiers sont spécifiés comme suit.

- `files`- Le fichier de spécifications de déploiement désigne le `source:` et `destination:` pour les fichiers de déploiement.
- `scripts`- Les événements scénarisés pour le déploiement. Deux événements sont pris en charge : `BeforeDeploy` et`AfterDeploy`.
- `hooks`- Le cycle de vie dépend de l'événement. Les crochets suivants sont pris en charge : `ApplicationStop` `BeforeInstall`,`AfterInstall`,`ApplicationStart`, et`ValidateService`.

> **Note**
>
> Le paramètre hooks est disponible pour des raisons de AppSpec compatibilité avec CodeDeploy et n'est disponible que dans la version 0.0 (AppSpec format). Pour ce format, CodePipeline effectuera de son mieux pour cartographier les événements.

Un espacement YAML correct doit être utilisé dans le fichier de spécifications ; sinon, une erreur est générée si les emplacements et le nombre d'espaces dans un fichier de spécifications de déploiement ne sont pas corrects. Pour plus d'informations sur l'espacement, consultez la spécification YAML.

Vous trouverez ci-dessous un exemple de fichier de spécifications de déploiement.

```
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
```

Pour afficher une configuration d'action avec le fichier de spécifications de déploiement spécifié, consultez l'exemple dansDéclaration d'action avec exemple de spécification Deploy.

Déclaration d'action

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"
        }
    ]
},
```

Déclaration d'action avec exemple de spécification Deploy

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"
        }
    ]
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Tutoriel : Déploiement sur des instances Amazon EC2 avec CodePipeline— Ce didacticiel explique la création d'instances EC2 dans lesquelles vous allez déployer un fichier de script, ainsi que la création du pipeline à l'aide de l'action EC2.
- L'action de déploiement EC2 échoue avec un message d'erreur `Aucun fichier de ce type`— Cette rubrique décrit le dépannage des erreurs de fichier introuvable liées à l'action EC2.

Référence des actions source Amazon ECR

Déclenche le pipeline lorsqu'une nouvelle image est envoyée vers le référentiel Amazon ECR. Cette action fournit un fichier de définitions d'image faisant référence à l'URI de l'image qui a été transmise à Amazon ECR. Cette action source est souvent utilisée conjointement avec une autre action source, par exemple CodeCommit, pour attribuer un emplacement source à tous les autres artefacts source. Pour de plus amples informations, veuillez consulter Tutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR.

Lorsque vous utilisez la console pour créer ou modifier votre pipeline, CodePipeline crée une EventBridge règle qui démarre votre pipeline lorsqu'une modification se produit dans le référentiel.

> **Note**
>
> Pour Amazon ECR, Amazon S3 ou les CodeCommit sources, vous pouvez également créer un remplacement de source à l'aide d'une entrée de transformation en entrée

EventBridge pour utiliser l'`revisionValue`entrée de votre événement de pipeline, où le `revisionValue` est dérivé de la variable d'événement source pour votre clé d'objet, votre commit ou votre ID d'image. Pour plus d'informations, consultez l'étape facultative pour la saisie de la transformation d'entrée incluse dans les procédures sous Actions et ressources relatives aux sources Amazon ECR EventBridgeConnexion aux actions de source Amazon S3 avec une source activée pour les événements, ouCodeCommit actions à la source et EventBridge.

Vous devez déjà avoir créé un référentiel Amazon ECR et envoyé une image avant de connecter le pipeline via une action Amazon ECR.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables de sortie
- Autorisations relatives aux rôles de service : action Amazon ECR
- Déclaration d'action (exemple Amazon ECR)
- Consultez aussi

Type d'action

- Catégorie : `Source`
- Propriétaire : `AWS`
- Fournisseur : `ECR`
- Version : `1`

Paramètres de configuration

RepositoryName

Obligatoire : oui

Le nom du référentiel Amazon ECR dans lequel l'image a été envoyée.

ImageTag

Obligatoire : non

Balise utilisée pour l'image.

> **Note**
>
> Si aucune valeur n'est spécifiée pour `ImageTag`, la valeur par défaut est `latest`.

Artefacts d'entrée

- Nombre d'objets : `0`
- Description : Les artefacts d'entrée ne s'appliquent pas à ce type d'action.

Artefacts de sortie

- Nombre d'objets : `1`
- Description : Cette action produit un artefact qui contient un fichier `imageDetail.json` dans lequel figure l'URI de l'image ayant déclenché l'exécution du pipeline. Pour de plus amples informations concernant le fichier `imageDetail.json`, veuillez consulter Detail.json fichier image pour les actions de blue/green déploiement d'Amazon ECS.

Variables de sortie

Lorsque cette action est configurée, elle produit des variables qui peuvent être référencées par la configuration d'action d'une action en aval dans le pipeline. Cette action produit des variables qui peuvent être visualisées en tant que variables de sortie, même si l'action n'a pas d'espace de noms. Vous configurez une action avec un espace de noms pour rendre ces variables disponibles pour la configuration des actions en aval.

Pour de plus amples informations, veuillez consulter Référence des variables.

RegistryId

L'ID de AWS compte associé au registre qui contient le référentiel.

RepositoryName

Le nom du référentiel Amazon ECR dans lequel l'image a été envoyée.

ImageTag

Balise utilisée pour l'image.

> **Note**
>
> La variable `ImageTag` de sortie n'est pas générée lorsque la révision de la source est remplacée.

ImageDigest

Hachage `sha256` du manifeste de l'image.

ImageURI

URI de l'image.

Autorisations relatives aux rôles de service : action Amazon ECR

Pour le support Amazon ECR, ajoutez ce qui suit à votre déclaration de politique :

```
{
    "Effect": "Allow",
    "Action": [
        "ecr:DescribeImages"
    ],
    "Resource": "resource_ARN"
},
```

Pour plus d'informations sur cette action, consultezRéférence des actions source Amazon ECR.

Déclaration d'action (exemple 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"
        }
    ]
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Tutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR— Ce didacticiel fournit un exemple de AppSpec fichier et un exemple d' CodeDeploy application et de groupe de déploiement pour créer un pipeline avec une CodeCommit source Amazon ECR qui se déploie sur des instances Amazon ECS.

ECRBuildAndPublishcréer une référence d'action

Cette action de génération vous permet d'automatiser la création et la diffusion d'une nouvelle image lorsqu'une modification se produit dans votre source. Cette action est créée en fonction d'un emplacement Dockerfile spécifié et pousse l'image. Cette action de génération est différente de l'action source Amazon ECR dans CodePipeline, qui déclenche le pipeline lorsqu'une modification survient dans votre référentiel source Amazon ECR. Pour plus d'informations sur cette action, consultezRéférence des actions source Amazon ECR.

Il ne s'agit pas d'une action source qui déclenchera le pipeline. Cette action crée une image et l'envoie vers votre référentiel d'images Amazon ECR.

Vous devez déjà avoir créé un référentiel Amazon ECR et avoir ajouté un Dockerfile à votre référentiel de code source, par exemple GitHub, avant d'ajouter l'action à votre pipeline.

> **Important**
>
> Cette action utilise le CodeBuild calcul CodePipeline géré pour exécuter des commandes dans un environnement de génération. L'exécution de l'action des commandes entraînera des frais distincts dans AWS CodeBuild.

> **Note**
>
> Cette action n'est disponible que pour les pipelines de type V2.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables de sortie
- Autorisations relatives aux rôles de service : ECRBuildAndPublish action
- Déclaration d'action
- Consultez aussi

Type d'action

- Catégorie : `Build`
- Propriétaire : `AWS`
- Fournisseur : `ECRBuildAndPublish`
- Version : `1`

Paramètres de configuration

ECRRepositoryName

Obligatoire : oui

Le nom du référentiel Amazon ECR dans lequel l'image est envoyée.

DockerFilePath

Obligatoire : non

Emplacement du Dockerfile utilisé pour créer l'image. Vous pouvez éventuellement fournir un autre emplacement Dockerfile s'il ne se trouve pas au niveau de la racine.

> **Note**
>
> Si aucune valeur pour n'`DockerFilePath`est spécifiée, la valeur par défaut est le niveau racine du référentiel source.

ImageTags

Obligatoire : non

Les balises utilisées pour l'image. Vous pouvez saisir plusieurs balises sous la forme d'une liste de chaînes délimitée par des virgules.

> **Note**
>
> Si aucune valeur n'est spécifiée pour `ImageTags`, la valeur par défaut est `latest`.

RegistryType

Obligatoire : non

Spécifie si le référentiel est public ou privé. Les valeurs valides sont `private | public`.

> **Note**
>
> Si aucune valeur n'est spécifiée pour `RegistryType`, la valeur par défaut est `private`.

Artefacts d'entrée

- Nombre d'objets : `1`
- Description : artefact produit par l'action source qui contient le Dockerfile nécessaire à la création de l'image.

Artefacts de sortie

- Nombre d'objets : `0`

Variables de sortie

Lorsque cette action est configurée, elle produit des variables qui peuvent être référencées par la configuration d'action d'une action en aval dans le pipeline. Cette action produit des variables qui peuvent être visualisées en tant que variables de sortie, même si l'action n'a pas d'espace de noms.

Vous configurez une action avec un espace de noms pour rendre ces variables disponibles pour la configuration des actions en aval.

Pour de plus amples informations, veuillez consulter Référence des variables.

ECRImageDigestId

Hachage `sha256` du manifeste de l'image.

ECRRepositoryName

Nom du référentiel Amazon ECR dans lequel l'image a été envoyée.

Autorisations relatives aux rôles de service : **`ECRBuildAndPublish`** action

Pour l'aide à l'`ECRBuildAndPublish`action, ajoutez ce qui suit à votre déclaration de politique :

```
{
    "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": "*"
        }
    ]
}
```

En outre, si ce n'est pas déjà fait pour l'`Commands`action, ajoutez les autorisations suivantes à votre rôle de service afin d'afficher CloudWatch les journaux.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Réduisez les autorisations au niveau des ressources du pipeline en utilisant des autorisations basées sur les ressources dans la déclaration de politique des rôles de service.

Pour plus d'informations sur cette action, consultezECRBuildAndPublishcréer une référence d'action.

Déclaration d'action

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"
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Tutoriel : Créez et transférez une image Docker vers Amazon ECR avec CodePipeline (type V2)— Ce didacticiel fournit un exemple de Dockerfile et des instructions pour créer un pipeline qui transmet votre image vers ECR lors d'une modification de votre référentiel source, puis la déploie sur Amazon ECS.

Amazon Elastic Container Service et référence des actions de déploiement CodeDeploy bleu-vert

Vous pouvez configurer un pipeline AWS CodePipeline qui déploie des applications de conteneur à l'aide d'un blue/green déploiement. Lors d'un blue/green déploiement, vous pouvez lancer une nouvelle version de votre application parallèlement à l'ancienne version, et vous pouvez tester la nouvelle version avant de rediriger le trafic vers celle-ci. Vous pouvez également surveiller le processus de déploiement et revenir rapidement en arrière en cas de problème.

Le pipeline terminé détecte les modifications apportées à vos images ou à votre fichier de définition de tâches et permet CodeDeploy d'acheminer et de déployer le trafic vers un cluster Amazon ECS et un équilibreur de charge. CodeDeploy crée un nouvel écouteur sur votre équilibreur de charge qui peut cibler votre nouvelle tâche via un port spécial. Vous pouvez également configurer le pipeline pour utiliser un emplacement source, tel qu'un CodeCommit référentiel, dans lequel votre définition de tâche Amazon ECS est stockée.

Avant de créer votre pipeline, vous devez déjà avoir créé les ressources Amazon ECS, les CodeDeploy ressources, l'équilibreur de charge et le groupe cible. Vous devez déjà avoir balisé et stocké l'image dans votre référentiel d'images, et avoir téléchargé la définition de tâche et le AppSpec fichier dans votre référentiel de fichiers.

> **Note**
>
> Cette rubrique décrit l'action de CodeDeploy blue/green déploiement d'Amazon ECS pour CodePipeline. Pour obtenir des informations de référence sur les actions de déploiement standard d'Amazon ECS dans CodePipeline, consultezRéférence des actions de déploiement d'Amazon Elastic Container Service.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Autorisations relatives aux rôles de service : CodeDeployToECS action
- Déclaration d'action
- Consultez aussi

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `CodeDeployToECS`
- Version : `1`

Paramètres de configuration

ApplicationName

Obligatoire : oui

Le nom de l'application dans CodeDeploy. Avant de créer votre pipeline, vous devez avoir déjà créé l'application dans CodeDeploy.

DeploymentGroupName

Obligatoire : oui

Le groupe de déploiement spécifié pour les ensembles de tâches Amazon ECS que vous avez créés pour votre CodeDeploy application. Avant de créer votre pipeline, vous devez déjà avoir créé le groupe de déploiement dans CodeDeploy.

TaskDefinitionTemplateArtifact

Obligatoire : oui

Nom de l'artefact en entrée qui fournit le fichier de définition de tâche à l'action de déploiement. Il s'agit généralement du nom de l'artefact de sortie de l'action source. Lorsque vous utilisez la console, le nom par défaut de l'artefact de sortie de l'action source est`SourceArtifact`.

AppSpecTemplateArtifact

Obligatoire : oui

Nom de l'artefact en entrée qui fournit le AppSpec fichier à l'action de déploiement. Cette valeur est mise à jour lorsque votre pipeline est exécuté. Il s'agit généralement du nom de l'artefact de sortie de l'action source. Lorsque vous utilisez la console, le nom par défaut de l'artefact de sortie de l'action source est`SourceArtifact`. Pour `TaskDefinition` le AppSpec fichier, vous pouvez conserver le texte de l'`<TASK_DEFINITION>`espace réservé comme indiqué ici.

AppSpecTemplatePath

Obligatoire : non

Nom du AppSpec fichier stocké dans l'emplacement du fichier source du pipeline, tel que le CodeCommit référentiel de votre pipeline. Le nom de fichier par défaut est `appspec.yaml`. Si votre AppSpec fichier porte le même nom et est stocké à la racine dans votre référentiel de fichiers, il n'est pas nécessaire de fournir le nom du fichier. Si le chemin n'est pas le chemin par défaut, entrez le chemin et le nom du fichier.

TaskDefinitionTemplatePath

Obligatoire : non

Nom de fichier de la définition de tâche stockée dans l'emplacement source du fichier de pipeline, tel que le CodeCommit référentiel de votre pipeline. Le nom de fichier par défaut est `taskdef.json`. Si votre fichier de définition de tâches porte le même nom et est stocké au niveau racine dans votre référentiel de fichiers, il n'est pas nécessaire de fournir le nom de fichier. Si le chemin n'est pas le chemin par défaut, entrez le chemin et le nom du fichier.

Image <Number>ArtifactName

Obligatoire : non

Nom de l'artefact en entrée qui fournit l'image à l'action de déploiement. Il s'agit généralement de l'artefact de sortie du référentiel d'images, tel que la sortie de l'action source Amazon ECR.

Les valeurs disponibles `<Number>` sont comprises entre 1 et 4.

Image <Number>ContainerName

Obligatoire : non

Le nom de l'image disponible dans le référentiel d'images, tel que le référentiel source Amazon ECR.

Les valeurs disponibles `<Number>` sont comprises entre 1 et 4.

Artefacts d'entrée

- Nombre d'artefacts : `1 to 5`
- Description : L'`CodeDeployToECS`action recherche d'abord le fichier de définition de tâche et le AppSpec fichier dans le référentiel de fichiers source, recherche ensuite l'image dans le référentiel d'images, puis génère dynamiquement une nouvelle révision de la définition de tâche et exécute enfin les AppSpec commandes pour déployer l'ensemble de tâches et le conteneur sur le cluster.

  L'`CodeDeployToECS`action recherche un `imageDetail.json` fichier qui associe l'URI de l'image à l'image. Lorsque vous validez une modification dans votre référentiel d'images Amazon ECR, l'action source du pipeline ECR crée un `imageDetail.json` fichier pour cette validation. Vous pouvez également ajouter manuellement un `imageDetail.json` fichier pour un pipeline dont l'action n'est pas automatisée. Pour de plus amples informations concernant le fichier `imageDetail.json`, veuillez consulter Detail.json fichier image pour les actions de blue/green déploiement d'Amazon ECS.

  L'`CodeDeployToECS`action génère dynamiquement une nouvelle révision de la définition de la tâche. Au cours de cette phase, cette action remplace les espaces réservés du fichier de définition de tâche par un URI d'image extrait des Detail.json fichiers image. Par exemple, si vous définissez IMAGE1_NAME comme Image1ContainerName paramètre, vous devez spécifier l'espace réservé <IMAGE1_NAME>comme valeur du champ image dans votre fichier de définition de tâche. Dans ce cas, l' CodeDeployToECS action remplace l'espace réservé par l'<IMAGE1_NAME>URI de l'image réelle extraite de l'image Detail.json dans l'artefact que vous spécifiez comme étant. Image1ArtifactName

Pour les mises à jour des définitions de tâches, le CodeDeploy `AppSpec.yaml` fichier contient la `TaskDefinition` propriété.

```
TaskDefinition: <TASK_DEFINITION>
```

Cette propriété sera mise à jour par l'`CodeDeployToECS`action une fois la nouvelle définition de tâche créée.

<TASK_DEFINITION>Pour la valeur du `TaskDefinition` champ, le texte de l'espace réservé doit être. L'`CodeDeployToECS`action remplace cet espace réservé par l'ARN réel de la définition de tâche générée dynamiquement.

Artefacts de sortie

- Nombre d'artefacts : `0`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Autorisations relatives aux rôles de service : **`CodeDeployToECS`** action

Pour l'`CodeDeployToECS`action (blue/green déploiements), les autorisations minimales requises pour créer des pipelines avec une action de blue/green déploiement CodeDeploy vers Amazon ECS sont les suivantes.

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"
                    ]
                }
            }
        }
    ]
}
```

Vous pouvez choisir d'utiliser l'autorisation de balisage dans Amazon ECS. En vous inscrivant, vous devez accorder les autorisations suivantes :`ecs:TagResource`. Pour plus d'informations sur la manière de s'inscrire et de déterminer si l'autorisation est requise et si l'autorisation des balises est appliquée, consultez la chronologie des autorisations de balisage dans le guide du développeur Amazon Elastic Container Service.

Vous devez également ajouter les `iam:PassRole` autorisations permettant d'utiliser les rôles IAM pour les tâches. Pour plus d'informations, consultez le rôle IAM d'exécution des tâches Amazon ECS et les rôles IAM pour les tâches.

Vous pouvez également `ecs-tasks.amazonaws.com` compléter la liste des services concernés par `iam:PassedToService` cette condition, comme indiqué dans l'exemple ci-dessus.

Déclaration d'action

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"
        }
    ]
}
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Tutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR— Ce didacticiel explique comment créer CodeDeploy les ressources Amazon ECS dont vous avez besoin pour un blue/green déploiement. Le didacticiel explique comment envoyer une image Docker vers Amazon ECR et créer une définition de tâche Amazon ECS répertoriant le nom de votre image Docker, le nom du conteneur, le nom du service Amazon ECS et la configuration de l'équilibreur de charge. Le didacticiel vous explique ensuite comment créer le AppSpec fichier et le pipeline pour votre déploiement.

  > **Note**
  >
  > Cette rubrique et ce didacticiel décrivent l' CodeDeploy/ECS blue/green action à effectuer pour CodePipeline. Pour plus d'informations sur les actions standard d'ECS dans CodePipeline, consultez Tutoriel : Déploiement continu avec CodePipeline.

- AWS CodeDeploy Guide de l'utilisateur — Pour plus d'informations sur l'utilisation de l'équilibreur de charge, de l'écouteur de production, des groupes cibles et de votre application Amazon ECS dans un blue/green déploiement, consultez Tutoriel : Déployer un service Amazon ECS. Ces informations de référence du Guide de l'AWS CodeDeploy utilisateur fournissent une vue d'ensemble blue/green des déploiements avec Amazon ECS et AWS CodeDeploy.
- Guide du développeur Amazon Elastic Container Service — Pour plus d'informations sur l'utilisation des images et des conteneurs Docker, des services et clusters ECS et des ensembles de tâches ECS, consultez Qu'est-ce qu'Amazon ECS ?

Référence des actions de déploiement d'Amazon Elastic Container Service

Vous pouvez utiliser une action Amazon ECS pour déployer un service et un ensemble de tâches Amazon ECS. Un service Amazon ECS est une application de conteneur déployée sur un cluster Amazon ECS. Un cluster Amazon ECS est un ensemble d'instances qui hébergent votre application de conteneur dans le cloud. Le déploiement nécessite une définition de tâche que vous créez dans Amazon ECS et un fichier de définitions d'image CodePipeline utilisé pour déployer l'image.

> **Important**
>
> L'action de déploiement standard Amazon ECS pour CodePipeline crée sa propre révision de la définition de tâche en fonction de la révision utilisée par le service Amazon ECS. Si vous créez de nouvelles révisions pour la définition de tâche sans mettre à jour le service Amazon ECS, l'action de déploiement ignorera ces révisions.

Avant de créer votre pipeline, vous devez déjà avoir créé les ressources Amazon ECS, balisé et stocké l'image dans votre référentiel d'images, et téléchargé le fichier buildspec dans votre référentiel de fichiers.

> **Note**
>
> Cette rubrique de référence décrit l'action de déploiement standard d'Amazon ECS pour CodePipeline. Pour obtenir des informations de référence sur Amazon ECS relatives aux actions de CodeDeploy blue/green déploiement dans CodePipeline, consultezAmazon Elastic Container Service et référence des actions de déploiement CodeDeploy bleu-vert.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Autorisations relatives aux rôles de service : action standard Amazon ECS
- Déclaration d'action

- Consultez aussi

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `ECS`
- Version : `1`

Paramètres de configuration

ClusterName

Obligatoire : oui

Le cluster Amazon ECS dans Amazon ECS.

ServiceName

Obligatoire : oui

Le service Amazon ECS que vous avez créé dans Amazon ECS.

FileName

Obligatoire : non

Le nom de votre fichier de définitions d'image, le fichier JSON qui décrit le nom du conteneur de votre service, ainsi que l'image et la balise. Vous utilisez ce fichier pour les déploiements standard ECS. Pour plus d'informations, consultez Artefacts d'entrée et fichier imagedefinitions.json pour les actions de déploiement standard d'Amazon ECS.

DeploymentTimeout

Obligatoire : non

Délai d'expiration de l'action de déploiement d'Amazon ECS en minutes. Le délai d'expiration est configurable dans la limite du délai d'expiration par défaut maximal pour cette action. Par exemple :

```
"DeploymentTimeout": "15"
```

Artefacts d'entrée

- Nombre d'objets : `1`
- Description : L'action recherche un `imagedefinitions.json` fichier dans le référentiel de fichiers source du pipeline. Un document de définitions d'images est un fichier JSON qui décrit le nom de votre conteneur Amazon ECS, ainsi que l'image et la balise. CodePipeline utilise le fichier pour récupérer l'image depuis votre référentiel d'images tel qu'Amazon ECR. Vous pouvez ajouter manuellement un `imagedefinitions.json` fichier pour un pipeline dont l'action n'est pas automatisée. Pour de plus amples informations concernant le fichier `imagedefinitions.json`, veuillez consulter fichier imagedefinitions.json pour les actions de déploiement standard d'Amazon ECS.

  L'action nécessite une image existante qui a déjà été transférée vers votre référentiel d'images. Comme le mappage des images est fourni par le `imagedefinitions.json` fichier, l'action ne nécessite pas que la source Amazon ECR soit incluse en tant qu'action source dans le pipeline.

Artefacts de sortie

- Nombre d'objets : `0`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Autorisations relatives aux rôles de service : action standard Amazon ECS

Pour Amazon ECS, les autorisations minimales requises pour créer des pipelines avec une action de déploiement Amazon ECS sont les suivantes.

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"
                ]
            }
        }
    }
]
}
```

Vous pouvez choisir d'utiliser l'autorisation de balisage dans Amazon ECS. En vous inscrivant, vous devez accorder les autorisations suivantes :`ecs:TagResource`. Pour plus d'informations sur la manière de s'inscrire et de déterminer si l'autorisation est requise et si l'autorisation des balises est appliquée, consultez la chronologie des autorisations de balisage dans le guide du développeur Amazon Elastic Container Service.

Vous devez ajouter les `iam:PassRole` autorisations nécessaires pour utiliser les rôles IAM pour les tâches. Pour plus d'informations, consultez le rôle IAM d'exécution des tâches Amazon ECS et les rôles IAM pour les tâches. Utilisez le texte de politique suivant.

Déclaration d'action

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"
        }
    ]
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Consultez Tutoriel : Créez et transférez une image Docker vers Amazon ECR avec CodePipeline (type V2) un didacticiel qui explique comment utiliser l' ECRBuildandPublish action pour envoyer une image, puis utiliser l'action standard ECS pour déployer sur Amazon ECS.
- Tutoriel : Déploiement continu avec CodePipeline — Ce didacticiel vous montre comment créer un Dockerfile que vous stockez dans un référentiel de fichiers source tel que. CodeCommit Le didacticiel vous montre ensuite comment intégrer un fichier CodeBuild buildspec qui génère et transmet votre image Docker vers Amazon ECR et crée votre fichier imagedefinitions.json. Enfin, vous créez un service Amazon ECS et une définition de tâche, puis vous créez votre pipeline à l'aide d'une action de déploiement Amazon ECS.

> **Note**
>
> Cette rubrique et ce didacticiel décrivent l'action de déploiement standard d'Amazon ECS pour CodePipeline. Pour plus d'informations sur les actions de CodeDeploy blue/green déploiement d'Amazon ECS CodePipeline, consultezTutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR.

- Guide du développeur Amazon Elastic Container Service — Pour plus d'informations sur l'utilisation des images et des conteneurs Docker, des services et clusters Amazon ECS et des ensembles de tâches Amazon ECS, consultez Qu'est-ce qu'Amazon ECS ?

Référence des actions de déploiement d'Amazon Elastic Kubernetes Service **EKS**

Vous pouvez utiliser cette `EKSDeploy` action pour déployer un service Amazon EKS. Le déploiement nécessite un fichier manifeste Kubernetes CodePipeline utilisé pour déployer l'image.

Avant de créer votre pipeline, vous devez déjà avoir créé les ressources Amazon EKS et avoir stocké l'image dans votre référentiel d'images. Vous pouvez éventuellement fournir des informations VPC pour votre cluster.

> **Important**
>
> Cette action utilise le CodeBuild calcul CodePipeline géré pour exécuter des commandes dans un environnement de génération. L'exécution de l'action des commandes entraînera des frais distincts dans AWS CodeBuild.

> **Note**
>
> L'action de `EKS` déploiement n'est disponible que pour les pipelines de type V2.

L'action EKS soutient les clusters EKS publics et privés. Les clusters privés sont le type recommandé par EKS ; toutefois, les deux types sont pris en charge.

L'action EKS est prise en charge pour les actions entre comptes. Pour ajouter une action EKS intercomptes, ajoutez-la `actionRoleArn` depuis votre compte cible dans la déclaration d'action.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables d'environnement
- Variables de sortie
- Autorisations relatives aux politiques de rôle de service
- Déclaration d'action
- Consultez aussi

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `EKS`
- Version : `1`

Paramètres de configuration

ClusterName

Obligatoire : oui

Le cluster Amazon EKS dans Amazon EKS.

Options sous contrôle

Les options disponibles sont les suivantes lorsque Helm est l'outil de déploiement sélectionné.

HelmReleaseName

Obligatoire : Oui (obligatoire uniquement pour le type de casque)

Le nom de version de votre déploiement.

HelmChartLocation

Obligatoire : Oui (obligatoire uniquement pour le type de casque)

Emplacement de votre déploiement sur la carte.

HelmValuesFiles

Obligatoire : Non (facultatif uniquement pour le type de casque)

Pour remplacer les fichiers de valeurs Helm, entrez les fichiers de valeurs Helm séparés par des virgules à l'emplacement du graphique Helm.

Options sous Kubectl

Les options disponibles sont les suivantes lorsque Kubectl est l'outil de déploiement sélectionné.

ManifestFiles

Obligatoire : Oui (obligatoire uniquement pour le type Kubectl)

Le nom de votre fichier manifeste, le fichier texte qui décrit le nom du conteneur de votre service, ainsi que l'image et la balise. Ce fichier vous permet de paramétrer l'URI de votre image et d'autres informations. Vous pouvez utiliser une variable d'environnement à cette fin.

Vous stockez ce fichier dans le référentiel source de votre pipeline.

Namespace

Obligatoire : non

L'espace de noms Kubernetes à utiliser dans nos commandes. `kubectl` `helm`

Subnets

Obligatoire : non

Les sous-réseaux du VPC de votre cluster. Ils font partie du même VPC qui est attaché à votre cluster. Vous pouvez également fournir des sous-réseaux qui ne sont pas encore rattachés à votre cluster et les spécifier ici.

SecurityGroupIds

Obligatoire : non

Les groupes de sécurité du VPC de votre cluster. Ils font partie du même VPC qui est attaché à votre cluster. Vous pouvez également fournir des groupes de sécurité qui ne sont pas encore attachés à votre cluster et les spécifier ici.

Artefacts d'entrée

- Nombre d'objets : `1`
- Description : L'action recherche le fichier manifeste Kubernetes ou le graphique Helm dans le référentiel de fichiers source du pipeline. Si vous souhaitez utiliser des graphiques Helm au format .tgz stockés dans un compartiment S3, vous pouvez le faire en configurant le S3 Bucket/Key comme action source. Par exemple, la clé d'objet fournie seraitmy`-chart-0.1.0.tgz`.

Artefacts de sortie

- Nombre d'objets : `0`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Variables d'environnement

Utilisé pour remplacer des variables telles que des référentiels d'images ou des balises d'image dans des fichiers manifestes ou des fichiers de valeurs de graphiques Helm.

Clé

La clé d'une paire de variables d'environnement clé-valeur, telle que. `$IMAGE_TAG`

Value

La valeur de la paire clé-valeur, telle que. `v1.0` La valeur peut être paramétrée à l'aide de variables de sortie issues d'actions de pipeline ou de variables de pipeline. Par exemple, le pipeline peut avoir une ECRBuildAndPublish action qui crée une image ECR avec`${codepipeline.PipelineExecutionId}`, et l'action EKS peut utiliser cette image en utilisant `${codepipeline.PipelineExecutionId}` comme valeur de variable d'environnement.

Variables de sortie

EKSClusterName

Le cluster Amazon EKS dans Amazon EKS.

Autorisations relatives aux politiques de rôle de service

Pour exécuter cette action, les autorisations suivantes doivent être disponibles dans la politique de rôle de service de votre pipeline.

- Actions EC2 : lors de l' CodePipeline exécution de l'action, des autorisations d'instance EC2 sont requises. Notez que ce n'est pas la même chose que le rôle d'instance EC2 requis lorsque vous créez votre cluster EKS.

  Si vous utilisez un rôle de service existant, pour utiliser cette action, vous devez ajouter les autorisations suivantes pour ce rôle de service.

  - EC2 : CreateNetworkInterface
  - EC2 : DescribeDhcpOptions
  - EC2 : DescribeNetworkInterfaces
  - EC2 : DeleteNetworkInterface
  - EC2 : DescribeSubnets
  - EC2 : DescribeSecurityGroups
  - EC2 : DescribeVpcs
- Actions EKS : lors de l' CodePipeline exécution de l'action, les autorisations du cluster EKS sont requises. Notez que ce n'est pas la même chose que le rôle de cluster IAM EKS requis lors de la création de votre cluster EKS.

  Si vous utilisez un rôle de service existant, pour utiliser cette action, vous devez ajouter l'autorisation suivante pour ce rôle de service.

  - 4 semaines : DescribeCluster
- Actions du flux de journalisation : lors de l' CodePipeline exécution de l'action, CodePipeline crée un groupe de journaux en utilisant le nom du pipeline comme suit. Cela vous permet de limiter les autorisations d'enregistrement des ressources à l'aide du nom du pipeline.

  ```
  /aws/codepipeline/MyPipelineName
  ```

  Si vous utilisez un rôle de service existant, pour utiliser cette action, vous devez ajouter les autorisations suivantes pour ce rôle de service.

  - journaux : CreateLogGroup
  - journaux : CreateLogStream

- journaux : PutLogEvents

Dans la déclaration de politique relative aux rôles de service, réduisez les autorisations au niveau des ressources, comme indiqué dans l'exemple suivant.

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:*"
            ]
        }
    ]
}
```

Pour afficher les journaux dans la console à l'aide de la page de dialogue des détails des actions, l'autorisation d'affichage des journaux doit être ajoutée au rôle de console. Pour plus d'informations, consultez l'exemple de politique d'autorisations de console dansAutorisations requises pour consulter les journaux de calcul dans la console.

Ajouter le rôle de service en tant qu'entrée d'accès pour votre cluster

Une fois les autorisations disponibles dans la politique de rôle de service de votre pipeline, vous configurez les autorisations de votre cluster en ajoutant le rôle de CodePipeline service en tant qu'entrée d'accès pour votre cluster.

Vous pouvez également utiliser un rôle d'action doté des autorisations mises à jour. Pour plus d'informations, consultez l'exemple du didacticiel dansÉtape 4 : Création d'une entrée d'accès pour le rôle CodePipeline de service.

Déclaration d'action

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"
        }
    ]
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Consultez Tutoriel : Déploiement sur Amazon EKS avec CodePipeline un didacticiel qui montre comment créer un cluster EKS et un fichier manifeste Kubernetes pour ajouter l'action à votre pipeline.

AWS Lambda référence des actions de déploiement

Vous utilisez une action de AWS Lambda déploiement pour gérer le déploiement de votre code d'application pour votre déploiement sans serveur. Vous pouvez déployer une fonction et utiliser des stratégies de déploiement pour le déploiement du trafic comme suit :

- Déploiements linéaires et canariens pour le transfert du trafic

- Déploiements simultanés

> **Note**
>
> Cette action n'est prise en charge que pour les pipelines de type V2.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables de sortie
- Autorisations relatives à la politique de rôle de service pour l'action de déploiement Lambda
- Déclaration d'action
- Consultez aussi

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `Lambda`
- Version : `1`

Paramètres de configuration

FunctionName

Obligatoire : oui

Le nom de la fonction que vous avez créée dans Lambda, par exemple`MyLambdaFunction`.

Vous devez déjà avoir créé une version.

FunctionAlias

Obligatoire : non

L'alias de la fonction que vous avez créée dans Lambda et qui est la fonction à déployer, par exemple`live`. L'alias doit exister et être associé à une version lorsque l'exécution de l'action démarre. (Ce sera la version cible de restauration.)

S'il n'est pas fourni, l'action déploie l'artefact source `$LATEST` et crée une nouvelle version. Dans ce cas d'utilisation, les options de stratégie de déploiement et de version cible ne sont pas disponibles.

PublishedTargetVersion

Obligatoire : non

La version de la fonction Lambda vers laquelle vous souhaitez déployer FunctionAlias. Il peut s'agir de variables de pipeline ou de niveau d'action, telles que`#{variables.lambdaTargetVersion}`. La version doit être publiée lorsque l'exécution de l'action commence.

Obligatoire si aucun artefact d'entrée n'est fourni.

DeployStrategy

Obligatoire : Non (par défaut`AllAtOnce`)

Détermine le débit adopté par l'action de déploiement Lambda pour déplacer le trafic de la version d'origine de la fonction Lambda vers la nouvelle version pour. FunctionAlias Les stratégies de déploiement disponibles sont canariennes ou linéaires. Formats acceptés :

- `AllAtOnce` -

  Déplace tout le trafic vers les fonctions Lambda mises à jour en une seule fois.

  Si aucune valeur n'est spécifiée, la valeur par défaut est `AllAtOnce`.
- `Canary10Percent5Minutes`- Déplace 10 % du trafic dès le premier incrément. Les 90 % restants sont déployés cinq minutes plus tard.

  Les valeurs du pourcentage et des minutes peuvent être modifiées.
- `Linear10PercentEvery1Minute`- Déplace 10 % du trafic toutes les minutes jusqu'à ce que tout le trafic soit modifié.

Les valeurs du pourcentage et des minutes peuvent être modifiées.

Les considérations suivantes s'appliquent à ce domaine :

- Le temps d'attente total maximum est de 2 jours.
- Disponible uniquement lorsque cela FunctionAlias est fourni.

Alarmes

Obligatoire : non

Liste de noms d'alarmes séparés par des virgules configurés pour le déploiement Lambda. Au maximum, 10 alarmes peuvent être ajoutées. L'action échoue lorsque les alarmes surveillées passent à l'état ALARME.

L'image suivante montre un exemple de page d'édition pour l'action.

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

Artefacts d'entrée

- Nombre d'objets : `1`
- Description : fichiers fournis, le cas échéant, pour prendre en charge les actions de script pendant le déploiement.

Artefacts de sortie

- Nombre d'objets : `0`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Variables de sortie

Lorsque cette action est configurée, elle produit des variables qui peuvent être référencées par la configuration d'action d'une action en aval dans le pipeline. Cette action produit des variables qui peuvent être visualisées en tant que variables de sortie, même si l'action n'a pas d'espace de noms. Vous configurez une action avec un espace de noms pour rendre ces variables disponibles pour la configuration des actions en aval.

Pour de plus amples informations, veuillez consulter Référence des variables.

FunctionVersion

La nouvelle version de la fonction Lambda qui a été déployée.

Autorisations relatives à la politique de rôle de service pour l'action de déploiement Lambda

Lors de l' CodePipeline exécution de l'action, le rôle de CodePipeline service nécessite les autorisations suivantes, correctement définies pour un accès avec le moins de privilèges.

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:*"
            ]
        }
    ]
}
```

Déclaration d'action

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"
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Tutoriel : Déploiements de fonctions Lambda avec CodePipeline— Ce didacticiel explique comment créer un exemple de fonction Lambda dans lequel vous allez créer un alias et une version, ajouter la fonction Lambda compressée à votre emplacement source et exécuter l'action Lambda dans votre pipeline.

Référence des actions de déploiement d'Amazon S3

Vous utilisez une action de déploiement Amazon S3 pour déployer des fichiers vers un compartiment Amazon S3 à des fins d'hébergement ou d'archivage de sites Web statiques. Vous pouvez spécifier si vous souhaitez extraire les fichiers de déploiement avant de les charger dans votre compartiment.

> **Note**
>
> Cette rubrique de référence décrit l'action de déploiement d'Amazon S3 CodePipeline lorsque la plateforme de déploiement est un compartiment Amazon S3 configuré pour l'hébergement. Pour obtenir des informations de référence sur l'action source Amazon S3 dans CodePipeline, consultezRéférence des actions source d'Amazon S3.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Autorisations relatives aux rôles de service : action de déploiement S3
- Exemple de configuration d'action
- Consultez aussi

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `S3`
- Version : `1`

Paramètres de configuration

BucketName

Obligatoire : oui

Nom du compartiment Amazon S3 dans lequel les fichiers doivent être déployés.

Extrait

Obligatoire : oui

Si true, indique que les fichiers doivent être extraits avant le téléchargement. Dans le cas contraire, les fichiers de l'application restent compressés pour le téléchargement, comme dans le cas d'un site Web statique hébergé. Si faux, le `ObjectKey` est obligatoire.

ObjectKey

Conditionnelle. Obligatoire si `Extract` = false

Le nom de la clé d'objet Amazon S3 qui identifie de manière unique l'objet dans le compartiment S3.

KMSEncryptionKeyARN

Obligatoire : non

L'ARN de la clé de AWS KMS chiffrement pour le compartiment hôte. Le `KMSEncryptionKeyARN` paramètre chiffre les artefacts téléchargés à l'aide de la valeur fournie AWS KMS key. Pour une clé KMS, vous pouvez utiliser l'ID de clé, l'ARN de clé ou l'alias ARN.

> **Note**
>
> Les alias sont reconnus uniquement dans le compte qui a créé la clé KMS. Pour les actions entre comptes, vous ne pouvez utiliser l'ID de clé ou l'ARN de clé que pour identifier la clé. Cross-account les actions impliquent l'utilisation du rôle de l'autre compte (AccountB). Par conséquent, la spécification de l'ID de clé utilisera la clé de l'autre compte (AccountB).

> **Important**
>
> CodePipeline ne prend en charge que les clés KMS symétriques. N'utilisez pas de clé KMS asymétrique pour chiffrer les données de votre compartiment S3.

CannedACL

Obligatoire : non

Le `CannedACL` paramètre applique l'ACL prédéfini spécifié aux objets déployés sur Amazon S3. Celle-ci remplace n'importe quelle liste ACL existante appliquée à l'objet.

CacheControl

Obligatoire : non

Le `CacheControl` paramètre contrôle le comportement de mise en cache requests/responses pour les objets du bucket. Pour obtenir la liste des valeurs valides, consultez le champ d'en-tête `Cache-Control` pour les opérations HTTP. Pour entrer plusieurs valeurs dans `CacheControl`, utilisez une virgule entre chaque valeur. Vous pouvez ajouter un espace après chaque virgule (facultatif), comme illustré dans cet exemple pour l'interface de ligne de commande :

```
"CacheControl": "public, max-age=0, no-transform"
```

Artefacts d'entrée

- Nombre d'artefacts : 1
- Description : Les fichiers à déployer ou à archiver sont obtenus à partir du référentiel source, compressés et chargés par CodePipeline.

Artefacts de sortie

- Nombre d'objets : `0`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Autorisations relatives aux rôles de service : action de déploiement S3

Pour la prise en charge des actions de déploiement S3, ajoutez ce qui suit à votre déclaration de politique :

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"
        }
```

```
      }
    }
  ]
}
```

Pour la prise en charge de l'action de déploiement S3, si vos objets S3 comportent des balises, vous devez également ajouter les autorisations suivantes à votre déclaration de politique :

```
"s3:GetObjectTagging",
"s3:GetObjectVersionTagging",
"s3:PutObjectTagging"
```

Exemple de configuration d'action

Les exemples suivants illustrent la configuration des actions.

Exemple de configuration lorsque **`Extract`** est défini sur **`false`**

L'exemple suivant montre la configuration d'action par défaut lorsque l'action est créée avec le `Extract` champ défini sur`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"
        }
    ]
},
```

Exemple de configuration lorsque **`Extract`** est défini sur **`true`**

L'exemple suivant montre la configuration d'action par défaut lorsque l'action est créée avec le `Extract` champ défini sur`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"
        }
    ]
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Tutoriel : Création d'un pipeline qui utilise Amazon S3 comme fournisseur de déploiement— Ce didacticiel présente deux exemples de création d'un pipeline avec une action de déploiement S3. Vous téléchargez des exemples de fichiers, vous chargez les fichiers dans votre CodeCommit référentiel, vous créez votre compartiment S3 et vous configurez votre compartiment pour l'hébergement. Ensuite, vous utilisez la CodePipeline console pour créer votre pipeline et spécifier une configuration de déploiement Amazon S3.
- Référence des actions source d'Amazon S3— Cette référence d'action fournit des informations de référence et des exemples pour les actions source d'Amazon S3 dans CodePipeline.

Référence des actions source d'Amazon S3

Une action source déclenche le pipeline lorsqu'un nouvel objet est chargé dans le bucket et la clé d'objet configurés.

Cette rubrique de référence décrit l'action source Amazon S3 CodePipeline lorsque l'emplacement source est un compartiment Amazon S3 configuré pour le versionnage. Pour obtenir des informations de référence sur l'action de déploiement d'Amazon S3 dans CodePipeline, consultezRéférence des actions de déploiement d'Amazon S3.

Avant de connecter le pipeline, créez un compartiment source Amazon S3, activez la gestion des versions et chargez vos fichiers sources dans un seul fichier ZIP.

Vous pouvez créer un nouveau compartiment dans Amazon S3 ou sélectionner un compartiment Amazon S3 existant à utiliser comme emplacement source pour vos fichiers d'application. Pour plus d'informations sur l'activation de la gestion des versions, consultez la section Utilisation de la gestion des versions dans le guide de l'utilisateur Amazon S3.

Lorsqu'Amazon S3 est le fournisseur source de votre pipeline, vous pouvez compresser votre ou vos fichiers source dans un seul fichier .zip et charger le fichier .zip dans votre compartiment source. Vous pouvez également charger un seul fichier décompressé ; toutefois, les actions en aval qui prévoient un fichier .zip échoueront.

Si vous utilisez la console pour créer ou modifier votre pipeline, CodePipeline crée une EventBridge règle qui démarre votre pipeline lorsqu'un changement se produit dans le compartiment source Amazon S3.

Vous pouvez également créer un remplacement de source à l'aide d'une entrée de transformation EventBridge pour utiliser l'`revisionValue`entrée de votre événement de pipeline, où le `revisionValue` est dérivé de la variable d'événement source pour votre clé d'objet, votre commit ou votre ID d'image. Pour plus d'informations, consultez l'étape facultative pour la saisie de la transformation d'entrée incluse dans les procédures ci-dessousConnexion aux actions de source Amazon S3 avec une source activée pour les événements.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables de sortie
- Autorisations relatives aux rôles de service : action source S3
- Déclaration d'action
- Consultez aussi

Type d'action

- Catégorie : `Source`
- Propriétaire : `AWS`
- Fournisseur : `S3`
- Version : `1`

Paramètres de configuration

S3Bucket

Obligatoire : oui

Nom du compartiment Amazon S3 dans lequel les modifications de source doivent être détectées.

S3ObjectKey

Obligatoire : oui

Nom de la clé d'objet Amazon S3 où les modifications de source doivent être détectées.

AllowOverrideForS3ObjectKey

Obligatoire : non

`AllowOverrideForS3ObjectKey`contrôle si les remplacements de source `StartPipelineExecution` peuvent remplacer ceux déjà configurés `S3ObjectKey` dans l'action source. Pour plus d'informations sur les remplacements de source à l'aide de la clé d'objet S3, consultezDémarrer un pipeline avec un remplacement de révision de la source.

> **Important**
>
> Si vous l'omettez`AllowOverrideForS3ObjectKey`, il est possible CodePipeline par défaut de remplacer le S3 ObjectKey dans l'action source en définissant ce paramètre sur. `false`

Valeurs valides pour ce paramètre :

- `true`: Si elle est définie, la clé d'objet S3 préconfigurée peut être remplacée par des modifications de révision de la source lors de l'exécution d'un pipeline.

  > **Note**
  >
  > Si vous souhaitez autoriser tous les CodePipeline utilisateurs à remplacer la clé d'objet S3 préconfigurée lors du démarrage d'une nouvelle exécution de pipeline, vous devez définir sur`AllowOverrideForS3ObjectKey`. `true`

- `false`:

  Si cette option est définie, la clé d'objet S3 ne CodePipeline pourra pas être remplacée à l'aide de modifications de révision de la source. Il s'agit également de la valeur par défaut de ce paramètre.

PollForSourceChanges

Obligatoire : non

`PollForSourceChanges`contrôle si le compartiment source Amazon S3 CodePipeline interroge les modifications apportées à la source. Nous vous recommandons d'utiliser CloudWatch Events et de CloudTrail détecter les modifications de source à la place. Pour plus d'informations sur la configuration CloudWatch des événements, consultez Migrer les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3 (CLI) ouMigrez les pipelines de sondage à l'aide d'une source et d'une CloudTrail piste S3 (CloudFormation modèle).

> **Important**
>
> Si vous avez l'intention de configurer CloudWatch Events, vous devez définir cette `PollForSourceChanges` valeur pour `false` éviter les exécutions de pipeline en double.

Valeurs valides pour ce paramètre :

- `true`: si ce paramètre est défini, CodePipeline interroge votre emplacement source pour détecter les modifications apportées à la source.

  > **Note**
  >
  > Si vous omettez`PollForSourceChanges`, la CodePipeline valeur par défaut est interrogée sur votre emplacement source pour détecter les modifications apportées à la source. Ce comportement est le même que si `PollForSourceChanges` est inclus et défini sur `true`.

- `false`: Si ce paramètre est défini, CodePipeline n'interroge pas votre emplacement source pour détecter les modifications apportées à la source. Utilisez ce paramètre si vous avez l'intention de configurer une règle d' CloudWatch événements pour détecter les modifications de source.

Artefacts d'entrée

- Nombre d'artefacts : `0`
- Description : Les artefacts d'entrée ne s'appliquent pas à ce type d'action.

Artefacts de sortie

- Nombre d'objets : 1
- Description : Fournit les artefacts disponibles dans le compartiment source configuré pour se connecter au pipeline. Les artefacts générés à partir du compartiment sont les artefacts de sortie pour l'action Amazon S3. Les métadonnées de l'objet Amazon S3 (ETag et ID de version) sont affichées en CodePipeline tant que révision source pour l'exécution du pipeline déclenchée.

Variables de sortie

Lorsque cette action est configurée, elle produit des variables qui peuvent être référencées par la configuration d'action d'une action en aval dans le pipeline. Cette action produit des variables qui peuvent être visualisées en tant que variables de sortie, même si l'action n'a pas d'espace de noms. Vous configurez une action avec un espace de noms pour rendre ces variables disponibles pour la configuration des actions en aval.

Pour plus d'informations sur les variables dans CodePipeline, consultezRéférence des variables.

BucketName

Le nom du compartiment Amazon S3 lié à la modification de source qui a déclenché le pipeline.

ETag

Balise d'entité de l'objet lié à la modification de la source ayant déclenché le pipeline. L'ETag est un hachage MD5 de l'objet. ETag reflète uniquement les modifications apportées au contenu d'un objet, non à ses métadonnées.

ObjectKey

Le nom de la clé d'objet Amazon S3 associée au changement de source qui a déclenché le pipeline.

VersionId

ID de version de la version de l'objet lié à la modification de la source ayant déclenché le pipeline.

Autorisations relatives aux rôles de service : action source S3

Pour la prise en charge de l'action source S3, ajoutez ce qui suit à votre déclaration de politique :

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"
        }
      }
    }
  ]
}
```

Déclaration d'action

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": []
        }
    ]
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Didacticiel : Création d'un pipeline simple (compartiment S3)— Ce didacticiel fournit un exemple de AppSpec fichier et un exemple d' CodeDeployapplication et de groupe de déploiement. Utilisez ce didacticiel pour créer un pipeline avec une source Amazon S3 qui se déploie sur les instances Amazon EC2.

AWS AppConfig référence d'action de déploiement

AWS AppConfig est une capacité de AWS Systems Manager. AppConfig prend en charge les déploiements contrôlés vers des applications de toutes tailles et inclut des contrôles de validation et une surveillance intégrés. Vous pouvez l'utiliser AppConfig avec des applications hébergées sur des instances Amazon EC2, des conteneurs AWS Lambda, des applications mobiles ou des appareils IoT.

L'action de `AppConfig` déploiement est une AWS CodePipeline action qui déploie les configurations stockées dans l'emplacement source de votre pipeline vers une AppConfig application, un environnement et un profil de configuration spécifiques. Il utilise les préférences définies dans une stratégie AppConfig de déploiement.

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `AppConfig`
- Version : `1`

Paramètres de configuration

Application

Obligatoire : oui

L'ID de l' AWS AppConfig application avec les détails de votre configuration et de votre déploiement.

Environnement

Obligatoire : oui

ID de l' AWS AppConfig environnement dans lequel la configuration est déployée.

ConfigurationProfile

Obligatoire : oui

ID du profil de AWS AppConfig configuration à déployer.

InputArtifactConfigurationPath

Obligatoire : oui

Le chemin de fichier des données de configuration dans l'artefact d'entrée à déployer.

DeploymentStrategy

Obligatoire : non

Stratégie de AWS AppConfig déploiement à utiliser pour le déploiement.

Artefacts d'entrée

- Nombre d'objets : 1
- Description : artefact d'entrée pour l'action de déploiement.

Artefacts de sortie

Non applicable.

Autorisations relatives aux rôles de service : **`AppConfig`** action

Lors de l' CodePipeline exécution de l'action, la politique de rôle de CodePipeline service nécessite les autorisations suivantes, correctement limitées au niveau de la ressource afin de maintenir l'accès avec le moins de privilèges.

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"
        }
    ]
}
```

Exemple de configuration d'action

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"
        }
    ]
}
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- AWS AppConfig— Pour plus d'informations sur AWS AppConfig les déploiements, consultez le guide de l'AWS Systems Manager utilisateur.
- Tutoriel : Création d'un pipeline utilisé AWS AppConfig comme fournisseur de déploiement— Ce didacticiel vous permet de commencer à configurer des AppConfig ressources et des fichiers de configuration de déploiement simples, et vous montre comment utiliser la console pour créer un pipeline avec une action de AWS AppConfig déploiement.

CloudFormation référence d'action de déploiement

Exécute une opération sur une CloudFormation pile. Une pile est un ensemble de AWS ressources que vous pouvez gérer comme une seule unité. Les ressources d'une pile sont définies par son modèle CloudFormation . Un jeu de modifications crée une comparaison qui peut être affichée sans modifier la pile d'origine. Pour plus d'informations sur les types d' CloudFormation actions pouvant être effectuées sur les piles et les ensembles de modifications, consultez le `ActionMode` paramètre.

Pour créer un message d'erreur pour une CloudFormation action où une opération de pile a échoué, CodePipeline appelez l' CloudFormation `DescribeStackEvents`API. Si un rôle IAM d'action est autorisé à accéder à cette API, les détails concernant la première ressource défaillante seront inclus dans le message CodePipeline d'erreur. Sinon, si la politique de rôle ne dispose pas de l'autorisation appropriée, elle CodePipeline ignorera l'accès à l'API et affichera un message d'erreur générique à la place. Pour ce faire, l'`cloudformation:DescribeStackEvents`autorisation doit être ajoutée au rôle de service ou à d'autres rôles IAM pour le pipeline.

Si vous ne souhaitez pas que les détails des ressources apparaissent dans les messages d'erreur du pipeline, vous pouvez révoquer cette autorisation pour le rôle IAM d'action en la supprimant. `cloudformation:DescribeStackEvents`

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables de sortie
- Autorisations relatives aux rôles de service : CloudFormation action
- Déclaration d'action
- Consultez aussi

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `CloudFormation`
- Version : `1`

Paramètres de configuration

ActionMode

Obligatoire : oui

`ActionMode`est le nom de l'action exécutée CloudFormation sur une pile ou un ensemble de modifications. Les modes d'action suivants sont disponibles :

- `CHANGE_SET_EXECUTE` exécute un jeu de modifications pour la pile de ressources basé sur un ensemble de mises à jour de ressources spécifiées. Avec cette action, CloudFormation commence à modifier la pile.
- `CHANGE_SET_REPLACE` crée le jeu de modification, s'il n'existe pas, en fonction du nom de la pile et du modèle que vous soumettez. Si l'ensemble de modifications existe, CloudFormation il est supprimé, puis en crée un nouveau.
- `CREATE_UPDATE` crée la pile si elle n'existe pas. Si la pile existe, la CloudFormation met à jour. Utilisez cette action pour mettre à jour les piles existantes. Au contraire`REPLACE_ON_FAILURE`, si la pile existe et est en état d'échec, elle CodePipeline ne sera ni supprimée ni remplacée.
- `DELETE_ONLY` supprime une pile. Si vous spécifiez une pile qui n'existe pas, l'action se termine avec succès sans supprimer de pile.
- `REPLACE_ON_FAILURE` crée une pile, si elle n'existe pas. Si la pile existe et est en état d'échec, CloudFormation supprime la pile, puis crée une nouvelle pile. Si la pile n'est pas en état d'échec, mettez-la CloudFormation à jour.

  La pile présente un état d'échec lorsque l'un des types d'état suivants est affiché dans CloudFormation :

  - `ROLLBACK_FAILED`
  - `CREATE_FAILED`
  - `DELETE_FAILED`

- `UPDATE_ROLLBACK_FAILED`

Utilisez cette action pour remplacer automatiquement les piles ayant échoué sans les récupérer ou les dépanner.

> **Important**
>
> Nous vous recommandons d'utiliser `REPLACE_ON_FAILURE` à des fins de test uniquement, car cela peut supprimer votre pile.

StackName

Obligatoire : oui

`StackName` correspond au nom d'une pile existante ou d'une pile que vous souhaitez créer.

Fonctionnalités

Obligatoire : Conditionnelle

L'utilisation de `Capabilities` confirme que le modèle peut avoir les capacités de créer et de mettre à jour certaines ressources par lui-même, et que ces capacités sont déterminées en fonction des types de ressources figurant dans le modèle.

Cette propriété est requise si vous avez des ressources IAM dans votre modèle de pile ou si vous créez une pile directement à partir d'un modèle contenant des macros. Pour que l' CloudFormation action fonctionne correctement de cette manière, vous devez explicitement reconnaître que vous souhaitez qu'elle le fasse avec l'une des fonctionnalités suivantes :

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`
- `CAPABILITY_AUTO_EXPAND`

Vous pouvez spécifier plusieurs capacités en utilisant une virgule (pas d'espace) entre les capacités. L'exemple fourni dans Déclaration d'action montre une entrée avec les propriétés CAPABILITY_IAM et CAPABILITY_AUTO_EXPAND.

Pour plus d'informations`Capabilities`, consultez les propriétés ci-dessous UpdateStackdans la référence de l'AWS CloudFormation API.

ChangeSetName

Obligatoire : Conditionnelle

`ChangeSetName` est le nom d'un jeu de modifications existant ou d'un nouveau jeu de modifications que vous souhaitez créer pour la pile spécifiée.

Cette propriété est obligatoire pour les modes d'action suivants : CHANGE_SET_REPLACE et CHANGE_SET_EXECUTE. Pour tous les autres modes d'action, cette propriété est ignorée.

RoleArn

Obligatoire : Conditionnelle

`RoleArn` est l'ARN du rôle de service IAM assumé par CloudFormation lorsqu'il fonctionne sur les ressources de la pile spécifiée. `RoleArn` n'est pas appliqué lors de l'exécution d'un jeu de modifications. Si vous ne l'utilisez pas CodePipeline pour créer l'ensemble de modifications, assurez-vous qu'un rôle est associé à l'ensemble ou à la pile de modifications.

> **Note**
>
> Ce rôle doit se trouver dans le même compte que le rôle de l'action en cours d'exécution, tel que configuré dans la déclaration d'action`RoleArn`.

Cette propriété est requise pour les modes d'action suivants :

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- DELETE_ONLY
- CHANGE_SET_REPLACE

> **Note**
>
> CloudFormation reçoit une URL signée S3 vers le modèle ; par conséquent, aucune autorisation `RoleArn` n'est requise pour accéder au compartiment d'artefacts. Toutefois, l'action `RoleArn` nécessite une autorisation pour accéder au compartiment d'artefacts afin de générer l'URL signée.

TemplatePath

Obligatoire : Conditionnelle

`TemplatePath`représente le fichier CloudFormation modèle. Vous incluez ce fichier dans un artefact d'entrée pour cette action. Le nom de fichier respecte le format suivant :

`Artifactname``::`*`TemplateFileName`*

`Artifactname` est le nom de l'artefact d'entrée tel qu'il apparaît dans CodePipeline. Par exemple, une étape source avec le nom d'artefact `SourceArtifact` et un nom de fichier `template-export.json` crée un nom `TemplatePath`, tel qu'illustré dans l'exemple suivant :

```
"TemplatePath": "SourceArtifact::template-export.json"
```

Cette propriété est requise pour les modes d'action suivants :

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- CHANGE_SET_REPLACE

Pour tous les autres modes d'action, cette propriété est ignorée.

> **Note**
>
> Le fichier CloudFormation modèle contenant le corps du modèle a une longueur minimale de 1 octet et une longueur maximale de 1 Mo. Pour les actions de CloudFormation déploiement dans CodePipeline, la taille maximale de l'artefact en entrée est toujours de 256 Mo. Pour plus d'informations, consultez Quotas dans AWS CodePipeline et Limites CloudFormation.

OutputFileName

Obligatoire : non

`OutputFileName`À utiliser pour spécifier un nom de fichier de sortie, tel que`CreateStackOutput.json`, qui s' CodePipeline ajoute à l'artefact de sortie du pipeline pour cette action. Le fichier JSON contient le contenu de la `Outputs` section issue de la CloudFormation pile.

Si vous ne spécifiez pas de nom, CodePipeline cela ne génère pas de fichier de sortie ou d'artefact.

ParameterOverrides

Obligatoire : non

Les paramètres sont définis dans votre modèle de pile et vous permettent de fournir leurs valeurs au moment de la création ou de la mise à jour de la pile. Vous pouvez utiliser un objet JSON pour définir les valeurs des paramètres dans votre modèle. (Ces valeurs remplacent celles définies dans le fichier de configuration du modèle.) Pour de plus amples informations sur l'utilisation des remplacements de paramètres, veuillez consulter Propriétés de configuration (objet JSON).

Nous vous recommandons d'utiliser le fichier de configuration de modèle pour la plupart de vos valeurs de paramètres. Utilisez des remplacements de paramètres uniquement pour les valeurs qui ne sont pas connues tant que le pipeline n'est pas en cours d'exécution. Pour plus d'informations, consultez la section Utilisation des fonctions de remplacement de paramètres avec des CodePipeline pipelines dans le guide de l'AWS CloudFormation utilisateur.

> **Note**
>
> Tous les noms de paramètres doivent être présents dans le modèle de la pile.

TemplateConfiguration

Obligatoire : non

`TemplateConfiguration` est le fichier de configuration de modèle. Vous incluez ce fichier dans un artefact d'entrée pour cette action. Il peut contenir des valeurs de paramètre de modèle et une stratégie de pile. Pour plus d'informations sur le format de fichier de configuration du modèle, consultez AWS CloudFormation Artefacts.

Le nom de fichier de configuration de modèle suit ce format :

`Artifactname``::`*`TemplateConfigurationFileName`*

`Artifactname` est le nom de l'artefact d'entrée tel qu'il apparaît dans CodePipeline. Par exemple, une étape source avec le nom d'artefact `SourceArtifact` et un nom de fichier `test-configuration.json` crée un nom `TemplateConfiguration`, tel qu'illustré dans l'exemple suivant :

```
"TemplateConfiguration": "SourceArtifact::test-configuration.json"
```

Artefacts d'entrée

- Nombre d'objets : `0 to 10`
- Description : en entrée, l' CloudFormation action accepte éventuellement des artefacts aux fins suivantes :
  - Pour fournir le fichier de modèle de pile à exécuter. (Voir le paramètre `TemplatePath`.)
  - Pour fournir le fichier de configuration de modèle à utiliser. (Voir le paramètre `TemplateConfiguration`.) Pour plus d'informations sur le format de fichier de configuration du modèle, consultez AWS CloudFormation Artefacts.
  - Fournir l'artefact pour une fonction Lambda à déployer dans le cadre de la CloudFormation pile.

Artefacts de sortie

- Nombre d'objets : `0 to 1`
- Description : si le paramètre `OutputFileName` est spécifié, un artefact de sortie produit par cette action contient un fichier JSON portant le nom spécifié. Le fichier JSON contient le contenu de la section Outputs de la pile CloudFormation .

  Pour de plus amples informations sur la section Outputs que vous pouvez créer pour votre action CloudFormation , veuillez consulter Sorties.

Variables de sortie

Lorsque cette action est configurée, elle produit des variables qui peuvent être référencées par la configuration d'action d'une action en aval dans le pipeline. Vous configurez une action avec un espace de noms pour rendre ces variables disponibles pour la configuration des actions en aval.

Pour les CloudFormation actions, les variables sont produites à partir de toutes les valeurs désignées dans la `Outputs` section d'un modèle de pile. Notez que les seuls modes CloudFormation d'action qui génèrent des sorties sont ceux qui aboutissent à la création ou à la mise à jour d'une pile, tels que la création de pile, les mises à jour de piles et l'exécution d'ensembles de modifications. Les modes d'action correspondants qui génèrent des variables sont les suivants :

- `CHANGE_SET_EXECUTE`
- `CHANGE_SET_REPLACE`
- `CREATE_UPDATE`

- REPLACE_ON_FAILURE

Pour de plus amples informations, veuillez consulter Référence des variables. Pour un didacticiel expliquant comment créer un pipeline avec une action de CloudFormation déploiement dans un pipeline utilisant des variables CloudFormation de sortie, voirTutoriel : Création d'un pipeline qui utilise des variables provenant de AWS CloudFormation actions de déploiement.

Autorisations relatives aux rôles de service : CloudFormation action

Lors de l' CodePipeline exécution de l'action, la politique de rôle de CodePipeline service nécessite les autorisations suivantes, correctement limitées à l'ARN de la ressource du pipeline afin de maintenir l'accès avec le moins de privilèges. Par exemple, ajoutez ce qui suit à votre déclaration de politique :

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"
                    ]
                }
            }
        }
    ]
}
```

Notez que l'`cloudformation:DescribeStackEvents`autorisation est facultative. Cela permet à l' CloudFormation action d'afficher un message d'erreur plus détaillé. Cette autorisation peut être révoquée pour le rôle IAM si vous ne souhaitez pas que les détails des ressources apparaissent dans les messages d'erreur du pipeline.

Déclaration d'action

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"
        }
```

```
    ]
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Référence des propriétés de configuration — Ce chapitre de référence du Guide de AWS CloudFormation l'utilisateur fournit des descriptions et des exemples supplémentaires pour ces CodePipeline paramètres.
- AWS CloudFormation Référence d'API — Le CreateStackparamètre de la référence AWS CloudFormation d'API décrit les paramètres de pile pour les CloudFormation modèles.

CloudFormation StackSets référence des actions de déploiement

CodePipeline offre la possibilité d'effectuer des CloudFormation StackSets opérations dans le cadre de votre CI/CD processus. Vous utilisez un ensemble de piles pour créer des piles dans les AWS comptes des différentes AWS régions à l'aide d'un CloudFormation modèle unique. Toutes les ressources incluses dans chaque pile sont définies par le CloudFormation modèle de l'ensemble de piles. Lorsque vous créez le jeu de piles, vous spécifiez le modèle à utiliser, ainsi que tous les paramètres et fonctionnalités requis par le modèle.

Pour plus d'informations sur les concepts de CloudFormation StackSets, consultez StackSets les concepts du Guide de AWS CloudFormation l'utilisateur.

Vous intégrez votre pipeline CloudFormation StackSets grâce à deux types d'actions distincts que vous utilisez ensemble :

- L'`CloudFormationStackSet`action crée ou met à jour un ensemble de piles ou des instances de pile à partir du modèle stocké dans l'emplacement source du pipeline. Chaque fois qu'un ensemble de piles est créé ou mis à jour, il lance le déploiement de ces modifications sur des instances spécifiées. Dans la console, vous pouvez choisir le fournisseur d'actions CloudFormation Stack Set lorsque vous créez ou modifiez votre pipeline.
- L'`CloudFormationStackInstances`action déploie les modifications apportées à l'`CloudFormationStackSet`action vers des instances spécifiées, crée de nouvelles instances de pile et définit des remplacements de paramètres pour les instances spécifiées. Dans la console,

vous pouvez choisir le fournisseur d'actions CloudFormation Stack Instances lorsque vous modifiez un pipeline existant.

Vous pouvez utiliser ces actions pour déployer sur des AWS comptes cibles ou des identifiants d'unité organisationnelle des AWS organisations cibles.

> **Note**
>
> Pour effectuer un déploiement sur AWS des comptes ou des identifiants d'unités organisationnelles cibles et utiliser le modèle d'autorisations gérées par les services, vous devez activer l'accès sécurisé entre AWS CloudFormation StackSets et AWS les organisations. Pour plus d'informations, consultez la section Activation de l'accès sécurisé avec AWS CloudFormation Stacksets.

Rubriques

- Comment ? CloudFormation StackSets les actions fonctionnent
- Comment structurer StackSets les actions dans un pipeline
- L'action CloudFormationStackSet
- L' CloudFormationStackInstances action
- Autorisations relatives aux rôles de service : CloudFormationStackSet action
- Autorisations relatives aux rôles de service : CloudFormationStackInstances action
- Modèles d'autorisations pour les opérations relatives aux ensembles de piles
- Types de données des paramètres du modèle
- Consultez aussi

Comment ? CloudFormation StackSets les actions fonctionnent

Une `CloudFormationStackSet` action crée ou met à jour des ressources selon qu'elle est exécutée pour la première fois ou non.

L'`CloudFormationStackSet`action crée ou met à jour l'ensemble de piles et déploie ces modifications sur les instances spécifiées.

> **Note**
>
> Si vous utilisez cette action pour effectuer une mise à jour qui inclut l'ajout d'instances de pile, les nouvelles instances sont déployées en premier et la mise à jour est terminée en dernier. Les nouvelles instances reçoivent d'abord l'ancienne version, puis la mise à jour est appliquée à toutes les instances.

- Créer : lorsqu'aucune instance n'est spécifiée et que le jeu de piles n'existe pas, l'CloudFormationStackSetaction crée le jeu de piles sans créer d'instances.
- Mise à jour : lorsque l'CloudFormationStackSetaction est exécutée pour un ensemble de piles déjà créé, l'action met à jour l'ensemble de piles. Si aucune instance n'est spécifiée et que l'ensemble de piles existe déjà, toutes les instances sont mises à jour. Si cette action est utilisée pour mettre à jour des instances spécifiques, toutes les instances restantes passent au statut OBSOLÈTE.

  Vous pouvez utiliser cette CloudFormationStackSet action pour mettre à jour l'ensemble de piles de la manière suivante.

  - Mettez à jour le modèle sur certaines instances ou sur toutes les instances.
  - Mettez à jour les paramètres de certaines ou de toutes les instances.
  - Mettez à jour le rôle d'exécution pour l'ensemble de piles (celui-ci doit correspondre au rôle d'exécution spécifié dans le rôle Administrateur).
  - Modifiez le modèle d'autorisations (uniquement si aucune instance n'a été créée).
  - Enable/Disable `AutoDeployment`si le modèle d'autorisations de l'ensemble de piles est`Service Managed`.
  - Agissez en tant qu'administrateur délégué dans un compte membre si le modèle d'autorisations de l'ensemble de piles l'est`Service Managed`.
  - Mettez à jour le rôle d'administrateur.
  - Mettez à jour la description de l'ensemble de piles.
  - Ajoutez des cibles de déploiement à la mise à jour de l'ensemble de piles pour créer de nouvelles instances de pile.

L'`CloudFormationStackInstances`action crée de nouvelles instances de pile ou met à jour des instances de pile obsolètes. Une instance devient obsolète lorsqu'un ensemble de piles est mis à jour, mais toutes les instances qu'il contient ne sont pas mises à jour.

- Créer : si la pile existe déjà, l'`CloudFormationStackInstances`action met uniquement à jour les instances et ne crée pas d'instances de pile.
- Mise à jour : une fois l'`CloudFormationStackSet`action exécutée, si le modèle ou les paramètres n'ont été mis à jour que dans certains cas, le reste sera marqué`OUTDATED`. Lors des étapes ultérieures du pipeline, `CloudFormationStackInstances` met à jour le reste des instances de la pile par vagues afin que toutes les instances soient marquées`CURRENT`. Cette action peut également être utilisée pour ajouter des instances supplémentaires ou modifier des paramètres sur des instances nouvelles ou existantes.

Dans le cadre d'une mise à jour, les `CloudFormationStackInstances` actions `CloudFormationStackSet` et peuvent spécifier de nouvelles cibles de déploiement, ce qui crée de nouvelles instances de pile.

Dans le cadre d'une mise à jour, les `CloudFormationStackInstances` actions `CloudFormationStackSet` et ne suppriment pas les ensembles de piles, les instances ou les ressources. Lorsque l'action met à jour une pile mais ne spécifie pas toutes les instances à mettre à jour, les instances qui n'ont pas été spécifiées pour la mise à jour sont supprimées de la mise à jour et leur statut est défini sur`OUTDATED`.

Au cours d'un déploiement, les instances de pile peuvent également indiquer `OUTDATED` si le déploiement vers les instances a échoué.

Comment structurer StackSets les actions dans un pipeline

Il est recommandé de construire votre pipeline de manière à ce que l'ensemble de piles soit créé et initialement déployé sur un sous-ensemble ou une seule instance. Après avoir testé votre déploiement et visualisé l'ensemble de piles généré, ajoutez l'`CloudFormationStackInstances`action pour que les instances restantes soient créées et mises à jour.

Utilisez la console ou l'interface de ligne de commande pour créer la structure de pipeline recommandée comme suit :

1. Créez un pipeline avec une action source (obligatoire) et l'`CloudFormationStackSet`action en tant qu'action de déploiement. Gérez votre pipeline.
2. Lorsque votre pipeline s'exécute pour la première fois, l'`CloudFormationStackSet`action crée votre stack set et au moins une instance initiale. Vérifiez la création de l'ensemble de piles et passez en revue le déploiement sur votre instance initiale. Par exemple, pour la création initiale

de l'ensemble de piles pour le compte Account-A où se `us-east-1` trouve la région spécifiée, l'instance de pile est créée avec l'ensemble de piles :

Instance de pile	Région	Statut
StackInstanceID-1	us-east-1	CURRENT

3. Modifiez votre pipeline pour l'ajouter `CloudFormationStackInstances` comme deuxième action de déploiement afin d' create/update empiler les instances pour les cibles que vous désignez. Par exemple, pour la création d'une instance de pile pour un compte `Account-A` où les `eu-central-1` régions `us-east-2` et sont spécifiées, les instances de pile restantes sont créées et l'instance initiale reste mise à jour comme suit :

Instance de pile	Région	Statut
StackInstanceID-1	us-east-1	CURRENT
StackInstanceID-2	us-east-2	CURRENT
StackInstanceID-3	eu-central-1	CURRENT

4. Exécutez votre pipeline selon vos besoins pour mettre à jour votre ensemble de piles et mettre à jour ou créer des instances de pile.

Lorsque vous lancez une mise à jour de pile au cours de laquelle vous avez supprimé des cibles de déploiement de la configuration d'action, les instances de pile qui n'étaient pas destinées à être mises à jour sont supprimées du déploiement et passent à l'état OBSOLÈTE. Par exemple, pour la mise à jour d'une instance de pile pour un compte `Account-A` dont la `us-east-2` région est supprimée de la configuration de l'action, les instances de pile restantes sont créées et l'instance supprimée est définie sur OBSOLETED comme suit :

Instance de pile	Région	Statut
StackInstanceID-1	us-east-1	CURRENT
StackInstanceID-2	us-east-2	DÉPASSÉ
StackInstanceID-3	eu-central-1	CURRENT

Pour plus d'informations sur les meilleures pratiques en matière de déploiement de stack sets, consultez la section Meilleures pratiques du Guide de AWS CloudFormation l'utilisateur. StackSets

L'action `CloudFormationStackSet`

Cette action crée ou met à jour un ensemble de piles à partir du modèle stocké dans l'emplacement source du pipeline.

Après avoir défini un ensemble de piles, vous pouvez créer, mettre à jour ou supprimer des piles dans les comptes cibles et les régions spécifiés dans les paramètres de configuration. Lors de la création, de la mise à jour et de la suppression de piles, vous pouvez spécifier d'autres préférences, telles que l'ordre des régions pour les opérations à effectuer, le pourcentage de tolérance aux pannes au-delà duquel les opérations de pile s'arrêtent et le nombre de comptes dans lesquels les opérations sont effectuées simultanément sur les piles.

Un ensemble de piles est une ressource régionale. Si vous créez un ensemble de piles dans une AWS région, vous ne pouvez pas y accéder depuis d'autres régions.

Lorsque cette action est utilisée comme action de mise à jour de l'ensemble de piles, les mises à jour de la pile ne sont pas autorisées sans un déploiement sur au moins une instance de pile.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables de sortie
- Exemple CloudFormationStackSet configuration des actions

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `CloudFormationStackSet`
- Version : `1`

Paramètres de configuration

StackSetName

Obligatoire : oui

Nom à associer à l'ensemble de piles. Ce nom doit être unique dans la région où il a été créé.

Le nom ne peut contenir que des caractères alphanumériques et des tirets. Il doit commencer par un caractère alphabétique et comporter 128 caractères ou moins.

Description

Obligatoire : non

Description de l'ensemble de piles. Vous pouvez l'utiliser pour décrire l'objectif de l'ensemble de piles ou pour toute autre information pertinente.

TemplatePath

Obligatoire : oui

Emplacement du modèle qui définit les ressources de l'ensemble de piles. Cela doit pointer vers un modèle d'une taille maximale de 460 800 octets.

Entrez le chemin d'accès au nom de l'artefact source et au fichier modèle au format `"InputArtifactName::TemplateFileName"` indiqué dans l'exemple suivant.

```
SourceArtifact::template.txt
```

Parameters

Obligatoire : non

Liste des paramètres de modèle pour votre stack set qui sont mis à jour lors d'un déploiement.

Vous pouvez fournir des paramètres sous forme de liste littérale ou de chemin de fichier :

- Vous pouvez saisir des paramètres dans le format de syntaxe abrégé suivant :. `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  Pour plus d'informations sur ces types de données, consultezTypes de données des paramètres du modèle.

L'exemple suivant montre un paramètre nommé `BucketName` avec la valeur`amzn-s3-demo-source-bucket`.

```
ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
```

L'exemple suivant montre une entrée avec plusieurs paramètres :

```
ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
ParameterKey=Asset1,ParameterValue=true
ParameterKey=Asset2,ParameterValue=true
```

- Vous pouvez entrer l'emplacement du fichier contenant la liste des remplacements de paramètres de modèle saisis au format"`InputArtifactName::ParametersFileName`", comme illustré dans l'exemple suivant.

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

L'exemple suivant montre le contenu du fichier pour`parameters.txt`.

```
[
    {
        "ParameterKey": "KeyName",
        "ParameterValue": "true"
    },
    {
        "ParameterKey": "KeyName",
        "ParameterValue": "true"
    }
]
```

Fonctionnalités

Obligatoire : non

Indique que le modèle peut créer et mettre à jour des ressources, en fonction des types de ressources du modèle.

Vous devez utiliser cette propriété si votre modèle de pile contient des ressources IAM ou si vous créez une pile directement à partir d'un modèle contenant des macros. Pour que l' CloudFormation action fonctionne correctement de cette manière, vous devez utiliser l'une des fonctionnalités suivantes :

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`

Vous pouvez spécifier plusieurs fonctionnalités en utilisant une virgule et sans espace entre les fonctionnalités. L'exemple ci-dessous Exemple CloudFormationStackSet configuration des actions montre une entrée avec de multiples fonctionnalités.

PermissionModel

Obligatoire : non

Détermine la manière dont les rôles IAM sont créés et gérés. Si le champ n'est pas spécifié, la valeur par défaut est utilisée. Pour plus d'informations, consultez Modèles d'autorisations pour les opérations relatives aux ensembles de piles.

Les valeurs valides sont :

- `SELF_MANAGED`(par défaut) : vous devez créer des rôles d'administrateur et d'exécution à déployer sur les comptes cibles.
- `SERVICE_MANAGED`: crée CloudFormation StackSets automatiquement les rôles IAM nécessaires au déploiement sur les comptes gérés par AWS les organisations. Cela nécessite un compte pour être membre d'une organisation.

> **Note**
>
> Ce paramètre ne peut être modifié que s'il n'existe aucune instance de pile dans le jeu de piles.

AdministrationRoleArn

> **Note**
>
> Comme il CloudFormation StackSets effectue des opérations sur plusieurs comptes, vous devez définir les autorisations nécessaires sur ces comptes avant de pouvoir créer le stack set.

Obligatoire : non

> **Note**
>
> Ce paramètre est facultatif pour le modèle d'autorisations SELF_MANAGED et n'est pas utilisé pour le modèle d'autorisations SERVICE_MANAGED.

L'ARN du rôle IAM dans le compte administrateur utilisé pour effectuer des opérations d'ensemble de piles.

Le nom peut contenir des caractères alphanumériques, l'un des caractères suivants : _+=, .@-, et aucun espace. Le nom ne fait pas la distinction entre majuscules et minuscules. Ce nom de rôle doit avoir une longueur minimale de 20 caractères et une longueur maximale de 2048 caractères. Les noms de rôle doivent être uniques au sein du compte. Le nom de rôle spécifié ici doit être un nom de rôle existant. Si vous ne spécifiez pas le nom du rôle, celui-ci est défini sur AWSCloudFormationStackSetAdministrationRole. Si vous le spécifiez ServiceManaged, vous ne devez pas définir de nom de rôle.

ExecutionRoleName

> **Note**
>
> Comme il CloudFormation StackSets effectue des opérations sur plusieurs comptes, vous devez définir les autorisations nécessaires sur ces comptes avant de pouvoir créer le stack set.

Obligatoire : non

> **Note**
>
> Ce paramètre est facultatif pour le modèle d'autorisations SELF_MANAGED et n'est pas utilisé pour le modèle d'autorisations SERVICE_MANAGED.

Nom du rôle IAM dans les comptes cibles utilisés pour effectuer des opérations d'ensemble de piles. Le nom peut contenir des caractères alphanumériques, l'un des caractères suivants : _ +=, .@-, et aucun espace. Le nom ne fait pas la distinction entre majuscules et minuscules.

Ce nom de rôle doit avoir une longueur minimale de 1 caractère et une longueur maximale de 64 caractères. Les noms de rôle doivent être uniques au sein du compte. Le nom de rôle spécifié ici doit être un nom de rôle existant. Ne spécifiez pas ce rôle si vous utilisez des rôles d'exécution personnalisés. Si vous ne spécifiez pas le nom du rôle, celui-ci est défini sur`AWSCloudFormationStackSetExecutionRole`. Si vous définissez Service_Managed ce paramètre sur true, vous ne devez pas définir de nom de rôle.

OrganizationsAutoDeployment

Obligatoire : non

> **Note**
>
> Ce paramètre est facultatif pour le modèle d'autorisations SERVICE_MANAGED et n'est pas utilisé pour le modèle d'autorisations SELF_MANAGED.

Décrit si le déploiement est CloudFormation StackSets automatique vers AWS les comptes des organisations ajoutés à une organisation ou à une unité organisationnelle (UO) cible. Si `OrganizationsAutoDeployment` c'est spécifié, ne spécifiez pas `DeploymentTargets` et`Regions`.

> **Note**
>
> Si aucune entrée n'est fournie`OrganizationsAutoDeployment`, la valeur par défaut est`Disabled`.

Les valeurs valides sont :

- `Enabled`. Obligatoire : Non.

  StackSets déploie automatiquement des instances de pile supplémentaires sur AWS les comptes des organisations qui sont ajoutés à une organisation ou à une unité organisationnelle (UO) cible dans les régions spécifiées. Si un compte est supprimé d'une organisation ou d'une unité d'organisation cible, CloudFormation StackSets supprime les instances de pile du compte dans les régions spécifiées.
- `Disabled`. Obligatoire : Non.

StackSets ne déploie pas automatiquement des instances de pile supplémentaires sur AWS les comptes des organisations qui sont ajoutés à une organisation ou à une unité organisationnelle (UO) cible dans les régions spécifiées.

- `EnabledWithStackRetention`. Obligatoire : Non.

  Les ressources de la pile sont conservées lorsqu'un compte est supprimé d'une organisation ou d'une unité d'organisation cible.

DeploymentTargets

Obligatoire : non

> **Note**
>
> Pour le modèle d'autorisations SERVICE_MANAGED, vous pouvez fournir l'ID racine de l'organisation ou les ID d'unité organisationnelle pour les cibles de déploiement. Pour le modèle d'autorisations SELF_MANAGED, vous ne pouvez fournir que des comptes.

> **Note**
>
> Lorsque ce paramètre est sélectionné, vous devez également sélectionner Régions.

Une liste de AWS comptes ou d'identifiants d'unités organisationnelles où doivent se trouver les instances de stack set created/updated.

- Comptes :

  Vous pouvez fournir des comptes sous forme de liste littérale ou de chemin de fichier :

  - Littéral : entrez les paramètres dans le format de syntaxe abrégé`account_ID,account_ID`, comme indiqué dans l'exemple suivant.

    ```
    111111222222,333333444444
    ```

  - Chemin du fichier : emplacement du fichier contenant une liste de AWS comptes où doivent se trouver les instances de stack set created/updated, saisi au format`InputArtifactName::AccountsFileName`. Si vous utilisez le chemin du fichier

pour spécifier des comptes ou OrganizationalUnitIds, le format de fichier doit être au format JSON, comme illustré dans l'exemple suivant.

```
SourceArtifact::accounts.txt
```

L'exemple suivant montre le contenu du fichier pour`accounts.txt`.

```
[
    "111111222222"
]
```

L'exemple suivant montre le contenu du fichier `accounts.txt` lorsque plusieurs comptes sont répertoriés :

```
[
    "111111222222","333333444444"
]
```

- OrganizationalUnitIds:

> **Note**
>
> Ce paramètre est facultatif pour le modèle d'autorisations SERVICE_MANAGED et n'est pas utilisé pour le modèle d'autorisations SELF_MANAGED. Ne l'utilisez pas si vous sélectionnez OrganizationsAutoDeployment.

Les unités AWS organisationnelles dans lesquelles mettre à jour les instances de pile associées.

Vous pouvez fournir des ID d'unité organisationnelle sous forme de liste littérale ou de chemin de fichier :

- Littéral : entrez un tableau de chaînes séparées par des virgules, comme illustré dans l'exemple suivant.

```
ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
```

- Chemin du fichier : emplacement du fichier contenant la liste des instances OrganizationalUnitIds dans lesquelles créer ou mettre à jour des instances d'ensembles

de piles. Si vous utilisez le chemin du fichier pour spécifier des comptes ou OrganizationalUnitIds, le format de fichier doit être au format JSON, comme illustré dans l'exemple suivant.

Entrez le chemin d'accès au fichier au format`InputArtifactName::OrganizationalUnitIdsFileName`.

```
SourceArtifact::OU-IDs.txt
```

L'exemple suivant montre le contenu du fichier pour `OU-IDs.txt` :

```
[
    "ou-examplerootid111-exampleouid111","ou-examplerootid222-exampleouid222"
]
```

Régions

Obligatoire : non

> **Note**
>
> Lorsque ce paramètre est sélectionné, vous devez également le sélectionner DeploymentTargets.

Liste des AWS régions dans lesquelles les instances d'ensembles de piles sont créées ou mises à jour. Les régions sont mises à jour dans l'ordre dans lequel elles ont été saisies.

Entrez une liste de AWS régions valides au format `Region1,Region2` indiqué dans l'exemple suivant.

```
us-west-2,us-east-1
```

FailureTolerancePercentage

Obligatoire : non

Le pourcentage de comptes par région pour lesquels cette opération de pile peut échouer avant l' CloudFormation arrêt de l'opération dans cette région. Si l'opération est arrêtée dans une région,

CloudFormation ne tente pas l'opération dans les régions suivantes. Lorsque vous calculez le nombre de comptes en fonction du pourcentage spécifié, CloudFormation arrondissez au nombre entier inférieur.

MaxConcurrentPercentage

Obligatoire : non

Pourcentage maximum de comptes dans lequel vous souhaitez effectuer cette opération simultanément. Lorsque vous calculez le nombre de comptes en fonction du pourcentage spécifié, CloudFormation arrondissez au nombre entier inférieur. Si l'arrondissement vers le bas CloudFormation donne zéro, définissez plutôt le nombre comme un. Bien que vous utilisiez ce paramètre pour spécifier le maximum, pour les déploiements de grande envergure, le nombre réel de comptes traités simultanément peut être inférieur en raison de la limitation des services.

RegionConcurrencyType

Obligatoire : non

Vous pouvez spécifier si l'ensemble de piles doit être déployé de Régions AWS manière séquentielle ou parallèle en configurant le paramètre de déploiement simultané des régions. Lorsque la simultanéité régionale est spécifiée pour déployer des piles sur plusieurs Régions AWS en parallèle, cela peut se traduire par des temps de déploiement globaux plus rapides.

- Parallèle : les déploiements d'ensembles de piles seront effectués en même temps, à condition que les échecs de déploiement d'une région ne dépassent pas une tolérance de défaillance spécifiée.
- Séquentiel : les déploiements des ensembles de piles seront effectués un par un, à condition que les échecs de déploiement d'une région ne dépassent pas une tolérance de défaillance spécifiée. Le déploiement séquentiel est la sélection par défaut.

ConcurrencyMode

Obligatoire : non

Le mode de simultanéité vous permet de choisir le comportement du niveau de concurrence lors des opérations d'ensemble de piles, que ce soit avec une tolérance de défaillance stricte ou souple. La Tolérance stricte aux pannes réduit la vitesse de déploiement en cas de défaillance des opérations d'ensemble de piles, car la simultanéité diminue à chaque défaillance. Soft Failure Tolerance donne la priorité à la vitesse de déploiement tout en tirant parti des capacités CloudFormation de sécurité.

- `STRICT_FAILURE_TOLERANCE`: cette option abaisse dynamiquement le niveau de simultanéité pour garantir que le nombre de comptes défaillants ne dépasse jamais une certaine tolérance de défaillance. Il s'agit du comportement de par défaut.
- `SOFT_FAILURE_TOLERANCE`: cette option dissocie la tolérance aux pannes de la simultanéité réelle. Cela permet aux opérations d'ensemble de piles de s'exécuter à un niveau de simultanéité défini, quel que soit le nombre d'échecs.

CallAs

Obligatoire : non

> **Note**
>
> Ce paramètre est facultatif pour le modèle `SERVICE_MANAGED` d'autorisations et n'est pas utilisé pour le modèle `SELF_MANAGED` d'autorisations.

Spécifie si vous agissez sur le compte de gestion de l'organisation ou en tant qu'administrateur délégué sur un compte de membre.

> **Note**
>
> Si ce paramètre est défini sur`DELEGATED_ADMIN`, assurez-vous que le rôle IAM du pipeline est `organizations:ListDelegatedAdministrators` autorisé. Sinon, l'action échouera lors de l'exécution avec une erreur similaire à la suivante : `Account used is not a delegated administrator`

- `SELF`: le déploiement de Stack Set utilisera les autorisations gérées par le service lorsque vous êtes connecté au compte de gestion.
- `DELEGATED_ADMIN`: le déploiement de Stack Set utilisera les autorisations gérées par les services lorsque vous êtes connecté à un compte d'administrateur délégué.

Artefacts d'entrée

Vous devez inclure au moins un artefact d'entrée contenant le modèle de l'ensemble de piles dans une `CloudFormationStackSet` action. Vous pouvez inclure d'autres artefacts d'entrée pour les listes de cibles de déploiement, de comptes et de paramètres.

- Nombre d'objets : `1 to 3`
- Description : Vous pouvez inclure des artefacts pour fournir :
  - Le fichier modèle de pile. (Voir le paramètre `TemplatePath`.)
  - Le fichier de paramètres. (Voir le paramètre `Parameters`.)
  - Le fichier des comptes. (Voir le paramètre `DeploymentTargets`.)

Artefacts de sortie

- Nombre d'objets : `0`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Variables de sortie

Si vous configurez cette action, elle produit des variables qui peuvent être référencées par la configuration de l'action d'une action en aval dans le pipeline. Vous configurez une action avec un espace de noms pour rendre ces variables disponibles pour la configuration des actions en aval.

- StackSetId: ID de l'ensemble de piles.
- OperationId: ID de l'opération d'ensemble de piles.

Pour de plus amples informations, veuillez consulter Référence des variables.

Exemple CloudFormationStackSet configuration des actions

Les exemples suivants montrent la configuration de l'CloudFormationStackSetaction.

Exemple de modèle d'autorisations autogérées

L'exemple suivant montre une CloudFormationStackSet action dans laquelle la cible de déploiement saisie est un ID de AWS compte.

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"
}
```

Exemple de modèle d'autorisations gérées par les services

L'exemple suivant montre une CloudFormationStackSet action pour le modèle d'autorisations gérées par les services dans laquelle l'option de déploiement automatique auprès des AWS organisations est activée avec la rétention de la pile.

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"
}
```

L' CloudFormationStackInstances action

Cette action crée de nouvelles instances et déploie des ensembles de piles sur des instances spécifiées. Une instance de pile est une référence à une pile d'un compte de destination dans une région. Une instance de pile peut exister sans pile ; par exemple, si la création de pile échoue, l'instance de pile indique la raison de l'échec de la création de la pile. L'instance de pile est associée à un seul ensemble de piles.

Après la création initiale d'un ensemble de piles, vous pouvez ajouter de nouvelles instances de pile à l'aide de`CloudFormationStackInstances`. Les valeurs des paramètres du modèle peuvent être remplacées au niveau de l'instance de pile lors des opérations de création ou de mise à jour d'instances d'ensemble de piles.

Chaque ensemble de piles possède un modèle et un ensemble de paramètres de modèle. Lorsque vous mettez à jour le modèle ou les paramètres du modèle, vous les mettez à jour pour l'ensemble complet. Tous les statuts de l'instance sont ensuite définis sur `OUTDATED` jusqu'à ce que les modifications soient déployées sur cette instance.

Pour remplacer les valeurs de paramètres sur des instances spécifiques, par exemple, si le modèle contient un paramètre pour `stage` ayant une valeur de`prod`, vous pouvez remplacer la valeur de ce paramètre par ou. `beta` `gamma`

Rubriques

- Type d'action

- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables de sortie
- Exemple de configuration d'action

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `CloudFormationStackInstances`
- Version : `1`

Paramètres de configuration

StackSetName

Obligatoire : oui

Nom à associer à l'ensemble de piles. Ce nom doit être unique dans la région où il a été créé.

Le nom ne peut contenir que des caractères alphanumériques et des tirets. Il doit commencer par un caractère alphabétique et comporter 128 caractères ou moins.

DeploymentTargets

Obligatoire : non

> **Note**
>
> Pour le modèle d'autorisations SERVICE_MANAGED, vous pouvez fournir l'ID racine de l'organisation ou les ID d'unité organisationnelle pour les cibles de déploiement. Pour le modèle d'autorisations SELF_MANAGED, vous ne pouvez fournir que des comptes.

> **Note**
>
> Lorsque ce paramètre est sélectionné, vous devez également sélectionner Régions.

Une liste de AWS comptes ou d'identifiants d'unités organisationnelles où doivent se trouver les instances de stack set created/updated.

- Comptes :

  Vous pouvez fournir des comptes sous forme de liste littérale ou de chemin de fichier :

  - Littéral : entrez les paramètres dans le format de syntaxe abrégé`account_ID,account_ID`, comme indiqué dans l'exemple suivant.

    ```
    111111222222,333333444444
    ```

  - Chemin du fichier : emplacement du fichier contenant une liste de AWS comptes où doivent se trouver les instances de stack set created/updated, saisi au format`InputArtifactName::AccountsFileName`. Si vous utilisez le chemin du fichier pour spécifier des comptes ou OrganizationalUnitIds, le format de fichier doit être au format JSON, comme illustré dans l'exemple suivant.

    ```
    SourceArtifact::accounts.txt
    ```

    L'exemple suivant montre le contenu du fichier pour `accounts.txt` :

    ```
    [
        "111111222222"
    ]
    ```

    L'exemple suivant montre le contenu du fichier `accounts.txt` lorsque plusieurs comptes sont répertoriés :

    ```
    [
        "111111222222","333333444444"
    ]
    ```

- OrganizationalUnitIds:

> **Note**
>
> Ce paramètre est facultatif pour le modèle d'autorisations SERVICE_MANAGED et n'est pas utilisé pour le modèle d'autorisations SELF_MANAGED. Ne l'utilisez pas si vous sélectionnez OrganizationsAutoDeployment.

Les unités AWS organisationnelles dans lesquelles mettre à jour les instances de pile associées.

Vous pouvez fournir des ID d'unité organisationnelle sous forme de liste littérale ou de chemin de fichier.

- Littéral : entrez un tableau de chaînes séparées par des virgules, comme illustré dans l'exemple suivant.

```
ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
```

- Chemin du fichier : emplacement du fichier contenant la liste des instances OrganizationalUnitIds dans lesquelles créer ou mettre à jour des instances d'ensembles de piles. Si vous utilisez le chemin du fichier pour spécifier des comptes ou OrganizationalUnitIds, le format de fichier doit être au format JSON, comme illustré dans l'exemple suivant.

  Entrez le chemin d'accès au fichier au formatInputArtifactName::OrganizationalUnitIdsFileName.

```
SourceArtifact::OU-IDs.txt
```

  L'exemple suivant montre le contenu du fichier pour `OU-IDs.txt` :

```
[
    "ou-examplerootid111-exampleouid111","ou-examplerootid222-exampleouid222"
]
```

Régions

Obligatoire : oui

> **Note**
>
> Lorsque ce paramètre est sélectionné, vous devez également le sélectionner DeploymentTargets.

Liste des AWS régions dans lesquelles les instances d'ensembles de piles sont créées ou mises à jour. Les régions sont mises à jour dans l'ordre dans lequel elles ont été saisies.

Entrez une liste de AWS régions valides au format :`Region1,Region2`, comme illustré dans l'exemple suivant.

```
us-west-2,us-east-1
```

ParameterOverrides

Obligatoire : non

Liste des paramètres d'ensemble de piles que vous souhaitez remplacer dans les instances de pile sélectionnées. Les valeurs de paramètres remplacées sont appliquées à toutes les instances de pile dans les comptes et les régions spécifiés.

Vous pouvez fournir des paramètres sous forme de liste littérale ou de chemin de fichier :

- Vous pouvez saisir des paramètres dans le format de syntaxe abrégé suivant :. `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  Pour plus d'informations sur ces types de données, consultezTypes de données des paramètres du modèle.

  L'exemple suivant montre un paramètre nommé `BucketName` avec la valeur`amzn-s3-demo-source-bucket`.

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ```

  L'exemple suivant montre une entrée avec plusieurs paramètres.

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ```

```
  ParameterKey=Asset1,ParameterValue=true
  ParameterKey=Asset2,ParameterValue=true
```

- Vous pouvez entrer l'emplacement du fichier contenant la liste des remplacements de paramètres de modèle saisis au format`InputArtifactName::ParameterOverridesFileName`, comme illustré dans l'exemple suivant.

  ```
  SourceArtifact::parameter-overrides.txt
  ```

  L'exemple suivant montre le contenu du fichier pour`parameter-overrides.txt`.

  ```
  [
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      },
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      }
  ]
  ```

FailureTolerancePercentage

Obligatoire : non

Le pourcentage de comptes par région pour lesquels cette opération de pile peut échouer avant l' CloudFormation arrêt de l'opération dans cette région. Si l'opération est arrêtée dans une région, CloudFormation ne tente pas l'opération dans les régions suivantes. Lorsque vous calculez le nombre de comptes en fonction du pourcentage spécifié, CloudFormation arrondissez au nombre entier inférieur.

MaxConcurrentPercentage

Obligatoire : non

Pourcentage maximum de comptes sur lesquels effectuer cette opération en même temps. Lorsque vous calculez le nombre de comptes en fonction du pourcentage spécifié, CloudFormation arrondissez au nombre entier inférieur. Si l'arrondissement vers le bas CloudFormation donne zéro, définissez plutôt le nombre comme un. Bien que vous indiquiez

le maximum, pour les déploiements de grande envergure, le nombre réel de comptes traités simultanément peut être inférieur en raison de la limitation des services.

RegionConcurrencyType

Obligatoire : non

Vous pouvez spécifier si l'ensemble de piles doit être déployé de Régions AWS manière séquentielle ou parallèle en configurant le paramètre de déploiement simultané des régions. Lorsque la simultanéité régionale est spécifiée pour déployer des piles sur plusieurs Régions AWS en parallèle, cela peut se traduire par des temps de déploiement globaux plus rapides.

- Parallèle : les déploiements d'ensembles de piles seront effectués en même temps, à condition que les échecs de déploiement d'une région ne dépassent pas une tolérance de défaillance spécifiée.
- Séquentiel : les déploiements des ensembles de piles seront effectués un par un, à condition que les échecs de déploiement d'une région ne dépassent pas une tolérance de défaillance spécifiée. Le déploiement séquentiel est la sélection par défaut.

ConcurrencyMode

Obligatoire : non

Le mode de simultanéité vous permet de choisir le comportement du niveau de concurrence lors des opérations d'ensemble de piles, que ce soit avec une tolérance de défaillance stricte ou souple. La Tolérance stricte aux pannes réduit la vitesse de déploiement en cas de défaillance des opérations d'ensemble de piles, car la simultanéité diminue à chaque défaillance. Soft Failure Tolerance donne la priorité à la vitesse de déploiement tout en tirant parti des capacités CloudFormation de sécurité.

- `STRICT_FAILURE_TOLERANCE`: cette option abaisse dynamiquement le niveau de simultanéité afin de garantir que le nombre de comptes défaillants ne dépasse jamais une certaine tolérance de défaillance. Il s'agit du comportement de par défaut.
- `SOFT_FAILURE_TOLERANCE`: cette option dissocie la tolérance aux pannes de la simultanéité réelle. Cela permet aux opérations d'ensemble de piles de s'exécuter à un niveau de simultanéité défini, quel que soit le nombre d'échecs.

CallAs

Obligatoire : non

> **Note**
>
> Ce paramètre est facultatif pour le modèle `SERVICE_MANAGED` d'autorisations et n'est pas utilisé pour le modèle `SELF_MANAGED` d'autorisations.

Spécifie si vous agissez sur le compte de gestion de l'organisation ou en tant qu'administrateur délégué sur un compte de membre.

> **Note**
>
> Si ce paramètre est défini sur`DELEGATED_ADMIN`, assurez-vous que le rôle IAM du pipeline est `organizations:ListDelegatedAdministrators` autorisé. Sinon, l'action échouera lors de l'exécution avec une erreur similaire à la suivante : `Account used is not a delegated administrator`

- `SELF`: le déploiement de Stack Set utilisera les autorisations gérées par le service lorsque vous êtes connecté au compte de gestion.
- `DELEGATED_ADMIN`: le déploiement de Stack Set utilisera les autorisations gérées par les services lorsque vous êtes connecté à un compte d'administrateur délégué.

Artefacts d'entrée

`CloudFormationStackInstances`peut contenir des artefacts répertoriant les cibles et les paramètres de déploiement.

- Nombre d'objets : `0 to 2`
- Description : En entrée, l'action Stack Set accepte éventuellement des artefacts aux fins suivantes :
  - Fournir le fichier de paramètres à utiliser. (Voir le paramètre `ParameterOverrides`.)
  - Fournir le fichier de comptes cibles à utiliser. (Voir le paramètre `DeploymentTargets`.)

Artefacts de sortie

- Nombre d'objets : `0`

- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Variables de sortie

Lorsque cette action est configurée, elle produit des variables qui peuvent être référencées par la configuration d'action d'une action en aval dans le pipeline. Vous configurez une action avec un espace de noms pour rendre ces variables disponibles pour la configuration des actions en aval.

- StackSetId: ID de l'ensemble de piles.
- OperationId: ID de l'opération de stack set.

Pour de plus amples informations, veuillez consulter Référence des variables.

Exemple de configuration d'action

Les exemples suivants montrent la configuration de l'CloudFormationStackInstancesaction.

Exemple de modèle d'autorisations autogérées

L'exemple suivant montre une CloudFormationStackInstances action dans laquelle la cible de déploiement saisie est un Compte 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"
}
```

Exemple de modèle d'autorisations gérées par les services

L'exemple suivant montre une CloudFormationStackInstances action pour le modèle d'autorisations gérées par les services où la cible de déploiement est l'ID d'une unité organisationnelle de l' AWS organisation. `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"
}
```

Autorisations relatives aux rôles de service : **`CloudFormationStackSet`** action

Pour CloudFormation StackSets les actions, les autorisations minimales suivantes sont requises.

Pour l'`CloudFormationStackSet`action, ajoutez ce qui suit à votre déclaration de politique :

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

```
    "Action": [
        "cloudformation:CreateStackSet",
        "cloudformation:UpdateStackSet",
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:DescribeStackSet",
        "cloudformation:ListStackInstances"
    ],
    "Resource": "resource_ARN"
},
```

Autorisations relatives aux rôles de service : **`CloudFormationStackInstances`** action

Pour l'`CloudFormationStackInstances`action, ajoutez ce qui suit à votre déclaration de politique :

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation"
    ],
    "Resource": "resource_ARN"
},
```

Modèles d'autorisations pour les opérations relatives aux ensembles de piles

Comme il CloudFormation StackSets effectue des opérations sur plusieurs comptes, vous devez définir les autorisations nécessaires sur ces comptes avant de pouvoir créer le stack set. Vous pouvez définir des autorisations par le biais d'autorisations autogérées ou d'autorisations gérées par service.

Avec les autorisations autogérées, vous créez les deux rôles IAM requis par StackSets : un rôle d'administrateur, par exemple, AWSCloudFormationStackSetAdministrationRole dans le compte où vous définissez le stack set, et un rôle d'exécution, tel que AWSCloudFormationStackSetExecutionRole dans chacun des comptes où vous déployez des instances de stack set. À l'aide de ce modèle d'autorisations, StackSets vous pouvez effectuer un

déploiement AWS sur n'importe quel compte dans lequel l'utilisateur est autorisé à créer un rôle IAM. Pour plus d'informations, voir Accorder des autorisations autogérées dans le Guide de l'AWS CloudFormation utilisateur.

> **Note**
>
> Comme il CloudFormation StackSets effectue des opérations sur plusieurs comptes, vous devez définir les autorisations nécessaires sur ces comptes avant de pouvoir créer le stack set.

Grâce aux autorisations gérées par les services, vous pouvez déployer des instances de stack sur des comptes gérés par AWS des organisations. À l'aide de ce modèle d'autorisations, vous n'avez pas à créer les rôles IAM nécessaires, car les rôles IAM sont StackSets créés en votre nom. Avec ce modèle, vous pouvez également activer les déploiements automatiques sur les comptes qui seront ajoutés ultérieurement à l'organisation. Consultez la section Activer l'accès sécurisé auprès AWS des organisations dans le Guide de AWS CloudFormation l'utilisateur.

Types de données des paramètres du modèle

Les paramètres de modèle utilisés dans les opérations d'ensembles de piles incluent les types de données suivants. Pour de plus amples informations, veuillez consulter DescribeStackSet.

ParameterKey

- Description : clé associée au paramètre. Si vous ne spécifiez pas de clé ni de valeur pour un paramètre particulier, AWS CloudFormation utilise la valeur par défaut spécifiée dans le modèle.
- Exemple :

```
"ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket"
```

ParameterValue

- Description : valeur d'entrée associée au paramètre.
- Exemple :

```
"ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket"
```

UsePreviousValue

- Lors d'une mise à jour de pile, utilisez la valeur de paramètre existante que la pile utilise pour une clé de paramètre donnée. Si vous spécifiez`true`, ne spécifiez pas de valeur de paramètre.
- Exemple :

```
"ParameterKey=Asset1,UsePreviousValue=true"
```

Chaque ensemble de piles possède un modèle et un ensemble de paramètres de modèle. Lorsque vous mettez à jour le modèle ou les paramètres du modèle, vous les mettez à jour pour l'ensemble complet. Ensuite, tous les statuts d'instance sont définis sur OBSOLÈTE jusqu'à ce que les modifications soient déployées sur cette instance.

Pour remplacer les valeurs de paramètres sur des instances spécifiques, par exemple, si le modèle contient un paramètre pour `stage` ayant une valeur de`prod`, vous pouvez remplacer la valeur de ce paramètre par ou. `beta` `gamma`

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Types de paramètres — Ce chapitre de référence du Guide de l'AWS CloudFormation utilisateur fournit des descriptions et des exemples supplémentaires pour les paramètres des CloudFormation modèles.
- Meilleures pratiques : pour plus d'informations sur les meilleures pratiques en matière de déploiement de stack sets, consultez https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-bestpractices.html le Guide de AWS CloudFormation l'utilisateur.
- AWS CloudFormation Référence d'API : vous pouvez faire référence aux CloudFormation actions suivantes dans la référence d'AWS CloudFormation API pour plus d'informations sur les paramètres utilisés dans les opérations d'ensemble de piles :

  - L'CreateStackSetaction crée un ensemble de piles.
  - L'UpdateStackSetaction met à jour l'ensemble de piles et les instances de pile associées dans les comptes et les régions spécifiés. Même si l'opération d'ensemble de piles créée par la mise à jour de l'ensemble de piles échoue (complètement ou partiellement, en dessous ou au-dessus d'une tolérance de défaillance spécifiée), l'ensemble de piles est mis à jour avec ces

modifications. Les CreateStackInstances appels suivants sur l'ensemble de piles spécifié utilisent le jeu de piles mis à jour.

- L'CreateStackInstancesaction crée une instance de pile pour toutes les régions spécifiées au sein de tous les comptes spécifiés sur un modèle d'autorisation autogéré, ou dans toutes les cibles de déploiement spécifiées sur un modèle d'autorisation géré par les services. Vous pouvez modifier les paramètres des instances créées par cette action. Si les instances existent déjà, CreateStackInstances appels UpdateStackInstances avec les mêmes paramètres d'entrée. Lorsque vous utilisez cette action pour créer des instances, cela ne modifie pas le statut des autres instances de pile.
- L'UpdateStackInstancesaction met à jour les instances de pile avec la pile définie pour toutes les régions spécifiées dans tous les comptes spécifiés sur un modèle d'autorisation autogéré, ou dans toutes les cibles de déploiement spécifiées sur un modèle d'autorisation géré par les services. Vous pouvez modifier les paramètres des instances mises à jour par cette action. Lorsque vous utilisez cette action pour mettre à jour un sous-ensemble d'instances, cela ne modifie pas le statut des autres instances de pile.
- L'DescribeStackSetOperationaction renvoie la description de l'opération d'ensemble de piles spécifiée.
- L'DescribeStackSetaction renvoie la description de l'ensemble de piles spécifié.

AWS CodeBuild créer et tester une référence d'action

Vous permet d'exécuter des générations et des tests dans le cadre de votre pipeline. Lorsque vous exécutez une action de CodeBuild génération ou de test, les commandes spécifiées dans la spécification de construction sont exécutées à l'intérieur d'un CodeBuild conteneur. Tous les artefacts qui sont spécifiés comme artefacts d'entrée pour une CodeBuild action sont disponibles dans le conteneur qui exécute les commandes. CodeBuild peut fournir une action de génération ou de test. Pour plus d'informations, consultez le Guide de l'utilisateur AWS CodeBuild.

Lorsque vous utilisez l' CodePipeline assistant de la console pour créer un projet de génération, le projet de CodeBuild génération indique que c'est le fournisseur source CodePipeline. Lorsque vous créez un projet de génération dans la CodeBuild console, vous ne pouvez pas le spécifier CodePipeline en tant que fournisseur source, mais l'ajout de l'action de génération à votre pipeline ajuste la source dans la CodeBuild console. Pour plus d'informations, consultez ProjectSource dans la Référence d'API AWS CodeBuild .

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables de sortie
- Autorisations relatives aux rôles de service : CodeBuild action
- Déclaration d'action (CodeBuild exemple)
- Consultez aussi

Type d'action

- Catégorie : `Build` ou `Test`
- Propriétaire : `AWS`
- Fournisseur : `CodeBuild`
- Version : `1`

Paramètres de configuration

ProjectName

Obligatoire : oui

`ProjectName`est le nom du projet de construction dans CodeBuild.

PrimarySource

Obligatoire : Conditionnelle

La valeur du `PrimarySource` paramètre doit être le nom de l'un des artefacts d'entrée de l'action. CodeBuild recherche le fichier buildspec et exécute les commandes buildspec dans le répertoire qui contient la version décompressée de cet artefact.

Ce paramètre est obligatoire si plusieurs artefacts d'entrée sont spécifiés pour une action CodeBuild . Lorsqu'il n'y a qu'un seul artefact source pour l'action, l'artefact `PrimarySource` correspond par défaut à cet artefact.

BatchEnabled

Obligatoire : non

La valeur booléenne du `BatchEnabled` paramètre permet à l'action d'exécuter plusieurs builds lors de la même exécution de build.

Lorsque cette option est activée, elle est disponible. `CombineArtifacts`

Pour des exemples de pipeline dans lesquels la génération par lots est activée, consultez la section CodePipeline Intégration avec CodeBuild et génération par lots.

BuildspecOverride

Obligatoire : non

Une définition de spécification de construction en ligne ou une déclaration de fichier de spécification de construction qui remplace la dernière définition définie dans le projet de construction, pour cette version uniquement. La spécification de construction définie sur le projet n'est pas modifiée.

Si cette valeur est définie, il peut s'agir de l'une des valeurs suivantes :

- Une définition de spécification de construction en ligne. Pour plus d'informations, consultez la référence de syntaxe sur Buildspec syntax.
- Le chemin d'accès à un autre fichier buildspec par rapport à la valeur de la variable d'`CODEBUILD_SRC_DIR`environnement intégrée ou le chemin d'accès à un compartiment S3. Le compartiment doit se trouver dans le même emplacement Région AWS que le projet de construction. Spécifiez le fichier buildspec à l'aide de son nom ARN (par exemple, `arn:aws:s3:::my-codebuild-sample2/buildspec.yml`). Si cette valeur n'est pas fournie ou est définie sur une chaîne vide, le code source doit contenir un fichier buildspec dans son répertoire racine. Pour plus d'informations sur l'ajout d'un chemin, consultez la section Nom du fichier Buildspec et emplacement de stockage.

> **Note**
>
> Comme cette propriété vous permet de modifier les commandes de génération qui seront exécutées dans le conteneur, vous devez noter qu'un principal IAM capable d'appeler cette API et de définir ce paramètre peut remplacer les paramètres par défaut. De plus, nous vous recommandons d'utiliser un emplacement de spécification de construction fiable, tel qu'un fichier de votre référentiel source ou un compartiment Amazon S3.

CombineArtifacts

Obligatoire : non

La valeur booléenne du `CombineArtifacts` paramètre combine tous les artefacts de génération d'une génération par lots dans un seul fichier d'artefacts pour l'action de génération.

Pour utiliser cette option, le `BatchEnabled` paramètre doit être activé.

EnvironmentVariables

Obligatoire : non

La valeur de ce paramètre est utilisée pour définir des variables d'environnement pour l'action CodeBuild dans votre pipeline. La valeur du paramètre `EnvironmentVariables` prend la forme d'un tableau JSON d'objets variables d'environnement. Consultez l'exemple de paramètre dans Déclaration d'action (CodeBuild exemple).

Chaque objet comporte trois parties, qui sont toutes des chaînes :

- `name` : Nom ou clé de la variable d'environnement.
- `value` : Valeur de la variable d'environnement. Lorsque vous utilisez le `SECRETS_MANAGER` type `PARAMETER_STORE` ou, cette valeur doit être le nom d'un paramètre que vous avez déjà stocké dans le magasin de paramètres de AWS Systems Manager ou un secret que vous avez déjà enregistré dans AWS Secrets Manager, respectivement.

  > **Note**
  >
  > Nous déconseillons vivement l'utilisation de variables d'environnement pour stocker des valeurs sensibles, en particulier des AWS informations d'identification. Lorsque vous utilisez la CodeBuild console ou l' AWS interface de ligne de commande, les variables d'environnement sont affichées en texte brut. Pour les valeurs sensibles, nous vous recommandons d'utiliser plutôt le type `SECRETS_MANAGER`.

- `type` : (facultatif) Type de la variable d'environnement. Les valeurs valides sont `PARAMETER_STORE`, `SECRETS_MANAGER` ou `PLAINTEXT`. Si rien spécifié, la valeur par défaut est `PLAINTEXT`.

> **Note**
>
> Lorsque vous entrez la configuration `namevalue`, et `type` pour vos variables d'environnement, en particulier si la variable d'environnement contient la syntaxe de la

variable de CodePipeline sortie, ne dépassez pas la limite de 1 000 caractères pour le champ de valeur de la configuration. Une erreur de validation est renvoyée lorsque cette limite est dépassée.

Pour plus d'informations, consultez EnvironmentVariable la référence de AWS CodeBuild l'API. Pour un exemple CodeBuild d'action avec une variable d'environnement dont la résolution est le nom de la GitHub branche, consultezExemple : utilisation d'une BranchName variable avec des variables d' CodeBuild environnement.

ServiceRoleArnOverride

Obligatoire : non

Un ARN de rôle IAM qui est CodeBuild utilisé comme rôle de service pour les builds déclenchés par cette action, remplaçant ainsi le rôle de service par défaut configuré sur le CodeBuild projet. En cas d'absence, le CodeBuild projet utilise son rôle de service par défaut.

Le rôle spécifié doit disposer d'une politique de confiance qui permet au `codebuild.amazonaws.com` service de l'assumer. L'identité utilisée pour créer ou mettre à jour le pipeline et le rôle de service de pipeline doivent disposer d'une `iam:PassRole` autorisation pour le rôle spécifié. Pour un exemple de politique, consultez Autorisations relatives aux rôles de service : CodeBuild action.

En tant que meilleure pratique en matière de sécurité, lorsqu'un CodeBuild projet est partagé entre plusieurs pipelines, nous vous recommandons de spécifier un rôle délimité pour chaque pipeline afin de respecter le principe du moindre privilège. Pour les pipelines utilisant des actions de CodeConnections source, consultez un Ajoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com exemple de définition de la portée des autorisations de connexion du rôle à un référentiel spécifique.

> **Note**
>
> Le rôle doit figurer dans le même AWS compte que le CodeBuild projet.

Artefacts d'entrée

- Nombre d'objets : `1 to 5`

- Description : CodeBuild recherche le fichier buildspec et exécute les commandes buildspec depuis le répertoire de l'artefact source principal. Lorsqu'une seule source d'entrée est spécifiée ou lorsque plusieurs sources d'entrée sont spécifiées pour l' CodeBuild action, l'artefact unique, ou l'artefact principal dans le cas de plusieurs sources d'entrée, doit être défini à l'aide du paramètre de configuration de l'`PrimarySource`action dans. CodePipeline

  Chaque artefact d'entrée est extrait dans son propre répertoire, dont les emplacements sont stockés dans des variables d'environnement. Le répertoire de l'artefact source principal est mis à disposition avec `$CODEBUILD_SRC_DIR`. Les répertoires de tous les autres artefacts d'entrée sont mis à disposition avec `$CODEBUILD_SRC_DIR_yourInputArtifactName`.

  > **Note**
  >
  > L'artefact configuré dans votre CodeBuild projet devient l'artefact d'entrée utilisé par l' CodeBuild action dans votre pipeline.

Artefacts de sortie

- Nombre d'objets : `0 to 5`
- Description : ils peuvent être utilisés pour mettre les artefacts définis dans le fichier CodeBuild buildspec à la disposition des actions ultérieures du pipeline. Lorsqu'un seul artefact de sortie est défini, cet artefact peut être défini directement dans la `artifacts` section du fichier buildspec. Lorsque plusieurs artefacts de sortie sont spécifiés, tous les artefacts référencés doivent être définis comme artefacts secondaires dans le fichier buildspec. Les noms des artefacts de sortie CodePipeline doivent correspondre aux identificateurs d'artefacts contenus dans le fichier buildspec.

  > **Note**
  >
  > L'artefact configuré dans votre CodeBuild projet devient l'artefact CodePipeline d'entrée dans votre action de pipeline.

  Si le `CombineArtifacts` paramètre est sélectionné pour les constructions par lots, l'emplacement des artefacts en sortie contient les artefacts combinés de plusieurs builds exécutés lors de la même exécution.

Variables de sortie

Cette action produira en tant que variables toutes les variables d'environnement ayant été exportées dans le cadre de la génération. Pour plus de détails sur la façon d'exporter des variables d'environnement, consultez EnvironmentVariable le Guide des AWS CodeBuild API.

Pour plus d'informations sur l'utilisation des variables d' CodeBuild environnement dans CodePipeline, consultez les exemples figurant dansCodeBuild variables de sortie d'action. Pour obtenir la liste des variables d'environnement que vous pouvez utiliser CodeBuild, consultez la section Variables d'environnement dans les environnements de génération du Guide de AWS CodeBuild l'utilisateur.

Autorisations relatives aux rôles de service : CodeBuild action

Pour obtenir de l' CodeBuild aide, ajoutez ce qui suit à votre déclaration de politique :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "codebuild:BatchGetBuilds",
                "codebuild:StartBuild",
                "codebuild:BatchGetBuildBatches",
                "codebuild:StartBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:*:111122223333:project/[[ProjectName]]"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Si vous utilisez le `ServiceRoleArnOverride` paramètre, ajoutez l'`iam:PassRole`autorisation suivante pour CodePipeline permettre de transmettre le rôle à CodeBuild :

```
{
```

```
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::111122223333:role/your-codebuild-override-role",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": "codebuild.amazonaws.com"
        }
    }
}
```

Déclaration d'action (CodeBuild exemple)

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"
                }
            ]
        }
    ]
}
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- AWS CodeBuild Guide de l'utilisateur — Pour un exemple de pipeline avec une CodeBuild action, voir Utiliser CodePipeline avec CodeBuild pour tester le code et exécuter des builds. Pour des exemples de projets comportant plusieurs CodeBuild artefacts d'entrée et de sortie, voir CodePipeline Intégration avec CodeBuild plusieurs sources d'entrée et échantillons d'artefacts de sortie et Exemples de sources d'entrée et d'artefacts de sortie multiples.
- Tutoriel : Créez un pipeline qui crée et teste votre application Android avec AWS Device Farm— Ce didacticiel fournit un exemple de fichier buildspec et un exemple d'application pour créer un pipeline avec une GitHub source qui crée et teste une application Android avec CodeBuild et. AWS Device Farm
- Référence de spécification de génération pour CodeBuild : cette rubrique de référence fournit des définitions et des exemples permettant de comprendre les fichiers CodeBuild buildspec. Pour obtenir la liste des variables d'environnement que vous pouvez utiliser CodeBuild, consultez la section Variables d'environnement dans les environnements de génération du Guide de AWS CodeBuild l'utilisateur.

AWS CodePipeline invoquer la référence de l'action

Vous utilisez une action d' CodePipeline invocation pour simplifier le déclenchement des exécutions de pipeline en aval et la transmission de variables de pipeline et de révisions de source entre les pipelines.

> **Note**
>
> Cette action n'est prise en charge que pour les pipelines de type V2.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie

- Autorisations relatives à la politique de rôle de service pour l'action d' CodePipeline invocation
- Déclaration d'action
- Consultez aussi

Type d'action

- Catégorie : `Invoke`
- Propriétaire : `AWS`
- Fournisseur : `CodePipeline`
- Version : `1`

Paramètres de configuration

PipelineName

Obligatoire : oui

Le nom du pipeline qui, lors de son exécution, démarrera le pipeline cible actuel. Vous devez déjà avoir créé le pipeline d'appel. L'action démarre le pipeline `s3-pipeline-test` (cible) lorsque le pipeline (d'appel) nommé `my-s3-pipeline` démarre une exécution.

SourceRevisions

Obligatoire : non

Les révisions de source que vous souhaitez que le pipeline cible utilise lorsqu'il est démarré par le pipeline d'appel. Par exemple, une action source S3 fournit des variables de sortie telles que l'ID de version S3 et la clé d'objet. Vous pouvez spécifier une valeur de révision à utiliser lorsque le pipeline est invoqué.

Pour la CLI, vous spécifiez les révisions de source sous la forme d'une chaîne JSON sérialisée. Pour plus d'informations sur l'utilisation des remplacements de révision de source, consultez SourceRevisionOverride le Guide des CodePipeline API.

Le mappage utilise un format de chaîne, comme illustré dans l'exemple suivant :

```
[{"actionName":"Source","revisionType":"S3_OBJECT_VERSION_ID","revision
```

```
Value":"zq8mjNEXAMPLE"}]
```

Variables

Obligatoire : non

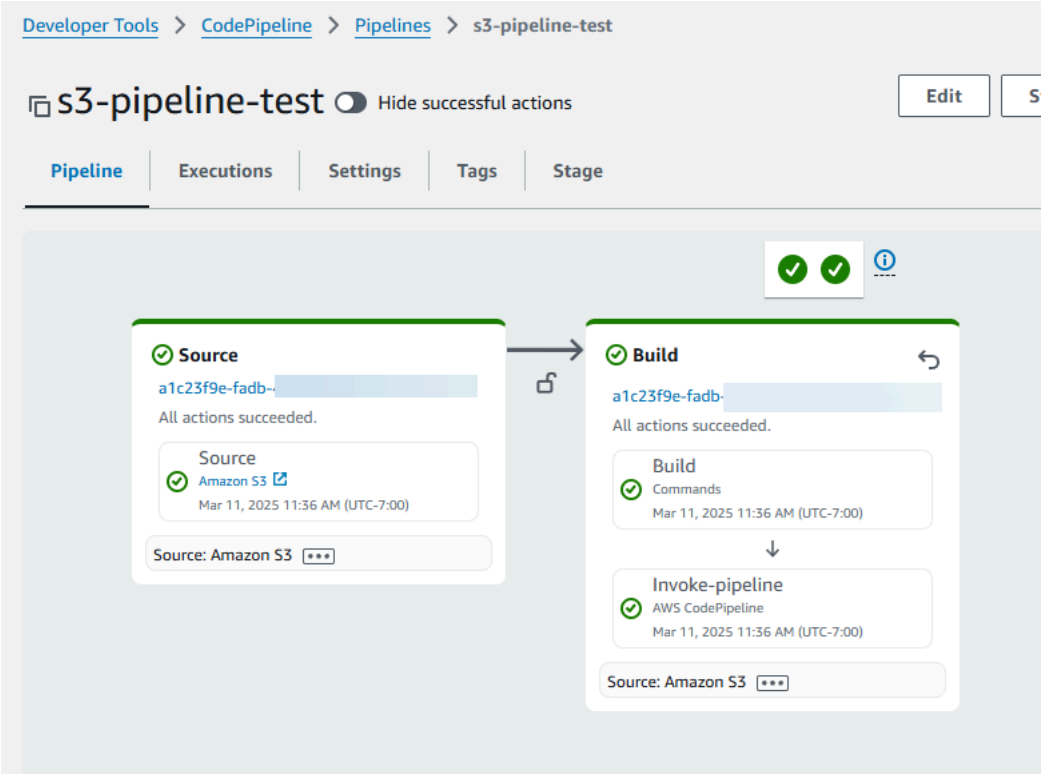
Les noms et les valeurs des variables que vous souhaitez que l'action prenne en charge.

Pour la CLI, vous spécifiez les variables sous la forme d'une chaîne JSON sérialisée. Pour plus d'informations sur l'utilisation des variables de pipeline, consultez PipelineVariable le Guide des CodePipeline API.

Le mappage utilise un format de chaîne, comme illustré dans l'exemple suivant :

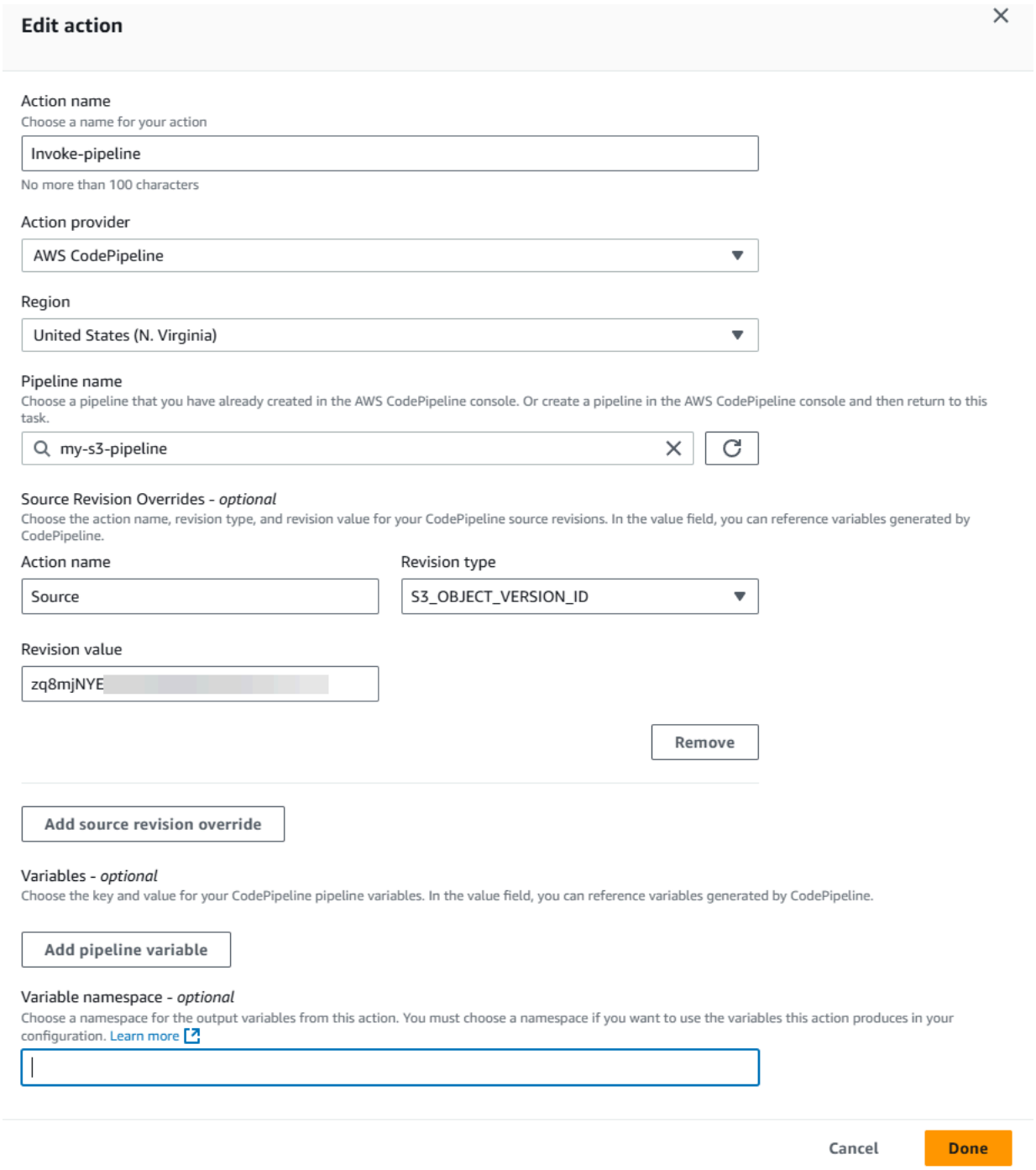
```
[{"name":"VAR1","value":"VALUE1"}]
```

L'image suivante montre un exemple de l'action ajoutée à un pipeline dans la console.



L'image suivante montre un exemple de page d'édition pour l'action. Dans l'exemple suivant, le pipeline nommé `s3-pipeline-test` possède une action d'appel de pipeline configurée comme indiqué pour la console. L'action démarre le `s3-pipeline-test` pipeline lorsque le pipeline nommé `my-s3-pipeline` termine une exécution. L'exemple montre que le remplacement de la révision

de la source pour le code source S3_OBJECT_VERSION_ID est remplacé par la valeur de révision spécifiée de. `zq8mjNYEexample`



Artefacts d'entrée

- Nombre d'objets : `0`
- Description : Les artefacts d'entrée ne s'appliquent pas à ce type d'action.

Artefacts de sortie

- Nombre d'objets : `0`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Autorisations relatives à la politique de rôle de service pour l'action d' CodePipeline invocation

Lors de l' CodePipeline exécution de l'action, la politique de rôle de CodePipeline service nécessite l'`codepipeline:StartPipelineExecution`autorisation, limitée de manière appropriée à l'ARN de la ressource du pipeline afin de conserver l'accès avec le moins de privilèges possible.

```
{
           "Sid": "StatementForPipelineInvokeAction",
           "Effect": "Allow",
           "Action": "codepipeline:StartPipelineExecution",
           "Resource": [
               "arn:aws:codepipeline:{{region}}:{{AccountId}}:{{pipelineName}}"
           ]
       }
```

Déclaration d'action

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\"}]"
    }
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Démarrer un pipeline avec un remplacement de révision de la source— Cette section décrit le démarrage d'un pipeline avec des révisions de source manuellement ou via le transformateur d'entrée d' EventBridge événements.

CodeCommit référence de l'action source

Démarre le pipeline lorsqu'un nouveau commit est effectué sur le CodeCommit référentiel et la branche configurés.

Si vous utilisez la console pour créer ou modifier le pipeline, CodePipeline crée une EventBridge règle qui démarre votre pipeline lorsqu'une modification se produit dans le référentiel.

> **Note**
>
> Pour Amazon ECR, Amazon S3 ou les CodeCommit sources, vous pouvez également créer un remplacement de source à l'aide d'une entrée de transformation en entrée

EventBridge pour utiliser l'`revisionValue`entrée de votre événement de pipeline, où le `revisionValue` est dérivé de la variable d'événement source pour votre clé d'objet, votre commit ou votre ID d'image. Pour plus d'informations, consultez l'étape facultative pour la saisie de la transformation d'entrée incluse dans les procédures sous Actions et ressources relatives aux sources Amazon ECR EventBridgeConnexion aux actions de source Amazon S3 avec une source activée pour les événements, ouCodeCommit actions à la source et EventBridge.

Vous devez déjà avoir créé un CodeCommit référentiel avant de connecter le pipeline par le biais d'une CodeCommit action.

Une fois qu'une modification de code est détectée, vous disposez des options suivantes pour transmettre le code aux actions suivantes :

- Par défaut : configure l'action CodeCommit source pour générer un fichier ZIP contenant une copie superficielle de votre commit.
- Clone complet : configure l'action source pour générer une référence URL Git vers le référentiel pour les actions suivantes.

  Actuellement, la référence URL Git ne peut être utilisée que par des CodeBuild actions en aval pour cloner le dépôt et les métadonnées Git associées. Toute tentative de transmettre une référence d'URL Git à des CodeBuild non-actions entraîne une erreur.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables de sortie
- Autorisations relatives aux rôles de service : CodeCommit action
- Exemple de configuration d'action
- Consultez aussi

Type d'action

- Catégorie : `Source`
- Propriétaire : `AWS`
- Fournisseur : `CodeCommit`
- Version : `1`

Paramètres de configuration

RepositoryName

Obligatoire : oui

Nom du référentiel où les modifications de la source doivent être détectées.

BranchName

Obligatoire : oui

Nom de la branche où les modifications de la source doivent être détectées.

PollForSourceChanges

Obligatoire : non

`PollForSourceChanges`contrôle si CodePipeline les modifications apportées à la source sont interrogées dans le CodeCommit référentiel. Nous vous recommandons plutôt d'utiliser CloudWatch Events pour détecter les modifications de source. Pour plus d'informations sur la configuration CloudWatch des événements, consultez Migrer les pipelines de sondage (CodeCommit source) (CLI) ouMigrer les pipelines de sondage (CodeCommit source) (CloudFormation modèle).

> **Important**
>
> Si vous avez l'intention de configurer une règle d' CloudWatch événements, vous devez la définir `PollForSourceChanges` pour `false` éviter les exécutions de pipeline en double.

Valeurs valides pour ce paramètre :

- `true`: si cette option est définie, CodePipeline interroge votre référentiel pour détecter les modifications apportées à la source.

  > **Note**
  >
  > Si vous l'omettez`PollForSourceChanges`, la CodePipeline valeur par défaut est interrogée dans votre référentiel pour détecter les modifications apportées à la source. Ce comportement est le même que si `PollForSourceChanges` est inclus et défini sur `true`.

- `false`: Si cette option est définie, CodePipeline n'interroge pas votre référentiel pour détecter les modifications apportées à la source. Utilisez ce paramètre si vous avez l'intention de configurer une règle d' CloudWatch événements pour détecter les modifications de source.

OutputArtifactFormat

Obligatoire : non

Format d'artefact de sortie. Les valeurs peuvent être soit, `CODEBUILD_CLONE_REF` soit`CODE_ZIP`. Si aucune valeur n'est spécifiée, la valeur par défaut est `CODE_ZIP`.

> **Important**
>
> L'option `CODEBUILD_CLONE_REF` ne peut être utilisée que par des actions CodeBuild en aval.
> Si vous choisissez cette option, vous devez ajouter l'`codecommit:GitPull`autorisation à votre rôle de CodeBuild service, comme indiqué dansAjouter CodeBuild GitClone des autorisations pour les actions CodeCommit source. Vous devez également ajouter l'`codecommit:GetRepository`autorisation à votre rôle CodePipeline de service, comme indiqué dansAjouter des autorisations au rôle CodePipeline de service. Pour un didacticiel qui explique comment utiliser l'option de clonage complet, consultezTutoriel : Utiliser un clone complet avec une source de CodeCommit pipeline.

Artefacts d'entrée

- Nombre d'objets : `0`
- Description : Les artefacts d'entrée ne s'appliquent pas à ce type d'action.

Artefacts de sortie

- Nombre d'objets : 1
- Description : l'artefact de sortie de cette action est un fichier ZIP qui regroupe le contenu du référentiel et de la branche configurés au moment de la validation spécifiée comme révision source pour l'exécution du pipeline. Les artefacts générés à partir du référentiel sont les artefacts de sortie de l' CodeCommit action. L'ID de validation du code source est affiché en CodePipeline tant que révision source pour l'exécution du pipeline déclenchée.

Variables de sortie

Lorsque cette action est configurée, elle produit des variables qui peuvent être référencées par la configuration d'action d'une action en aval dans le pipeline. Cette action produit des variables qui peuvent être visualisées en tant que variables de sortie, même si l'action n'a pas d'espace de noms. Vous configurez une action avec un espace de noms pour rendre ces variables disponibles pour la configuration des actions en aval.

Pour de plus amples informations, veuillez consulter Référence des variables.

CommitId

L'ID de CodeCommit validation qui a déclenché l'exécution du pipeline. Les ID de validation sont le SHA complet de la validation.

CommitMessage

Message de description, le cas échéant, associé à la validation ayant déclenché l'exécution du pipeline.

RepositoryName

Le nom du CodeCommit référentiel dans lequel le commit qui a déclenché le pipeline a été effectué.

BranchName

Nom de la branche du CodeCommit référentiel dans lequel la modification de source a été effectuée.

AuthorDate

Date à laquelle la validation a été créée, au format horodatage.

CommitterDate

Date à laquelle la validation a été validée, au format horodatage.

Autorisations relatives aux rôles de service : CodeCommit action

Lors de l' CodePipeline exécution de l'action, la politique de rôle de CodePipeline service nécessite les autorisations suivantes, dont la portée est appropriée jusqu'à l'ARN de la ressource du pipeline afin de conserver l'accès avec le moins de privilèges possible. Par exemple, ajoutez ce qui suit à votre déclaration de politique :

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]]"
            ]
        }
    ]
}
```

Exemple de configuration d'action

Exemple de format d'artefact de sortie par défaut

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"
}
```

Exemple de format d'artefact de sortie de clonage complet

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"
}
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Tutoriel : Création d'un pipeline simple (CodeCommit référentiel)— Ce didacticiel fournit un exemple de AppSpec fichier et un exemple d' CodeDeploy application et de groupe de déploiement. Utilisez ce didacticiel pour créer un pipeline avec une CodeCommit source qui se déploie sur les instances Amazon EC2.

AWS CodeDeploy référence d'action de déploiement

Vous utilisez une AWS CodeDeploy action pour déployer le code de l'application dans votre parc de déploiement. Votre flotte de déploiement peut être composée d'instances Amazon EC2, d'instances sur site ou des deux.

> **Note**
>
> Cette rubrique de référence décrit l'action de CodeDeploy déploiement CodePipeline lorsque la plate-forme de déploiement est Amazon EC2. Pour obtenir des informations de référence sur Amazon Elastic Container Service pour les actions de CodeDeploy blue/green déploiement dans CodePipeline, consultezAmazon Elastic Container Service et référence des actions de déploiement CodeDeploy bleu-vert.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Autorisations relatives aux rôles de service : AWS CodeDeploy action
- Déclaration d'action
- Consultez aussi

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `CodeDeploy`
- Version : `1`

Paramètres de configuration

ApplicationName

Obligatoire : oui

Le nom de l'application que vous avez créée dans CodeDeploy.

DeploymentGroupName

Obligatoire : oui

Le groupe de déploiement que vous avez créé dans CodeDeploy.

Artefacts d'entrée

- Nombre d'objets : 1
- Description : Le AppSpec fichier qui CodeDeploy permet de déterminer :

- Ce qu'il faut installer sur vos instances à partir de la révision de votre application dans Amazon S3 ou GitHub.
- Quels hooks d'événement de cycle de vie exécuter en réponse à des événements de cycle de vie du déploiement.

Pour plus d'informations sur le AppSpec fichier, consultez la référence du CodeDeploy AppSpec fichier.

Artefacts de sortie

- Nombre d'objets : `0`
- Description : les artefacts de sortie ne s'appliquent pas à ce type d'action.

Autorisations relatives aux rôles de service : AWS CodeDeploy action

Pour obtenir de l' AWS CodeDeploy aide, ajoutez ce qui suit à votre déclaration de politique :

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": [
                "*"
            ]
        }
    ]
}
```

Déclaration d'action

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"
        }
    ]
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Didacticiel : Création d'un pipeline simple (compartiment S3)— Ce didacticiel explique comment créer un compartiment source, des instances EC2 et des CodeDeploy ressources pour déployer

un exemple d'application. Vous créez ensuite votre pipeline à l'aide d'une action de CodeDeploy déploiement qui déploie le code conservé dans votre compartiment S3 vers votre instance Amazon EC2.

- Tutoriel : Création d'un pipeline simple (CodeCommit référentiel)— Ce didacticiel explique comment créer votre référentiel CodeCommit source, vos instances EC2 et les CodeDeploy ressources nécessaires au déploiement d'un exemple d'application. Vous créez ensuite votre pipeline à l'aide d'une action de CodeDeploy déploiement qui déploie le code de votre CodeCommit référentiel vers votre instance Amazon EC2.
- CodeDeploy AppSpec Référence de fichier — Ce chapitre de référence du guide de l'AWS CodeDeploy utilisateur fournit des informations de référence et des exemples de CodeDeploy AppSpec fichiers.

CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées

Les actions source pour les connexions sont prises en charge par AWS CodeConnections. AWS CodeConnections vous permet de créer et de gérer des connexions entre AWS des ressources et des référentiels tiers tels que GitHub. Démarre un pipeline lorsqu'un nouveau commit est effectué sur un référentiel de code source tiers. L'action source récupère les modifications de code lorsqu'un pipeline est exécuté manuellement ou lorsqu'un événement webhook est envoyé par le fournisseur source.

Vous pouvez configurer des actions dans votre pipeline pour utiliser une configuration Git qui vous permet de démarrer votre pipeline avec des déclencheurs. Pour configurer la configuration des déclencheurs du pipeline afin de filtrer avec des déclencheurs, voir plus de détails dansAjouter un déclencheur avec des types d'événements de type « code push » ou « pull request ».

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans les régions Asie-Pacifique (Hong Kong), Asie-Pacifique (Hyderabad), Asie-Pacifique (Jakarta), Asie-Pacifique (Melbourne), Asie-Pacifique (Osaka), Afrique (Le Cap), Moyen-Orient (Bahreïn), Moyen-Orient (Émirats arabes unis), Europe (Espagne), Europe (Zurich), Israël (Tel Aviv) ou (). AWS GovCloud US-West Pour faire référence aux autres actions disponibles, voirIntégrations de produits et de services avec CodePipeline. Pour les considérations relatives à cette action dans la région Europe

(Milan), voir la note dansCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

Les connexions peuvent associer vos AWS ressources aux référentiels tiers suivants :

- Bitbucket Cloud (via l'option fournisseur Bitbucket dans la CodePipeline console ou le `Bitbucket` fournisseur dans la CLI)

  > **Note**
  >
  > Vous pouvez créer des connexions à un référentiel Bitbucket Cloud. Les types de fournisseurs Bitbucket installés, tels que Bitbucket Server, ne sont pas pris en charge.

-

  > **Note**
  >
  > Si vous utilisez un espace de travail Bitbucket, vous devez disposer d'un accès administrateur pour créer la connexion.

- GitHub et GitHub Enterprise Cloud (via l'option fournisseur GitHub (via GitHub l'application) dans la CodePipeline console ou le `GitHub` fournisseur dans la CLI)

  > **Note**
  >
  > Si votre référentiel se trouve dans une GitHub organisation, vous devez être le propriétaire de l'organisation pour créer la connexion. Si vous utilisez un référentiel qui n'appartient pas à une organisation, vous devez en être le propriétaire.

- GitHub Serveur d'entreprise (via l'option fournisseur de serveur d'GitHub entreprise dans la CodePipeline console ou le `GitHub Enterprise Server` fournisseur dans la CLI)
- GitLab.com (via l'option GitLabprovider dans la CodePipeline console ou le `GitLab` provider dans la CLI)

  > **Note**
  >
  > Vous pouvez créer des connexions à un référentiel dans lequel vous avez le rôle de propriétaire GitLab, puis la connexion peut être utilisée avec le référentiel avec des

ressources telles que CodePipeline. Pour les référentiels dans des groupes, il n'est pas nécessaire d'être le propriétaire du groupe.

- Self-managed installation pour GitLab (Enterprise Edition ou Community Edition) (via l'option fournisseur GitLab autogéré dans la CodePipeline console ou le `GitLabSelfManaged` fournisseur dans la CLI)

> **Note**
>
> Chaque connexion prend en charge tous les référentiels que vous avez auprès de ce fournisseur. Il vous suffit de créer une nouvelle connexion pour chaque type de fournisseur.

Les connexions permettent à votre pipeline de détecter les modifications de source via l'application d'installation du fournisseur tiers. Par exemple, les webhooks sont utilisés pour s'abonner à des types d' GitHub événements et peuvent être installés sur une organisation, un référentiel ou une GitHub application. Votre connexion installe un webhook de référentiel sur votre GitHub application qui s'abonne aux événements de type GitHub push.

Une fois qu'une modification de code est détectée, vous disposez des options suivantes pour transmettre le code aux actions suivantes :

- Par défaut : comme les autres actions CodePipeline source existantes, `CodeStarSourceConnection` vous pouvez générer un fichier ZIP avec une copie superficielle de votre commit.
- Clone complet : `CodeStarSourceConnection` peut également être configuré pour générer une référence URL vers le dépôt pour les actions suivantes.

  Actuellement, la référence d'URL Git ne peut être utilisée que par des CodeBuild actions en aval pour cloner le dépôt et les métadonnées Git associées. Toute tentative de transmission d'une référence d'URL Git à CodeBuild des non-actions entraîne une erreur.

CodePipeline vous invite à ajouter l'application d'installation AWS Connector à votre compte tiers lorsque vous créez une connexion. Vous devez déjà avoir créé votre compte de fournisseur tiers et votre référentiel avant de pouvoir vous connecter via l'`CodeStarSourceConnection`action.

> **Note**
>
> Pour créer ou associer une politique à votre rôle avec les autorisations requises pour utiliser les AWS CodeStar connexions, consultez la section Référence des autorisations des connexions. En fonction de la date de création de votre rôle de CodePipeline service, vous devrez peut-être mettre à jour ses autorisations pour prendre en charge AWS CodeStar les connexions. Pour obtenir des instructions, consultez Ajouter des autorisations au rôle CodePipeline de service.

> **Note**
>
> Pour utiliser les connexions en Europe (Milan) Région AWS, vous devez :
>
> 1. Installer une Region-specific application
> 2. Activer la région
>
> Cette Region-specific application prend en charge les connexions dans la région Europe (Milan). Elle est publiée sur le site du fournisseur tiers et est distincte de l'application existante qui prend en charge les connexions pour d'autres régions. En installant cette application, vous autorisez les fournisseurs tiers à partager vos données avec le service pour cette région uniquement et vous pouvez révoquer les autorisations à tout moment en désinstallant l'application.
> Le service ne traitera ni ne stockera vos données à moins que vous n'activiez la région. En activant cette région, vous autorisez notre service à traiter et à stocker vos données. Même si la Région n'est pas activée, les fournisseurs tiers peuvent toujours partager vos données avec notre service si l' Region-specific application reste installée. Assurez-vous donc de désinstaller l'application une fois que vous aurez désactivé la Région. Pour plus d'informations, consultez Activer une région.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie

- Variables de sortie
- Autorisations relatives aux rôles de service : AWS CodeConnections action
- Déclaration d'action
- Installation de l'application d'installation et création d'une connexion
- Consultez aussi

Type d'action

- Catégorie : `Source`
- Propriétaire : `AWS`
- Fournisseur : `CodeStarSourceConnection`
- Version : `1`

Paramètres de configuration

ConnectionArn

Obligatoire : oui

ARN de connexion qui est configuré et authentifié pour le fournisseur de source.

FullRepositoryId

Obligatoire : oui

Propriétaire et nom du référentiel où les modifications de la source doivent être détectées.

Exemple : `some-user/my-repo`

> **Important**
>
> Vous devez conserver la bonne majuscule pour la FullRepositoryIdvaleur. Par exemple, si votre nom d'utilisateur est `some-user` et que le nom du dépôt l'est`My-Repo`, la valeur recommandée FullRepositoryIdest`some-user/My-Repo`.

BranchName

Obligatoire : oui

Nom de la branche où les modifications de la source doivent être détectées.

OutputArtifactFormat

Obligatoire : non

Spécifie le format de l'artefact de sortie. Peut avoir la valeur `CODEBUILD_CLONE_REF` ou `CODE_ZIP`. Si aucune valeur n'est spécifiée, la valeur par défaut est `CODE_ZIP`.

> **Important**
>
> L'option `CODEBUILD_CLONE_REF` ne peut être utilisée que par des actions CodeBuild en aval.
> Si vous choisissez cette option, vous devrez mettre à jour les autorisations associées à votre rôle de service de CodeBuild projet, comme indiqué dansAjoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com. Pour consulter un didacticiel expliquant comment utiliser l'option de clonage complet, voirTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

DetectChanges

Obligatoire : non

Contrôle le démarrage automatique de votre pipeline lorsqu'un nouveau commit est effectué sur le référentiel et la branche configurés. Si ce n'est pas spécifié, la valeur par défaut est`true`, et le champ ne s'affiche pas par défaut. Valeurs valides pour ce paramètre :

- `true`: démarre CodePipeline automatiquement votre pipeline lors de nouvelles validations.
- `false`: CodePipeline ne démarre pas votre pipeline lors de nouveaux commits.

Artefacts d'entrée

- Nombre d'objets : `0`
- Description : Les artefacts d'entrée ne s'appliquent pas à ce type d'action.

Artefacts de sortie

- Nombre d'objets : 1
- Description : Les artefacts générés à partir du référentiel sont les artefacts de sortie de l'action `CodeStarSourceConnection`. L'ID de validation du code source est affiché en CodePipeline tant que révision source pour l'exécution du pipeline déclenchée. Vous pouvez configurer l'artefact de sortie de cette action dans :
  - Un fichier ZIP qui regroupe le contenu du référentiel et de la branche configurés au moment de la validation spécifiée comme révision source pour l'exécution du pipeline.
  - Un fichier JSON qui contient une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git.

> **Important**
>
> Cette option ne peut être utilisée que par des actions CodeBuild en aval.
> Si vous choisissez cette option, vous devrez mettre à jour les autorisations associées à votre rôle de service de CodeBuild projet, comme indiqué dansRésolution des problèmes CodePipeline. Pour consulter un didacticiel expliquant comment utiliser l'option de clonage complet, voirTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

Variables de sortie

Lorsque cette action est configurée, elle produit des variables qui peuvent être référencées par la configuration d'action d'une action en aval dans le pipeline. Cette action produit des variables qui peuvent être visualisées en tant que variables de sortie, même si l'action n'a pas d'espace de noms. Vous configurez une action avec un espace de noms pour rendre ces variables disponibles pour la configuration des actions en aval.

Pour plus d'informations, consultez Référence des variables.

AuthorDate

Date à laquelle la validation a été créée, au format horodatage.

BranchName

Nom de la branche du référentiel où la modification de la source a été effectuée.

CommitId

ID de validation ayant déclenché l'exécution du pipeline.

CommitMessage

Message de description, le cas échéant, associé à la validation ayant déclenché l'exécution du pipeline.

ConnectionArn

ARN de connexion qui est configuré et authentifié pour le fournisseur de source.

FullRepositoryName

Nom du référentiel où la validation ayant déclenché le pipeline a été effectuée.

Autorisations relatives aux rôles de service : AWS CodeConnections action

En AWS CodeConnections effet, l'autorisation suivante est requise pour créer des pipelines avec une source utilisant une connexion, telle que Bitbucket Cloud.

```
{
    "Effect": "Allow",
    "Action": [
        "codeconnections:UseConnection",
        "codestar-connections:UseConnection"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Les deux actions sont obligatoires. Opérations invoquées par appel`codestar-connections:UseConnection`, tandis que les opérations sont invoquées en dehors de l'appel`codeconnections:UseConnection`. L'octroi des deux actions garantit que le rôle fonctionne pour les deux chemins de code, que l'ARN de la ressource de la connexion utilise `codeconnections` ou non le préfixe `codestar-connections` de service.

Pour plus d'informations sur les autorisations IAM pour les connexions, consultez la section Référence des autorisations de connexion.

Déclaration d'action

Dans l'exemple suivant, l'artefact de sortie est défini au format ZIP par défaut `CODE_ZIP` pour la connexion avec l'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"
        }
    ]
},
```

Installation de l'application d'installation et création d'une connexion

La première fois que vous utilisez la console pour ajouter une nouvelle connexion à un référentiel tiers, vous devez autoriser CodePipeline l'accès à vos référentiels. Vous choisissez ou créez une application d'installation qui vous aide à vous connecter au compte sur lequel vous avez créé votre référentiel de codes tiers.

Lorsque vous utilisez le modèle AWS CLI ou un CloudFormation modèle, vous devez fournir l'ARN de connexion d'une connexion déjà passée par le handshake d'installation. Sinon, le pipeline n'est pas déclenché.

> **Note**
>
> Pour une action `CodeStarSourceConnection` source, il n'est pas nécessaire de configurer un webhook ou d'effectuer un sondage par défaut. L'action Connexions gère pour vous la détection de votre changement de source.

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- AWS::CodeStarConnections: :Connection — La référence du CloudFormation modèle pour la ressource AWS CodeStar Connections fournit des paramètres et des exemples de connexions dans des CloudFormation modèles.
- AWS CodeStarRéférence de l'API AWS CodeStar Connections — La référence de l'API Connections fournit des informations de référence pour les actions de connexion disponibles.
- Pour connaître les étapes de création d'un pipeline avec des actions source prises en charge par des connexions, consultez les pages suivantes :
  - Pour Bitbucket Cloud, utilisez l'option Bitbucket dans la console ou l'`CodestarSourceConnection`action dans la CLI. Consultez Connexions Bitbucket Cloud.
  - Pour GitHub et GitHub Enterprise Cloud, utilisez l'option GitHubfournisseur dans la console ou l'`CodestarSourceConnection`action dans la CLI. Consultez GitHub connexions.
  - Pour GitHub Enterprise Server, utilisez l'option du fournisseur GitHub Enterprise Server dans la console ou l'`CodestarSourceConnection`action dans la CLI. Consultez GitHub Connexions aux serveurs d'entreprise.
  - Pour GitLab.com, utilisez l'option GitLabfournisseur dans la console ou l'`CodestarSourceConnection`action avec le `GitLab` fournisseur dans la CLI. Consultez GitLab.com connexions.
- Pour consulter un didacticiel de démarrage qui crée un pipeline avec une source Bitbucket et une CodeBuild action, consultez Getting started with connections.
- Pour consulter un didacticiel expliquant comment se connecter à un GitHub référentiel et utiliser l'option Clonage complet avec une CodeBuild action en aval, voirTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

Référence des actions des commandes

L'action Commandes vous permet d'exécuter des commandes shell dans une instance de calcul virtuelle. Lorsque vous exécutez l'action, les commandes spécifiées dans la configuration de l'action sont exécutées dans un conteneur distinct. Tous les artefacts spécifiés comme artefacts d'entrée pour une CodeBuild action sont disponibles dans le conteneur qui exécute les commandes. Cette action vous permet de spécifier des commandes sans créer de CodeBuild projet au préalable. Pour plus d'informations, veuillez consulter les sections ActionDeclaration et OutputArtifact (français non garanti) de la Référence d'API AWS CodePipeline .

Important

Cette action utilise le CodeBuild calcul CodePipeline géré pour exécuter des commandes dans un environnement de génération. L'exécution de l'action des commandes entraînera des frais distincts dans AWS CodeBuild.

Note

L'action Commandes n'est disponible que pour les pipelines de type V2.

Rubriques

- Considérations relatives à l'action des commandes
- Autorisations relatives aux politiques de rôle de service
- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables d'environnement
- Autorisations relatives aux rôles de service : action des commandes
- Déclaration d'action (exemple)
- Consultez aussi

Considérations relatives à l'action des commandes

Les considérations suivantes s'appliquent à l'action Commandes.

- L'action de commandes utilise CodeBuild des ressources similaires à l' CodeBuild action, tout en autorisant les commandes d'environnement shell dans une instance de calcul virtuelle sans qu'il soit nécessaire d'associer ou de créer un projet de génération.

  Note

  L'exécution de l'action des commandes entraînera des frais distincts dans AWS CodeBuild.

- Étant donné que l'action CodeBuild Commandes CodePipeline utilise des ressources, les builds exécutés par l'action seront attribués aux limites de génération de votre compte dans CodeBuild. Les builds exécutés par l'action Commandes seront pris en compte dans les limites de génération simultanées configurées pour ce compte.
- Le délai d'expiration des builds avec l'action Commandes est de 55 minutes, selon les CodeBuild builds.
- L'instance de calcul utilise un environnement de génération isolé dans CodeBuild.

> **Note**
>
> Étant donné que l'environnement de génération isolé est utilisé au niveau du compte, une instance peut être réutilisée pour une autre exécution de pipeline.

- Tous les formats sont pris en charge, à l'exception des formats multilignes. Vous devez utiliser le format sur une seule ligne lors de la saisie des commandes.
- L'action des commandes est prise en charge pour les actions entre comptes. Pour ajouter une action de commande inter-comptes, ajoutez-la `actionRoleArn` depuis votre compte cible dans la déclaration d'action.
- Pour cette action, CodePipeline assumera le rôle de service de pipeline et utilisera ce rôle pour autoriser l'accès aux ressources lors de l'exécution. Il est recommandé de configurer le rôle de service de manière à ce que les autorisations soient limitées au niveau de l'action.
- Les autorisations ajoutées au rôle CodePipeline de service sont détaillées dansAjouter des autorisations au rôle CodePipeline de service.
- L'autorisation requise pour afficher les journaux dans la console est détaillée dansAutorisations requises pour consulter les journaux de calcul dans la console.
- Contrairement aux autres actions de CodePipeline, vous ne définissez pas de champs dans la configuration des actions ; vous définissez les champs de configuration des actions en dehors de la configuration des actions.

Autorisations relatives aux politiques de rôle de service

Lorsque l'action est CodePipeline exécutée, CodePipeline crée un groupe de journaux en utilisant le nom du pipeline comme suit. Cela vous permet de limiter les autorisations d'enregistrement des ressources à l'aide du nom du pipeline.

```
/aws/codepipeline/MyPipelineName
```

Si vous utilisez un rôle de service existant, pour utiliser l'action Commandes, vous devez ajouter les autorisations suivantes pour ce rôle de service.

- journaux : CreateLogGroup
- journaux : CreateLogStream
- journaux : PutLogEvents

Dans la déclaration de politique relative aux rôles de service, réduisez les autorisations au niveau du pipeline, comme indiqué dans l'exemple suivant.

```
{
    "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:*"
   ]
}
```

Pour afficher les journaux dans la console à l'aide de la page de dialogue des détails des actions, l'autorisation d'affichage des journaux doit être ajoutée au rôle de console. Pour plus d'informations, consultez l'exemple de politique d'autorisations de console dansAutorisations requises pour consulter les journaux de calcul dans la console.

Type d'action

- Catégorie : `Compute`
- Propriétaire : `AWS`
- Fournisseur : `Commands`
- Version : `1`

Paramètres de configuration

Commandes

Obligatoire : oui

Vous pouvez fournir des commandes shell pour exécuter l'`Commands`action. Dans la console, les commandes sont saisies sur des lignes distinctes. Dans la CLI, les commandes sont saisies sous forme de chaînes distinctes.

> **Note**
>
> Multi-line les formats ne sont pas pris en charge et un message d'erreur s'affichera. Single-line format doit être utilisé pour saisir des commandes dans le champ Commandes.

> **Important**
>
> Les ComputeType valeurs EnvironmentType et correspondent à celles de CodeBuild. Nous prenons en charge un sous-ensemble des types disponibles. Pour de plus amples informations, veuillez consulter Types de calcul de l'environnement de génération.

EnvironmentType

Obligatoire : non

L'image du système d'exploitation de l'environnement de génération qui prend en charge l'action Commandes. Les valeurs valides pour les environnements de construction sont les suivantes :

- LINUX_CONTAINER
- WINDOWS_SERVER_2022_CONTAINER

La sélection de EnvironmentType autorisera ensuite le type de calcul pour ce système d'exploitation dans le ComputeType champ. Pour plus d'informations sur les types de CodeBuild calcul disponibles pour cette action, consultez la référence sur les modes et types de calcul de l'environnement de création dans le Guide de CodeBuild l'utilisateur.

> **Note**
>
> S'il n'est pas spécifié, le calcul prend par défaut la valeur suivante pour l'environnement de génération :
>
> - Type de calcul : BUILD_GENERAL1_SMALL
> - Type d'environnement : LINUX_CONTAINER

ComputeType

Obligatoire : non

Sur la base de la sélection pour EnvironmentType, le type de calcul peut être fourni. Les valeurs disponibles pour le calcul sont les suivantes. Notez toutefois que les options disponibles peuvent varier en fonction du système d'exploitation.

- BUILD_GENERAL1_SMALL
- BUILD_GENERAL1_MEDIUM
- BUILD_GENERAL1_LARGE

> **Important**
>
> Certains types de calcul ne sont pas compatibles avec certains types d'environnement. Par exemple, WINDOWS_SERVER_2022_CONTAINER n'est pas compatible avec BUILD_GENERAL1_SMALL. L'utilisation de combinaisons incompatibles entraîne l'échec de l'action et génère une erreur d'exécution.

Variables de sortie

Obligatoire : non

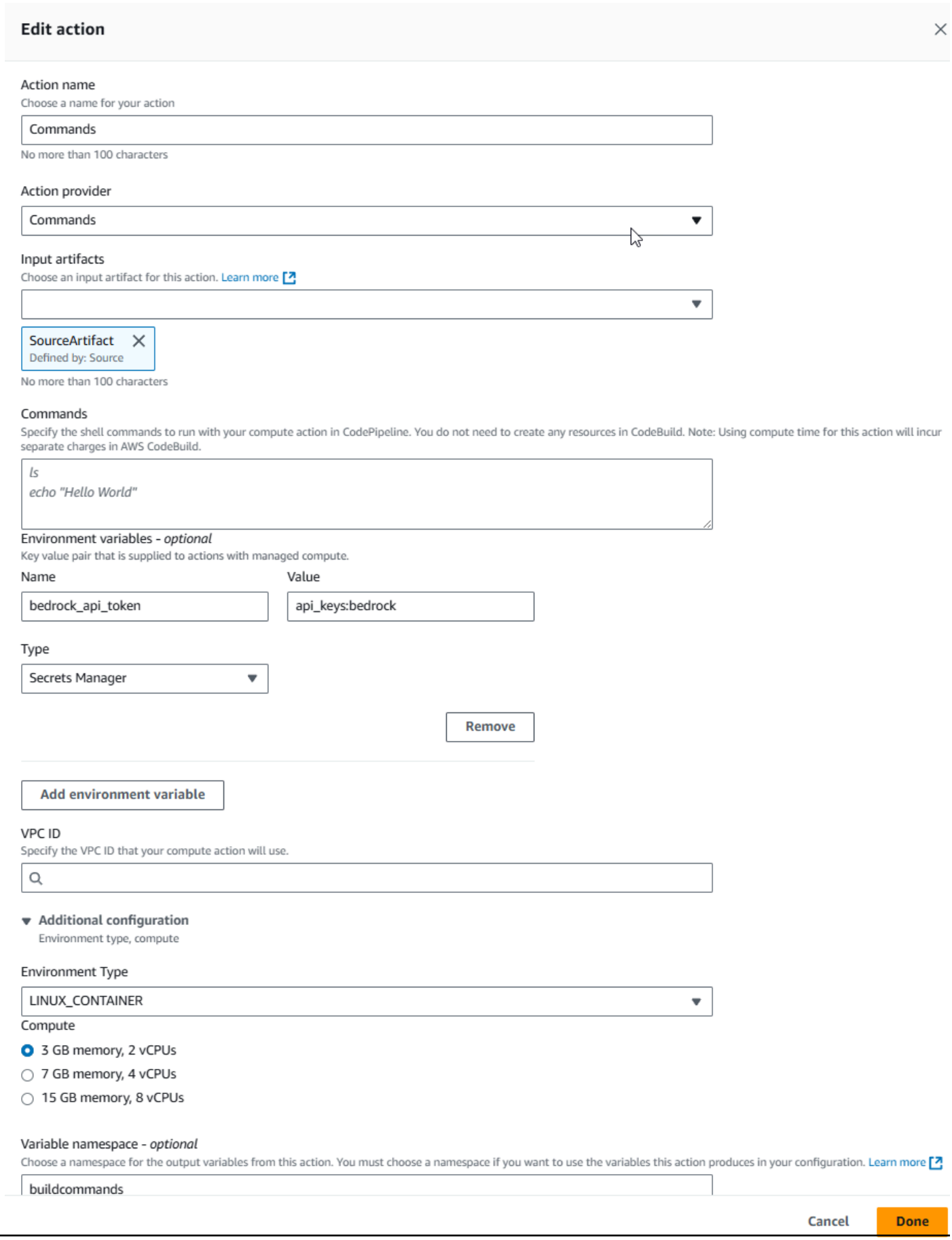
Spécifiez les noms des variables de votre environnement que vous souhaitez exporter. Pour obtenir une référence sur les variables d' CodeBuild environnement, consultez la section Variables d'environnement dans les environnements de génération du Guide de CodeBuild l'utilisateur.

Fichiers

Obligatoire : non

Vous pouvez fournir les fichiers que vous souhaitez exporter en tant qu'artefacts de sortie pour l'action.

Le format pris en charge pour les fichiers est le même que pour les modèles de CodeBuild fichiers. Par exemple, entrez `**/` pour tous les fichiers. Pour plus d'informations, consultez la section Créer une référence de spécification CodeBuild dans le Guide de CodeBuild l'utilisateur.


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

Obligatoire : non

L'ID VPC de vos ressources.

Subnets

Obligatoire : non

Les sous-réseaux du VPC. Ce champ est nécessaire lorsque vos commandes doivent se connecter aux ressources d'un VPC.

SecurityGroupIds

Obligatoire : non

Les groupes de sécurité pour le VPC. Ce champ est nécessaire lorsque vos commandes doivent se connecter aux ressources d'un VPC.

Voici un exemple JSON de l'action avec des champs de configuration affichés pour l'environnement et le type de calcul, ainsi qu'un exemple de variable d'environnement.

```
{
           "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
        }
```

Artefacts d'entrée

- Nombre d'objets : `1 to 10`

Artefacts de sortie

- Nombre d'objets : `0 to 1`

Variables d'environnement

Clé

La clé d'une paire de variables d'environnement clé-valeur, telle que. `BEDROCK_TOKEN`

Value

La valeur de la paire clé-valeur, telle que. `apiTokens:bedrockToken` La valeur peut être paramétrée à l'aide de variables de sortie issues d'actions de pipeline ou de variables de pipeline.

Lorsque vous utilisez le `SECRETS_MANAGER` type, cette valeur doit être le nom d'un secret que vous avez déjà stocké dans AWS Secrets Manager.

Type

Spécifie le type d'utilisation de la valeur de la variable d'environnement. La valeur peut être `PLAINTEXT` ou `SECRETS_MANAGER`. Si la valeur est`SECRETS_MANAGER`, indiquez la référence Secrets dans la `EnvironmentVariable` valeur. Si rien spécifié, la valeur par défaut est `PLAINTEXT`.

> **Note**
>
> Nous déconseillons vivement l'utilisation de variables d'environnement en texte brut pour stocker des valeurs sensibles, en particulier AWS des informations d'identification. Lorsque vous utilisez la CodeBuild console ou AWS CLI, les variables d'environnement en texte brut sont affichées en texte brut. Pour les valeurs sensibles, nous vous recommandons d'utiliser plutôt le type `SECRETS_MANAGER`.

> **Note**
>
> Lorsque vous entrez la configuration `namevalue`, et `type` pour vos variables d'environnement, en particulier si la variable d'environnement contient la syntaxe de la variable de CodePipeline sortie, ne dépassez pas la limite de 1 000 caractères pour le champ de valeur de la configuration. Une erreur de validation est renvoyée lorsque cette limite est dépassée.

Pour un exemple de déclaration d'action montrant une variable d'environnement, consultezParamètres de configuration.

> **Note**
>
> - Le `SECRETS_MANAGER` type n'est pris en charge que pour l'action Commandes.
> - Les secrets référencés dans l'action Commandes seront supprimés dans les journaux de construction de la même manière que CodeBuild. Toutefois, les utilisateurs du pipeline qui disposent d'un accès de modification au pipeline peuvent toujours accéder à ces valeurs secrètes en modifiant les commandes.
> - Pour utiliser le SecretsManager, vous devez ajouter les autorisations suivantes à votre rôle de service de pipeline :
>
> ```
> {
>             "Effect": "Allow",
>             "Action": [
>                 "secretsmanager:GetSecretValue"
>             ],
>             "Resource": [
> ```

```
                "SECRET_ARN"
            ]
        }
```

Autorisations relatives aux rôles de service : action des commandes

Pour la prise en charge des commandes, ajoutez ce qui suit à votre déclaration de politique :

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*"
            ]
        }
    ]
}
```

Déclaration d'action (exemple)

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"
}
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Tutoriel : Création d'un pipeline qui exécute des commandes avec compute (type V2)— Ce didacticiel fournit un exemple de pipeline avec l'action Commandes.

AWS Device Farm référence de l'action de test

Dans votre pipeline, vous pouvez configurer une action de test qui permet d' AWS Device Farm exécuter et de tester votre application sur des appareils. Device Farm utilise des pools de tests d'appareils et des frameworks de test pour tester des applications sur des appareils spécifiques. Pour plus d'informations sur les types de frameworks de test pris en charge par l'action Device Farm, voir Utilisation des types de test dans AWS Device Farm.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Autorisations relatives aux rôles de service : AWS Device Farm action
- Déclaration d'action
- Consultez aussi

Type d'action

- Catégorie : `Test`
- Propriétaire : `AWS`
- Fournisseur : `DeviceFarm`
- Version : `1`

Paramètres de configuration

AppType

Obligatoire : oui

Le système d'exploitation et le type d'application que vous testez. Voici une liste de valeurs valides :

- `iOS`
- `Android`
- `Web`

ProjectId

Obligatoire : oui

L'ID du projet Device Farm.

Pour trouver l'ID de votre projet, dans la console Device Farm, choisissez votre projet. Dans le navigateur, copiez l'URL de votre nouveau projet. L'URL contient l'ID de projet. L'ID du projet est la valeur de l'URL suivante`projects/`. Dans l'exemple suivant, l'ID du projet est`eec4905f-98f8-40aa-9afc-4c1cfexample`.

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

Appli

Obligatoire : oui

Le nom et l'emplacement du fichier de demande dans votre artefact d'entrée. Par exemple : `s3-ios-test-1.ipa`

TestSpec

Conditionnel : Oui

Emplacement du fichier de définition des spécifications de test dans votre artefact d'entrée. Ceci est requis pour le test du mode personnalisé.

DevicePoolArn

Obligatoire : oui

L'ARN du pool d'appareils Device Farm.

Pour obtenir les ARN du pool de périphériques disponibles pour le projet, y compris l'ARN des principaux appareils, utilisez l' AWS interface de ligne de commande pour entrer la commande suivante :

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

TestType

Obligatoire : oui

Spécifie le framework de test pris en charge pour votre test. Voici une liste de valeurs valides pour `TestType` :

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG
- APPIUM_NODE
- APPIUM RUBY
- APPIUM PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- FUZZ INTÉGRÉ
- INSTRUMENTATION

- XCTEST
- XCTEST_UI

> **Note**
>
> Les types de test suivants ne sont pas pris en charge par l'action dans CodePipeline : `WEB_PERFORMANCE_PROFILEREMOTE_ACCESS_RECORD`, et`REMOTE_ACCESS_REPLAY`.

Pour plus d'informations sur les types de tests Device Farm, voir Utilisation des types de test dans AWS Device Farm.

RadioBluetoothEnabled

Obligatoire : non

Valeur booléenne qui indique s'il faut activer le Bluetooth au début du test.

RecordAppPerformanceData

Obligatoire : non

Valeur booléenne qui indique s'il faut enregistrer les données de performance de l'appareil, telles que les performances du processeur, des images par seconde et de la mémoire pendant le test.

RecordVideo

Obligatoire : non

Valeur booléenne qui indique s'il faut enregistrer une vidéo pendant le test.

RadioWifiEnabled

Obligatoire : non

Valeur booléenne qui indique s'il faut l'activer Wi-Fi au début du test.

RadioNfcEnabled

Obligatoire : non

Valeur booléenne qui indique s'il faut activer la technologie NFC au début du test.

RadioGpsEnabled

Obligatoire : non

Valeur booléenne qui indique s'il faut activer le GPS au début du test.

test

Obligatoire : non

Le nom et le chemin du fichier de définition de test dans votre emplacement source. Ce chemin dépend de la racine de l'artefact d'entrée de votre test.

FuzzEventCount

Obligatoire : non

Nombre d'événements d'interface utilisateur à exécuter par le test Fuzz, compris entre 1 et 10 000.

FuzzEventThrottle

Obligatoire : non

Le nombre de millisecondes que le test fuzz doit attendre avant d'effectuer le prochain événement d'interface utilisateur, compris entre 1 et 1 000.

FuzzRandomizerSeed

Obligatoire : non

Une graine pour le test Fuzz à utiliser pour randomiser les événements de l'interface utilisateur. L'utilisation du même numéro pour les tests de fuzz ultérieurs donne des séquences d'événements identiques.

CustomHostMachineArtifacts

Obligatoire : non

Emplacement sur la machine hôte où les artefacts personnalisés seront stockés.

CustomDeviceArtifacts

Obligatoire : non

Emplacement sur l'appareil où les artefacts personnalisés seront stockés.

UnmeteredDevicesOnly

Obligatoire : non

Valeur booléenne qui indique s'il faut utiliser uniquement vos appareils illimités lors de l'exécution des tests à cette étape.

JobTimeoutMinutes

Obligatoire : non

Le nombre de minutes pendant lesquelles un test sera exécuté par appareil avant son expiration.

Latitude

Obligatoire : non

La latitude de l'appareil exprimée en degrés du système de coordonnées géographiques.

Longitude

Obligatoire : non

La longitude de l'appareil exprimée en degrés du système de coordonnées géographiques.

Artefacts d'entrée

- Nombre d'objets : `1`
- Description : ensemble d'artefacts à mettre à disposition pour l'action de test. Device Farm recherche l'application intégrée et les définitions de test à utiliser.

Artefacts de sortie

- Nombre d'artefacts : `0`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Autorisations relatives aux rôles de service : AWS Device Farm action

Lors de l' CodePipeline exécution de l'action, la politique de rôle de CodePipeline service nécessite les autorisations suivantes, dont la portée est appropriée jusqu'à l'ARN de la ressource du pipeline afin de conserver l'accès avec le moins de privilèges possible. Par exemple, ajoutez ce qui suit à votre déclaration de politique :

```
{
```

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

Déclaration d'action

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"
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Utilisation des types de test dans Device Farm — Ce chapitre de référence du Device Farm Developer Guide fournit une description plus détaillée des frameworks de test d'applications Android, iOS et Web pris en charge par Device Farm.
- Actions dans Device Farm — Les appels d'API et les paramètres du Device Farm API Reference peuvent vous aider à travailler sur des projets Device Farm.
- Tutoriel : Créez un pipeline qui crée et teste votre application Android avec AWS Device Farm— Ce didacticiel fournit un exemple de fichier buildspec et un exemple d'application pour créer un

pipeline avec une GitHub source qui crée et teste une application Android avec Device CodeBuild Farm.

- Tutoriel : Créez un pipeline qui teste votre application iOS avec AWS Device Farm— Ce didacticiel fournit un exemple d'application permettant de créer un pipeline avec une source Amazon S3 qui teste une application iOS intégrée avec Device Farm.

Référence des actions de déploiement d'Elastic Beanstalk

Elastic Beanstalk est une plateforme intégrée AWS qui permet de déployer et de faire évoluer des applications Web. Vous utilisez une action Elastic Beanstalk pour déployer le code d'application dans votre environnement de déploiement.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Autorisations relatives aux rôles de service : action de ElasticBeanstalk déploiement
- Déclaration d'action
- Consultez aussi

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `ElasticBeanstalk`
- Version : `1`

Paramètres de configuration

ApplicationName

Obligatoire : oui

Le nom de l'application que vous avez créée dans Elastic Beanstalk.

EnvironmentName

Obligatoire : oui

Le nom de l'environnement que vous avez créé dans Elastic Beanstalk. Un environnement est un ensemble de AWS ressources exécutant une version d'application. Chaque environnement n'exécute qu'une seule version d'application à la fois ; toutefois, vous pouvez exécuter la même version d'application ou différentes versions d'application dans de nombreux environnements simultanément.

Artefacts d'entrée

- Nombre d'objets : `1`
- Description : artefact d'entrée pour l'action.

Artefacts de sortie

- Nombre d'objets : `0`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Autorisations relatives aux rôles de service : action de **`ElasticBeanstalk`** déploiement

Pour Elastic Beanstalk, les autorisations minimales requises pour créer des pipelines avec une `ElasticBeanstalk` action de déploiement sont les suivantes.

```
{
    "Effect": "Allow",
    "Action": [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
```

```
        "cloudformation:*",
        "rds:*",
        "sqs:*",
        "ecs:*"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Vous devez remplacer les caractères génériques dans la politique des ressources par les ressources du compte auquel vous souhaitez limiter l'accès. Pour plus d'informations sur la création d'une politique qui accorde l'accès avec le moindre privilège, consultez. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege

Déclaration d'action

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"
        }
    ]
},
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Déploiement d'une application Flask sur Elastic Beanstalk : ce didacticiel explique comment créer les ressources de votre application et de votre environnement dans Elastic Beanstalk à l'aide d'un exemple d'application Flask. Vous pouvez ensuite créer votre pipeline à l'aide d'une action de déploiement Elastic Beanstalk qui déploie votre application depuis votre référentiel source vers votre environnement Elastic Beanstalk.

Amazon Inspector **`InspectorScan`** invoque une référence d'action

Amazon Inspector est un service de gestion des vulnérabilités qui découvre automatiquement les charges de travail et les analyse en permanence pour détecter les vulnérabilités logicielles et les risques d'exposition involontaire au réseau. L'`InspectorScan`action CodePipeline permet d'automatiser la détection et la correction des failles de sécurité dans votre code open source. Il s'agit d'une action de calcul gérée dotée de fonctionnalités d'analyse de sécurité. Vous pouvez l'utiliser InspectorScan avec le code source de l'application dans votre référentiel tiers, tel que GitHub Bitbucket Cloud, ou avec des images pour des applications de conteneurs. Votre action analysera et signalera les niveaux de vulnérabilité et les alertes que vous configurez.

> **Important**
>
> Cette action utilise le CodeBuild calcul CodePipeline géré pour exécuter des commandes dans un environnement de génération. L'exécution de l'action entraînera des frais distincts. AWS CodeBuild

Rubriques

- ID du type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables de sortie
- Autorisations relatives aux rôles de service : InspectorScan action
- Déclaration d'action
- Consultez aussi

ID du type d'action

- Catégorie : `Invoke`
- Propriétaire : `AWS`
- Fournisseur : `InspectorScan`

- Version : 1

Exemple :

```
        {
            "Category": "Invoke",
            "Owner": "AWS",
            "Provider": "InspectorScan",
            "Version": "1"
        },
```

Paramètres de configuration

InspectorRunMode

(Obligatoire) Chaîne indiquant le mode de numérisation. Les valeurs valides sont `SourceCodeScan | ECRImageScan`.

ECRRepositoryNom

Nom du référentiel Amazon ECR dans lequel l'image a été transférée.

ImageTag

Balise utilisée pour l'image.

Les paramètres de cette action analysent les niveaux de vulnérabilité que vous spécifiez. Les niveaux de seuils de vulnérabilité suivants sont disponibles :

CriticalThreshold

Le nombre de vulnérabilités de gravité critique détectées dans votre source au-delà desquelles l'action ne CodePipeline devrait pas être exécutée.

HighThreshold

Nombre de vulnérabilités de haut niveau de gravité détectées dans votre source au-delà desquelles l'action ne CodePipeline devrait pas être exécutée.

MediumThreshold

Le nombre de vulnérabilités de gravité moyenne détectées dans votre source au-delà desquelles l'action CodePipeline devrait échouer.

LowThreshold

Le nombre de vulnérabilités de faible gravité détectées dans votre source au-delà desquelles l'action CodePipeline devrait échouer.

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

Artefacts d'entrée

- Nombre d'objets : 1
- Description : code source à analyser pour détecter les vulnérabilités. Si le scan concerne un référentiel ECR, cet artefact d'entrée n'est pas nécessaire.

Artefacts de sortie

- Nombre d'objets : 1
- Description : détails de la vulnérabilité de votre source sous la forme d'un fichier de nomenclature logicielle (SBOM).

Variables de sortie

Lorsque cette action est configurée, elle produit des variables qui peuvent être référencées par la configuration d'action d'une action en aval dans le pipeline. Cette action produit des variables qui peuvent être visualisées en tant que variables de sortie, même si l'action n'a pas d'espace de noms. Vous configurez une action avec un espace de noms pour rendre ces variables disponibles pour la configuration des actions en aval.

Pour de plus amples informations, veuillez consulter Référence des variables.

HighestScannedSeverity

Résultat de gravité le plus élevé de l'analyse. Les valeurs valides sont `medium | high | critical`.

Autorisations relatives aux rôles de service : **`InspectorScan`** action

Pour le soutien à l'`InspectorScan`action, ajoutez ce qui suit à votre déclaration de politique :

```
{
        "Effect": "Allow",
        "Action": "inspector-scan:ScanSbom",
        "Resource": "*"
    },
    {
        "Effect": "Allow",
```

```
        "Action": [
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
        ],
        "Resource": "resource_ARN"
    },
```

En outre, si elles ne sont pas déjà ajoutées pour l'action Commandes, ajoutez les autorisations suivantes à votre rôle de service afin de consulter CloudWatch les journaux.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Réduisez les autorisations au niveau des ressources du pipeline en utilisant les autorisations basées sur les ressources dans la déclaration de politique relative aux rôles de service.

Déclaration d'action

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"
                    },
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Pour plus d'informations sur Amazon Inspector, consultez le guide de l'utilisateur d'Amazon Inspector.

AWS Lambda invoquer la référence de l'action

Vous permet d'exécuter une fonction Lambda en tant qu'action dans votre pipeline. En utilisant l'objet d'événement qui est une entrée de cette fonction, la fonction a accès à la configuration de l'action, aux emplacements des artefacts d'entrée, aux emplacements des artefacts de sortie et à d'autres informations requises pour accéder aux artefacts. Pour un exemple d'événement transmis à une fonction d'invocation Lambda, consultez. Exemple d'événement JSON Dans le cadre de l'implémentation de la fonction Lambda, il doit y avoir un appel à `PutJobSuccessResult API` ou `PutJobFailureResult API`. Sinon, l'exécution de cette action se bloque jusqu'à ce que l'action expire. Si vous spécifiez des artefacts de sortie pour cette action, ils doivent être chargés dans le compartiment S3 dans le cadre de l'implémentation de la fonction.

> **Important**
>
> N'enregistrez pas l'événement JSON CodePipeline envoyé à Lambda car cela peut entraîner la journalisation des informations d'identification de l'utilisateur dans CloudWatch les journaux. Le CodePipeline rôle utilise un événement JSON pour transmettre des informations d'identification temporaires à Lambda `artifactCredentials` sur le terrain. Pour voir un exemple d'événement, consultez la section Exemple d'événement JSON.

Type d'action

- Catégorie : `Invoke`
- Propriétaire : `AWS`
- Fournisseur : `Lambda`
- Version : `1`

Paramètres de configuration

FunctionName

Obligatoire : oui

`FunctionName`est le nom de la fonction créée dans Lambda.

UserParameters

Obligatoire : non

Chaîne qui peut être traitée comme entrée par la fonction Lambda.

Artefacts d'entrée

- Nombre d'artefacts : `0 to 5`
- Description : ensemble d'artefacts à mettre à la disposition de la fonction Lambda.

Artefacts de sortie

- Nombre d'artefacts : `0 to 5`
- Description : ensemble d'artefacts produits en sortie par la fonction Lambda.

Variables de sortie

Cette action produira en tant que variables toutes les paires clé-valeur incluses dans la `outputVariables` section de la demande d'PutJobSuccessResult API.

Pour plus d'informations sur les variables dans CodePipeline, consultezRéférence des variables.

Exemple de configuration d'action

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"
        }
    ]
},
```

Exemple d'événement JSON

L'action Lambda envoie un événement JSON qui contient l'ID de tâche, la configuration de l'action du pipeline, les emplacements des artefacts en entrée et en sortie et toutes les informations de chiffrement relatives aux artefacts. L'assistant accède à ces informations pour terminer l'action Lambda. Pour de plus amples informations, veuillez consulter les détails de la tâche. Voici un exemple d'événement.

```
{
    "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"
            }
        }
    }
}
```

L'événement JSON fournit les informations de tâche suivantes pour l'action Lambda dans CodePipeline :

- `id` : ID unique généré par le système de la tâche.
- `accountId`: ID de AWS compte associé à la tâche.
- `data` : autres informations requises pour qu'un exécutant de tâches termine la tâche.

- `actionConfiguration` : paramètres d'action pour l'action Lambda. Pour les définitions, veuillez consulter Paramètres de configuration.
- `inputArtifacts` : artefact fourni à l'action.
  - `location` : emplacement du magasin d'artefacts.
    - `s3Location` : informations sur l'emplacement des artefacts d'entrée pour l'action.
      - `bucketName`: nom du magasin d'artefacts du pipeline pour l'action (par exemple, un compartiment Amazon S3 nommé codepipeline-us-east-2-1234567890).
      - `objectKey` : nom de l'application (par exemple, `CodePipelineDemoApplication.zip`).
    - `type` : type d'artefact dans l'emplacement. Actuellement, `S3` est le seul type d'artefact valide.
  - `revision` : ID de révision de l'artefact. Selon le type d'objet, il peut s'agir d'un ID de validation (GitHub) ou d'un ID de révision (Amazon Simple Storage Service). Pour de plus amples informations, veuillez consulter ArtifactRevision.
  - `name` : nom de l'artefact à utiliser, tel que `MyApp`.
- `outputArtifacts` : sortie de l'action.
  - `location` : emplacement du magasin d'artefacts.
    - `s3Location` : informations sur l'emplacement des artefacts de sortie pour l'action.
      - `bucketName`: nom du magasin d'artefacts du pipeline pour l'action (par exemple, un compartiment Amazon S3 nommé codepipeline-us-east-2-1234567890).
      - `objectKey` : nom de l'application (par exemple, `CodePipelineDemoApplication.zip`).
    - `type` : type d'artefact dans l'emplacement. Actuellement, `S3` est le seul type d'artefact valide.
  - `revision` : ID de révision de l'artefact. Selon le type d'objet, il peut s'agir d'un ID de validation (GitHub) ou d'un ID de révision (Amazon Simple Storage Service). Pour de plus amples informations, veuillez consulter ArtifactRevision.
  - `name` : nom de la sortie d'un artefact, tel que `MyApp`.
- `artifactCredentials`: informations d'identification de AWS session utilisées pour accéder aux artefacts d'entrée et de sortie du compartiment Amazon S3. Ces informations d'identification sont des informations d'identification temporaires qui sont émises par AWS Security Token Service (AWS STS).

    - `secretAccessKey` : clé d'accès secrète pour la session.
    - `sessionToken` : jeton de la session.
    - `accessKeyId`: ID de clé d'accès pour la session.
- `continuationToken` : jeton généré par l'action. Les actions futures utilisent ce jeton pour identifier l'instance en cours d'exécution de l'action. Une fois l'action terminée, aucun jeton de continuation ne doit être fourni.
- `encryptionKey`: clé de chiffrement utilisée pour chiffrer les données du magasin d'artefacts, telle qu'une AWS KMS clé. Si cette valeur n'est pas définie, la clé par défaut d'Amazon Simple Storage Service est utilisée.
    - `id` : ID utilisé pour identifier la clé. Pour une clé AWS KMS , vous pouvez utiliser l'ID de clé, l'ARN de clé ou l'ARN d'alias.
    - `type` : type de clé de chiffrement, tel qu'une clé AWS KMS .

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- AWS CloudFormation Guide de l'utilisateur — Pour plus d'informations sur les actions Lambda et les CloudFormation artefacts pour les pipelines, voir Utilisation des fonctions de remplacement des paramètres avec les CodePipeline pipelines, Automatisation du déploiement des Lambda-based applications et Artefacts. AWS CloudFormation
- Invoquer un AWS Lambda fonction dans un pipeline dans CodePipeline— Cette procédure fournit un exemple de fonction Lambda et vous montre comment utiliser la console pour créer un pipeline avec une action d'invocation Lambda.

AWS OpsWorks référence des actions de déploiement

Vous utilisez une AWS OpsWorks action de déploiement à OpsWorks l'aide de votre pipeline.

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `OpsWorks`

- Version : 1

Paramètres de configuration

Appli

Obligatoire : oui

L' OpsWorks appli. L'application représente le code que vous souhaitez déployer et exécuter.

Pile

Obligatoire : oui

La OpsWorks pile. Une pile est un conteneur pour votre infrastructure d'applications.

Couche

Obligatoire : non

La OpsWorks couche. Une couche spécifie la configuration et les ressources d'un ensemble d'instances.

Artefacts d'entrée

- Nombre d'objets : `1`
- Description : Il s'agit de l'artefact d'entrée pour votre action.

Artefacts de sortie

- Nombre d'objets : `0 to 1`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Autorisations relatives aux rôles de service : AWS OpsWorks action

Pour obtenir de l' AWS OpsWorks aide, ajoutez ce qui suit à votre déclaration de politique :

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

```
    "Action": [
        "opsworks:CreateDeployment",
        "opsworks:DescribeApps",
        "opsworks:DescribeCommands",
        "opsworks:DescribeDeployments",
        "opsworks:DescribeInstances",
        "opsworks:DescribeStacks",
        "opsworks:UpdateApp",
        "opsworks:UpdateStack"
    ],
    "Resource": "resource_ARN"
},
```

Exemple de configuration d'action

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"
    }
}
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- AWS OpsWorks Guide de l'utilisateur : pour plus d'informations sur le déploiement avec AWS OpsWorks, consultez le Guide de AWS OpsWorks l'utilisateur.

AWS Service Catalog référence des actions de déploiement

Vous utilisez une AWS Service Catalog action pour déployer des modèles à l'aide de votre pipeline. Il s'agit de modèles de ressources que vous avez créés dans Service Catalog.

Type d'action

- Catégorie : `Deploy`
- Propriétaire : `AWS`
- Fournisseur : `ServiceCatalog`
- Version : `1`

Paramètres de configuration

TemplateFilePath

Obligatoire : oui

Le chemin du fichier de votre modèle de ressource dans votre emplacement source.

ProductVersionName

Obligatoire : oui

La version du produit dans le catalogue de services.

ProductType

Obligatoire : oui

Type de produit dans le catalogue de services.

ProductId

Obligatoire : oui

L'identifiant du produit dans le catalogue de services.

ProductVersionDescription

Obligatoire : non

Description de la version du produit dans le catalogue de services.

Artefacts d'entrée

- Nombre d'objets : `1`
- Description : Il s'agit de l'artefact d'entrée pour votre action.

Artefacts de sortie

- Nombre d'objets : `0`
- Description : Les artefacts de sortie ne s'appliquent pas à ce type d'action.

Autorisations relatives aux rôles de service : action Service Catalog

Pour le support du catalogue de services, ajoutez ce qui suit à votre déclaration de politique :

```
{
    "Effect": "Allow",
    "Action": [
        "servicecatalog:ListProvisioningArtifacts",
        "servicecatalog:CreateProvisioningArtifact",
        "servicecatalog:DescribeProvisioningArtifact",
        "servicecatalog:DeleteProvisioningArtifact",
        "servicecatalog:UpdateProduct"
```

```
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:ValidateTemplate"
    ],
    "Resource": "resource_ARN"
}
```

Exemples de configurations d'actions par type de fichier de configuration

L'exemple suivant montre une configuration valide pour une action de déploiement qui utilise Service Catalog, pour un pipeline créé dans la console sans fichier de configuration distinct :

```
"configuration": {
  "TemplateFilePath": "S3_template.json",
  "ProductVersionName": "devops S3 v2",
  "ProductType": "CLOUD_FORMATION_TEMPLATE",
  "ProductVersionDescription": "Product version description",
  "ProductId": "prod-example123456"
}
```

L'exemple suivant montre une configuration valide pour une action de déploiement qui utilise Service Catalog, pour un pipeline créé dans la console avec un fichier de `sample_config.json` configuration distinct :

```
"configuration": {
  "ConfigurationFilePath": "sample_config.json",
  "ProductId": "prod-example123456"
}
```

Exemple de configuration d'action

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"
    }
}
```

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Guide de l'utilisateur du catalogue de services : pour plus d'informations sur les ressources et les modèles du catalogue de services, consultez le guide de l'utilisateur du catalogue de services.

- Tutoriel : Création d'un pipeline qui se déploie vers Service Catalog— Ce didacticiel explique comment créer et configurer un pipeline pour déployer votre modèle de produit dans Service Catalog et appliquer les modifications que vous avez apportées à votre référentiel source.

AWS Step Functions invoquer une référence d'action

Une AWS CodePipeline action qui effectue les opérations suivantes :

- Démarre l'exécution AWS Step Functions d'une machine à états à partir de votre pipeline.
- Fournit un état initial à la machine d'état via une propriété dans la configuration de l'action ou un fichier situé dans un artefact de pipeline à transmettre en entrée.
- Définit éventuellement un préfixe d'ID d'exécution pour identifier les exécutions provenant de l'action.
- Prend en charge les machines d'état Standard et Express .

> **Note**
>
> L'action Step Functions s'exécute sur Lambda, et ses quotas de taille d'artefact sont donc les mêmes que ceux des fonctions Lambda. Pour plus d'informations, consultez la section Quotas Lambda dans le guide du développeur Lambda.

Type d'action

- Catégorie : `Invoke`
- Propriétaire : `AWS`
- Fournisseur : `StepFunctions`
- Version : `1`

Paramètres de configuration

StateMachineArn

Obligatoire : oui

ARN (Amazon Resource Name) de la machine d'état à appeler.

ExecutionNamePrefix

Obligatoire : non

Par défaut, l'ID d'exécution de l'action est utilisé comme nom d'exécution de la machine d'état. Si un préfixe est fourni, il est ajouté à l'ID d'exécution de l'action avec un trait d'union et utilisé conjointement comme nom d'exécution de la machine d'état.

```
myPrefix-1624a1d1-3699-43f0-8e1e-6bafd7fde791
```

Pour une machine d’état Express, le nom doit uniquement contenir 0-9, A-Z, a-z, - et _.

InputType

Obligatoire : non

- Littéral (par défaut) : lorsqu'elle est spécifiée, la valeur du champ Entrée est transmise directement à l'entrée de la machine d'état.

  Exemple d'entrée pour le champ Entrée lorsque Littéral est sélectionné :

  ```
  {"action": "test"}
  ```

- FilePath: Le contenu d'un fichier dans l'artefact d'entrée spécifié par le champ Entrée est utilisé comme entrée pour l'exécution de la machine à états. Un artefact d'entrée est requis lorsqu'il InputTypeest défini sur. FilePath

  Exemple d'entrée pour le champ de saisie lorsqu'il FilePathest sélectionné :

  ```
  assets/input.json
  ```

Input

Obligatoire : Conditionnelle

- Littéral : lorsqu'il InputTypeest défini sur Littéral (par défaut), ce champ est facultatif.

  S'il est renseigné, le champ Entrée est utilisé directement comme entrée pour l'exécution de la machine d'état. Sinon, la machine d'état est appelée avec un objet JSON {} vide.

- FilePath: Lorsque ce champ InputTypeest défini sur FilePath, ce champ est obligatoire.

  Un artefact d'entrée est également requis lorsqu'il InputTypeest défini sur. FilePath

Le contenu du fichier dans l'artefact d'entrée spécifié est utilisé comme entrée pour l'exécution de la machine d'état.

Artefacts d'entrée

- Nombre d'objets : `0 to 1`
- Description : S'il InputTypeest défini sur FilePath, cet artefact est obligatoire et est utilisé pour obtenir l'entrée pour l'exécution de la machine à états.

Artefacts de sortie

- Nombre d'objets : `0 to 1`
- Description :
  - Machines d'état standard: s'il est fourni, l'artefact de sortie est renseigné avec la sortie de la machine d'état. Ceci est obtenu à partir de la `output` propriété de la réponse de l' DescribeExecution API Step Functions une fois que l'exécution de la machine à états s'est terminée avec succès.
  - Machines d'état Express : non prises en charge.

Variables de sortie

Cette action produit des variables de sortie qui peuvent être référencées par la configuration d'action d'une action en aval dans le pipeline.

Pour de plus amples informations, veuillez consulter Référence des variables.

StateMachineArn

ARN de la machine d'état.

ExecutionArn

ARN de l'exécution de la machine d'état. Machines d'état standard uniquement.

Autorisations relatives aux rôles de service : **`StepFunctions`** action

En ce qui `StepFunctions` concerne l'action, les autorisations minimales requises pour créer des pipelines avec une action d'appel Step Functions sont les suivantes.

```
{
    "Effect": "Allow",
    "Action": [
        "states:DescribeStateMachine",
        "states:DescribeExecution",
        "states:StartExecution"
    ],
    "Resource": "resource_ARN"
},
```

Exemple de configuration d'action

Exemple pour l'entrée par défaut

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"
    }
}
```

Exemple pour l'entrée littérale

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\"}"
    }
}
```

Exemple pour le fichier d'entrée

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"
    }
}
```

Comportement

Au cours d'une version, CodePipeline exécute la machine à états configurée à l'aide de l'entrée spécifiée dans la configuration de l'action.

Lorsqu'il InputTypeest défini sur Literal, le contenu du champ de configuration de l'action d'entrée est utilisé comme entrée pour la machine à états. Lorsque l'entrée littérale n'est pas fournie, l'exécution de la machine d'état utilise un objet JSON {} vide. Pour plus d'informations sur l'exécution d'une exécution par machine à états sans saisie, consultez l' StartExecutionAPI Step Functions.

Lorsque InputTypece paramètre est défini sur FilePath, l'action décompresse l'artefact d'entrée et utilise le contenu du fichier spécifié dans le champ Configuration de l'action d'entrée comme entrée pour la machine à états. Lorsqu'il FilePathest spécifié, le champ de saisie est obligatoire et un artefact d'entrée doit exister ; sinon, l'action échoue.

Après un démarrage d'exécution réussi, le comportement diverge pour les deux types de machines d'état, standard et express.

Machines d'état standard

Si l'exécution de la machine à états standard a démarré avec succès, CodePipeline interroge l'`DescribeExecution`API jusqu'à ce que l'exécution atteigne le statut de terminal. Si l'exécution se termine correctement, l'action réussit ; sinon, elle échoue.

Si un artefact de sortie est configuré, l'artefact contiendra la valeur de retour de la machine d'état. Ceci est obtenu à partir de la `output` propriété de la réponse de l' DescribeExecution API Step Functions une fois que l'exécution de la machine à états s'est terminée avec succès. Remarque : des contraintes de longueur de sortie sont appliquées sur cette API.

Gestion des erreurs

- Si l'action ne parvient pas à démarrer l'exécution d'une machine d'état, l'exécution de l'action échoue.
- Si l'exécution de la machine à états ne parvient pas à atteindre le statut de terminal avant que l'action CodePipeline Step Functions n'atteigne son délai d'expiration (7 jours par défaut), l'exécution de l'action échoue. La machine d'état peut continuer malgré cet échec. Pour plus d'informations sur les délais d'exécution des machines à états dans Step Functions, voir Standard vs. Express Workflows.

  > **Note**
  >
  > Vous pouvez demander une augmentation du quota pour le délai d'expiration de l'action d'appel pour le compte avec l'action. Toutefois, l'augmentation du quota s'applique à toutes les actions de ce type dans toutes les régions pour ce compte.

- Si l'exécution de la machine d'état atteint un statut terminal de FAILED, TIMED_OUT ou ABORTED, l'exécution de l'action échoue.

Machines d'état express

Si l'exécution de la machine d'état express a été démarrée avec succès, l'exécution de l'action d'appel se termine avec succès.

Considérations relatives aux actions configurées pour les machines d'état express :

- Vous ne pouvez pas désigner un artefact de sortie.
- L'action n'attend pas la fin de l'exécution de la machine d'état.

- Une fois l'exécution de l'action démarrée CodePipeline, l'exécution de l'action réussit même si l'exécution de la machine à états échoue.

Gestion des erreurs

- Si CodePipeline l'exécution d'une machine à états échoue, l'exécution de l'action échoue. Sinon, l'action réussit immédiatement. L'action réussit CodePipeline quel que soit le temps nécessaire à l'exécution de la machine à états ou son résultat.

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- AWS Step Functions Guide du développeur — Pour plus d'informations sur les machines à états, les exécutions et les entrées pour les machines à états, consultez le guide du AWS Step Functions développeur.
- Tutoriel : Utiliser une AWS Step Functions action d'appel dans un pipeline— Ce didacticiel vous permet de démarrer avec un exemple de machine à états standard et vous montre comment utiliser la console pour mettre à jour un pipeline en ajoutant une action d'appel Step Functions.

Référence de structure de règles

Cette section est une référence pour la configuration des règles uniquement. Pour obtenir une présentation conceptuelle de la structure du pipeline, veuillez consulter CodePipeline référence de la structure du pipeline.

Chaque fournisseur de règles CodePipeline utilise un ensemble de champs de configuration obligatoires et facultatifs dans la structure du pipeline. Cette section fournit les informations de référence suivantes par fournisseur de règles :

- Valeurs valides pour les `RuleType` champs inclus dans le bloc de règles de structure du pipeline, telles que `Owner` et`Provider`.
- Descriptions et autres informations de référence relatives aux `Configuration` paramètres (obligatoires et facultatifs) inclus dans la section des règles relatives à la structure du pipeline.
- Exemples de champs de configuration de règles JSON et YAML valides.

Des informations de référence sont disponibles pour les fournisseurs de règles suivants :

Rubriques

- CloudWatchAlarm
- CodeBuild règle
- Commandes
- DeploymentWindow
- LambdaInvoke
- VariableCheck

CloudWatchAlarm

Lorsque vous créez une condition, vous pouvez ajouter la `CloudWatchAlarm` règle. Cette section fournit une référence pour les paramètres des règles. Pour plus d'informations sur les règles et conditions, consultezComment fonctionnent les conditions scéniques ?.

Vous devez déjà avoir créé une alarme dans Amazon en CloudWatch tant que ressource distincte.

Rubriques

- Type de règle
- Paramètres de configuration
- Exemple de configuration de règles
- Consultez aussi

Type de règle

- Catégorie : `Rule`
- Propriétaire : `AWS`
- Fournisseur : `CloudWatchAlarm`
- Version : `1`

Paramètres de configuration

AlarmName

Obligatoire : oui

Le nom de l' CloudWatch alarme. Il s'agit d'une ressource distincte créée dans CloudWatch.

AlarmStates

Obligatoire : non

Les états CloudWatch d'alarme souhaités pour la règle à surveiller. Les valeurs valides sont ALARM, OK et INSUSUFFISENT_DATA.

WaitTime

Obligatoire : non

Temps d'attente en minutes pour autoriser un changement d'état avant d'exécuter le résultat de la règle. Par exemple, configurez 20 minutes pour attendre qu'un état d'alarme passe à OK avant d'appliquer le résultat de la règle.

Exemple de configuration de règles

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"
                            }
                        ]
                    }
```

Consultez aussi

Les ressources connexes suivantes peuvent vous aider à utiliser cette règle.

- Création de conditions en cas de défaillance— Cette section décrit les étapes à suivre pour créer une condition En cas de défaillance avec une règle d'alarme.

CodeBuild règle

Lorsque vous créez une condition, vous pouvez ajouter la CodeBuild règle. Cette section fournit une référence pour les paramètres des règles. Pour plus d'informations sur les règles et conditions, consultezComment fonctionnent les conditions scéniques ?.

Vous pouvez utiliser la CodeBuild règle pour créer une condition dans laquelle l'exécution réussie de votre projet de génération répond aux critères de la règle, tels que la réussite de l'exécution de la génération pour une condition BeforeEntry.

> **Note**
>
> Pour les conditions BeforeEntry configurées avec le résultat Skip, seules les règles suivantes sont disponibles : `LambdaInvoke` et`VariableCheck`.

Rubriques

- Autorisations de politique des rôles de service
- Type de règle
- Paramètres de configuration
- Exemple de configuration de règles
- Consultez aussi

Autorisations de politique des rôles de service

Pour les autorisations relatives à cette règle, ajoutez ce qui suit à votre déclaration de politique de rôle de CodePipeline service. Limitez les autorisations au niveau des ressources.

```
{
```

```
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
    ],
    "Resource": "resource_ARN"
},
```

Type de règle

- Catégorie : `Rule`
- Propriétaire : `AWS`
- Fournisseur : `CodeBuild`
- Version : `1`

Paramètres de configuration

ProjectName

Obligatoire : oui

`ProjectName`est le nom du projet de construction dans CodeBuild.

PrimarySource

Obligatoire : Conditionnelle

La valeur du `PrimarySource` paramètre doit être le nom de l'un des artefacts d'entrée de l'action. CodeBuild recherche le fichier buildspec et exécute les commandes buildspec dans le répertoire qui contient la version décompressée de cet artefact.

Ce paramètre est obligatoire si plusieurs artefacts d'entrée sont spécifiés pour une action CodeBuild . Lorsqu'il n'y a qu'un seul artefact source pour l'action, l'artefact `PrimarySource` correspond par défaut à cet artefact.

BatchEnabled

Obligatoire : non

La valeur booléenne du `BatchEnabled` paramètre permet à l'action d'exécuter plusieurs builds au cours de la même exécution de build.

Lorsque cette option est activée, elle est disponible. `CombineArtifacts`

Pour des exemples de pipeline dans lesquels les builds par lots sont activés, voir CodePipeline Intégration avec CodeBuild et builds par lots.

CombineArtifacts

Obligatoire : non

La valeur booléenne du `CombineArtifacts` paramètre combine tous les artefacts de construction d'une génération par lots dans un seul fichier d'artefact pour l'action de génération.

Pour utiliser cette option, le `BatchEnabled` paramètre doit être activé.

EnvironmentVariables

Obligatoire : non

La valeur de ce paramètre est utilisée pour définir des variables d'environnement pour l'action CodeBuild dans votre pipeline. La valeur du paramètre `EnvironmentVariables` prend la forme d'un tableau JSON d'objets variables d'environnement. Consultez l'exemple de paramètre dans Déclaration d'action (CodeBuild exemple).

Chaque objet comporte trois parties, qui sont toutes des chaînes :

- `name` : Nom ou clé de la variable d'environnement.
- `value` : Valeur de la variable d'environnement. Lorsque vous utilisez le `SECRETS_MANAGER` type `PARAMETER_STORE` ou, cette valeur doit être le nom d'un paramètre que vous avez déjà stocké dans le magasin de paramètres de AWS Systems Manager ou d'un secret que vous avez déjà enregistré dans AWS Secrets Manager, respectivement.

  > **Note**
  >
  > Nous déconseillons vivement l'utilisation de variables d'environnement pour stocker des valeurs sensibles, en particulier des AWS informations d'identification. Lorsque vous utilisez la CodeBuild console ou la AWS CLI, les variables d'environnement sont affichées en texte brut. Pour les valeurs sensibles, nous vous recommandons d'utiliser plutôt le type `SECRETS_MANAGER`.

- `type` : (facultatif) Type de la variable d'environnement. Les valeurs valides sont `PARAMETER_STORE`, `SECRETS_MANAGER` ou `PLAINTEXT`. Si rien spécifié, la valeur par défaut est `PLAINTEXT`.

> **Note**
>
> Lorsque vous entrez la configuration `namevalue`, et `type` pour vos variables d'environnement, en particulier si la variable d'environnement contient une syntaxe de variable de CodePipeline sortie, ne dépassez pas la limite de 1 000 caractères pour le champ de valeur de la configuration. Une erreur de validation est renvoyée lorsque cette limite est dépassée.

Pour plus d'informations, consultez EnvironmentVariablela référence de AWS CodeBuild l'API. Pour un exemple CodeBuild d'action avec une variable d'environnement qui correspond au nom de la GitHub branche, consultezExemple : utilisation d'une BranchName variable avec des variables d' CodeBuild environnement.

Exemple de configuration de règles

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"
}
```

Consultez aussi

Les ressources connexes suivantes peuvent vous aider à utiliser cette règle.

- Pour plus d'informations sur les règles et conditions, consultez la section Condition et RuleExecutionle guide de l'CodePipeline API. RuleTypeId

Commandes

Lorsque vous créez une condition, vous pouvez ajouter la `Commands` règle. Cette section fournit une référence pour les paramètres des règles. Pour plus d'informations sur les règles et conditions, consultezComment fonctionnent les conditions scéniques ?.

Vous pouvez utiliser la `Commands` règle pour créer une condition dans laquelle les commandes réussies répondent aux critères de la règle, tels que le chemin de sortie et de fichier des commandes réussies pour une condition BeforeEntry.

> **Note**
>
> Pour les conditions BeforeEntry configurées avec le résultat Skip, seules les règles suivantes sont disponibles : `LambdaInvoke` et`VariableCheck`.

Rubriques

- Considérations relatives à la règle des commandes

- Autorisations de politique des rôles de service
- Type de règle
- Paramètres de configuration
- Exemple de configuration de règles
- Consultez aussi

Considérations relatives à la règle des commandes

Les considérations suivantes s'appliquent à la règle Commandes.

- La règle des commandes utilise CodeBuild des ressources similaires à l' CodeBuild action, tout en autorisant les commandes de l'environnement shell dans une instance de calcul virtuelle sans qu'il soit nécessaire d'associer ou de créer un projet de génération.

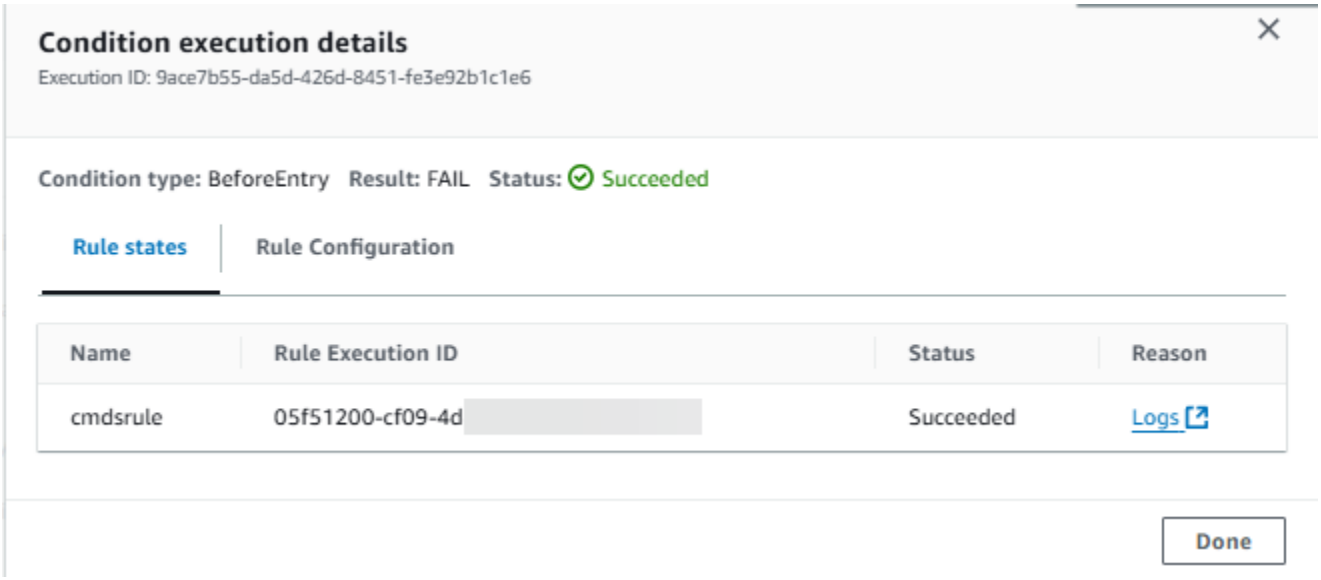
  > **Note**
  >
  > L'exécution de la règle des commandes entraînera des frais distincts. AWS CodeBuild

- Comme la règle Commands CodePipeline utilise CodeBuild des ressources, les builds exécutés par l'action seront attribués aux limites de build de votre compte dans CodeBuild. Les builds exécutés selon la règle Commands seront pris en compte dans les limites de build simultanées configurées pour ce compte.
- Le délai d'expiration des builds utilisant la règle Commands est de 55 minutes, selon les CodeBuild builds.
- L'instance de calcul utilise un environnement de construction isolé dans CodeBuild.

  > **Note**
  >
  > Étant donné que l'environnement de construction isolé est utilisé au niveau du compte, une instance peut être réutilisée pour une autre exécution de pipeline.

- Tous les formats sont pris en charge, à l'exception des formats multilignes. Vous devez utiliser le format d'une seule ligne lors de la saisie des commandes.
- Pour cette règle, CodePipeline assumera le rôle de service de pipeline et utilisera ce rôle pour autoriser l'accès aux ressources lors de l'exécution. Il est recommandé de configurer le rôle de service de manière à ce que les autorisations soient limitées au niveau de l'action.

- Les autorisations ajoutées au rôle CodePipeline de service sont détaillées dansAjouter des autorisations au rôle CodePipeline de service.
- Les autorisations nécessaires pour consulter les journaux dans la console sont détaillées dansAutorisations requises pour consulter les journaux de calcul dans la console. Dans les exemples d'écrans suivants, utilisez le lien Journaux pour afficher les journaux indiquant si une règle de commandes a été utilisée correctement dans les CloudWatch journaux.





- Contrairement aux autres actions dans CodePipeline, vous ne définissez pas de champs dans la configuration des actions ; vous définissez les champs de configuration des actions en dehors de la configuration des actions.

Autorisations de politique des rôles de service

Lors de l' CodePipeline exécution de la règle, CodePipeline crée un groupe de journaux en utilisant le nom du pipeline comme suit. Cela vous permet de limiter les autorisations de journalisation des ressources en utilisant le nom du pipeline.

```
/aws/codepipeline/MyPipelineName
```

Si vous utilisez un rôle de service existant, pour utiliser l'action Commandes, vous devez ajouter les autorisations suivantes pour le rôle de service.

- journaux : CreateLogGroup
- journaux : CreateLogStream
- journaux : PutLogEvents

Dans la déclaration de politique relative aux rôles de service, limitez les autorisations au niveau du pipeline, comme indiqué dans l'exemple suivant.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
}
```

Pour afficher les journaux dans la console à l'aide de la page de dialogue des détails de l'action, l'autorisation d'afficher les journaux doit être ajoutée au rôle de console. Pour plus d'informations, consultez l'exemple de politique d'autorisation de console dansAutorisations requises pour consulter les journaux de calcul dans la console.

Type de règle

- Catégorie : `Rule`
- Propriétaire : `AWS`

- Fournisseur : `Commands`
- Version : 1

Paramètres de configuration

Commandes

Obligatoire : oui

Vous pouvez fournir des commandes shell pour que la `Commands` règle s'exécute. Dans la console, les commandes sont entrées sur des lignes séparées. Dans la CLI, les commandes sont entrées sous forme de chaînes distinctes.

> **Note**
>
> Les formats multilignes ne sont pas pris en charge et un message d'erreur s'affichera. Le format d'une seule ligne doit être utilisé pour saisir des commandes dans le champ Commandes.

Les informations suivantes indiquent le calcul par défaut utilisé pour la règle Commands. Pour plus d'informations, consultez la section Référence des modes et types de calcul de l'environnement de génération dans le guide de CodeBuild l'utilisateur.

- CodeBuild image : aws/codebuild/amazonlinux 2-x86_64 standard : 5.0
- Type de calcul : Linux Small
- Valeur ComputeType de l'environnement : BUILD_ _SMALL GENERAL1
- Valeur du type d'environnement : LINUX_CONTAINER

Exemple de configuration de règles

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"
        }
    ]
}
```

Consultez aussi

Les ressources connexes suivantes peuvent vous aider à utiliser cette règle.

- Pour plus d'informations sur les règles et conditions, consultez la section Condition et RuleExecutionle guide de l'CodePipeline API. RuleTypeId

DeploymentWindow

Lorsque vous créez une condition, vous pouvez ajouter la `DeploymentWindow` règle. Cette section fournit une référence pour les paramètres des règles. Pour plus d'informations sur les règles et conditions, consultezComment fonctionnent les conditions scéniques ?.

Rubriques

- Type de règle
- Paramètres de configuration
- Exemple de configuration de règles
- Consultez aussi

Type de règle

- Catégorie : `Rule`
- Propriétaire : `AWS`
- Fournisseur : `DeploymentWindow`
- Version : `1`

Paramètres de configuration

Cron

Obligatoire : oui

Expression qui définit les jours et les heures auxquels le déploiement sera autorisé. Les expressions Cron sont composées de 6 champs obligatoires et d'un champ facultatif séparés par des espaces blancs. Les champs d'expression cron vous permettent de spécifier un modèle de planification avec une expression cron comme suit.

Nom de champ	Valeurs autorisées	Caractères spéciaux autorisés
Secondes	N/A	*
Minutes	0-59	, - * /
Heures	0-23	, - * /
Day-of-month	1-31	, - * ? / L W
Mois	1 à 12 ou JAN-DEC	, - * /
Day-of-Week	1 à 7 ou SUN-SAT	, - * ? / L #
Année (facultatif)	vide, 1970-2199	, - * /

- Le caractère « * » est utilisé pour spécifier toutes les valeurs. Par exemple, « * » dans le champ des minutes signifie « toutes les minutes ».
- Le « ? » Un caractère est autorisé pour les champs jour du mois et jour de la semaine. Il est utilisé pour spécifier « aucune valeur spécifique ». Ceci est utile lorsque vous devez spécifier quelque chose dans l'un des deux champs, mais pas dans l'autre.
- Le caractère « - » est utilisé pour spécifier des plages. Par exemple, « 10-12 » dans le champ horaire signifie « les heures 10, 11 et 12 ».
- Le caractère «, » est utilisé pour spécifier des valeurs supplémentaires. Par exemple, « LUNDI, MERCREDI, VENDREDI » dans le champ du jour de la semaine signifie « les jours lundi, mercredi et vendredi ».
- Le caractère «/» est utilisé pour spécifier les incréments. Par exemple, 0/15 « » dans le champ des secondes signifie « les secondes 0, 15, 30 et 45 ». Et « 5/15 » dans le champ des secondes signifie « les secondes 5, 20, 35 et 50 ». Spécifier « * » avant le «/» revient à spécifier 0 comme valeur de départ.
- Le caractère « L » est autorisé pour les champs du jour du mois et du jour de la semaine. Ce caractère est l'abréviation de « dernier », mais il a une signification différente dans chacun des deux champs. Par exemple, la valeur « L » dans le champ « jour du mois » signifie « le dernier jour du mois » : 31 pour janvier, 28 pour février pour les années non bissextiles. S'il est utilisé seul dans le champ du jour de la semaine, cela signifie simplement « 7 » ou « SAT ». Mais

s'il est utilisé dans le champ du jour de la semaine après une autre valeur, cela signifie « le dernier <specified_day>jour du mois ». Par exemple, « 6L » signifie « le dernier vendredi du mois ». Vous pouvez également spécifier un décalage par rapport au dernier jour du mois, tel que « L-3 », ce qui signifierait l'avant-dernier jour du mois civil.

- Le caractère « W » est autorisé pour le champ du jour du mois. Ce caractère est utilisé pour spécifier le jour de la semaine (Monday-Friday) le plus proche du jour donné. Par exemple, si vous deviez spécifier « 15W » comme valeur pour le champ du jour du mois, cela signifie : « le jour de la semaine le plus proche du 15 du mois ». Donc, si le 15 est un samedi, la gâchette se déclenchera le vendredi 14. Si le 15 est un dimanche, la gâchette se déclenchera le lundi 16. Si le 15 est un mardi, il se déclenchera le mardi 15.
- Les caractères « L » et « W » peuvent également être combinés pour l'expression du jour du mois afin d'obtenir « LW », qui se traduit par « dernier jour de semaine du mois ».
- Le caractère « # » est autorisé pour le champ du jour de la semaine. Ce caractère est utilisé pour spécifier « le nième » <specified_day>jour du mois. Par exemple, la valeur « 6 #3 » dans le champ du jour de la semaine signifie le troisième vendredi du mois (jour 6 = vendredi et « #3 » = le troisième du mois).
- Les caractères juridiques et les noms des mois et des jours de la semaine ne font pas la distinction entre majuscules et minuscules.

TimeZone

Obligatoire : non

Fuseau horaire de la fenêtre de déploiement. L'expression régulière correspond à des modèles dans les formats suivants :

- Region/City format. La valeur correspond à un fuseau horaire au format Region/City Region/City _City. Par exemple, `America/New_York` ou `Europe/Berlin`.
- Format UTC. La valeur correspond à la chaîne UTC éventuellement suivie d'un décalage au format + HH:MM ou -HH:MM. Par exemple, `UTC`, `UTC+05:30` ou `UTC-03:00`. Il s'agit du format par défaut si le paramètre n'est pas défini autrement.
- Format d'abréviation. La valeur correspond à une abréviation de trois à cinq caractères pour un fuseau horaire. Par exemple, `EST` ou `IST`.

Pour obtenir un tableau des valeurs TimeZone d'ID valides, consultez https://docs.oracle.com/middleware/1221/wcs/tag-ref/MISC/TimeZones.html. Notez que certaines abréviations sont des abréviations dupliquées, comme CST pour l'heure normale du centre, l'heure normale de Chine et l'heure normale de Cuba.

Exemple de configuration de règles

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"
                    }
                ]
```

Consultez aussi

Les ressources connexes suivantes peuvent vous aider à utiliser cette règle.

- Créer des conditions de réussite— Cette section décrit les étapes à suivre pour créer une condition En cas de réussite avec une règle de fenêtre de déploiement.
- Pour plus d'informations sur les règles et conditions, consultez la section Condition RuleTypeId, et RuleExecution dans le Guide de l'CodePipeline API.

LambdaInvoke

Lorsque vous créez une condition, vous pouvez ajouter la `LambdaInvoke` règle. Cette section fournit une référence pour les paramètres des règles. Pour plus d'informations sur les règles et conditions, consultezComment fonctionnent les conditions scéniques ?.

Vous devez déjà avoir créé une fonction dans Lambda en tant que ressource distincte.

Rubriques

- Type de règle
- Paramètres de configuration
- Exemple de configuration de règles
- Consultez aussi

Type de règle

- Catégorie : `Rule`
- Propriétaire : `AWS`
- Fournisseur : `LambdaInvoke`
- Version : `1`

Paramètres de configuration

FunctionName

Obligatoire : oui

Nom de la fonction Lambda.

UserParameters

Obligatoire : non

Il s'agit de paramètres fournis en entrée pour la fonction sous forme de paire clé-valeur.

Exemple de configuration de règles

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"
    }
]
```

Consultez aussi

Les ressources connexes suivantes peuvent vous aider à utiliser cette règle.

- Création de conditions en cas de défaillance— Cette section décrit les étapes à suivre pour créer une condition En cas de défaillance avec une règle d'alarme.

VariableCheck

Lorsque vous créez une condition, vous pouvez ajouter la `VariableCheck` règle. Cette section fournit une référence pour les paramètres des règles. Pour plus d'informations sur les règles et conditions, consultezComment fonctionnent les conditions scéniques ?.

Vous pouvez utiliser la `VariableCheck` règle pour créer une condition dans laquelle la variable de sortie est vérifiée par rapport à une expression fournie. La règle passe le contrôle lorsque la valeur de la variable répond aux critères de la règle, tels que la valeur égale ou supérieure à une variable de sortie spécifiée.

Rubriques

- Type de règle
- Paramètres de configuration
- Exemple de configuration de règles
- Consultez aussi

Type de règle

- Catégorie : `Rule`
- Propriétaire : `AWS`
- Fournisseur : `VariableCheck`
- Version : `1`

Paramètres de configuration

Opérateur

Obligatoire : oui

L'opérateur qui indique l'opération à effectuer pour le contrôle des variables.

Dans l'exemple suivant, la variable de sortie pour le nom du référentiel sera vérifiée pour savoir si elle est égale à`MyDemoRepo`.

```
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
```

Les opérateurs suivants sont disponibles pour créer une expression comme suit.

- Égal - Choisissez cet opérateur pour vérifier si la variable est égale à la valeur de la chaîne.

  Paramètre CLI : `EQ`
- Contient : choisissez cet opérateur pour vérifier si la variable contient la valeur de la chaîne sous forme de sous-chaîne.

  Paramètre CLI : `CONTAINS`
- Correspondances : choisissez cet opérateur pour vérifier si la variable correspond à une expression régulière donnée en tant que valeur de chaîne.

  Toutes les expressions régulières CodePipeline sont conformes à la syntaxe Java regex. Pour une description complète de cette syntaxe et de ses constructions, consultez java.util.regex.Pattern.

  Paramètre CLI : `MATCHES`
- Non égal - Choisissez cet opérateur pour vérifier si la variable n'est pas égale à la valeur de la chaîne.

  Paramètre CLI : `NE`

Variable

Obligatoire : oui

Les variables de pipeline à vérifier.

Value

Obligatoire : oui

La valeur de l'expression à vérifier.

Dans l'exemple suivant, la variable de sortie pour le nom du référentiel sera vérifiée pour savoir si elle est égale à`MyDemoRepo`.

```
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
```

Dans l'exemple JSON suivant, deux règles distinctes sont définies, l'une pour une instruction `EQ` (equals) qui vérifie le dépôt et le nom de branche au format # {SourceVariables. RepositoryName} et un pour `CONTAINS` cela vérifie la variable de sortie du message de validation formatée sous la forme # {SourceVariables. CommitMessage} contre la valeur « update » fournie.

```
"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"
                            }
                        ]
                    }
                ]
            }
        }
    ],
```

Exemple de configuration de règles

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"
    }
]
```

Consultez aussi

Les ressources connexes suivantes peuvent vous aider à utiliser cette règle.

- Tutoriel : Création d'une règle de vérification des variables pour un pipeline en tant que condition d'entrée— Cette section fournit un didacticiel expliquant les étapes à suivre pour créer une condition On Entry avec une règle de vérification des variables.
- Référence des variables— Cette section fournit des informations de référence et des exemples pour les variables de pipeline.
- Pour plus d'informations sur les règles et conditions, consultez la section Condition et RuleExecutionle guide de l'CodePipeline API. RuleTypeId

Référence du modèle d'intégration

Il existe plusieurs intégrations prédéfinies pour les services tiers afin de faciliter l'intégration des outils clients existants dans le processus de publication du pipeline. Les partenaires ou les fournisseurs de services tiers utilisent un modèle d'intégration pour implémenter les types d'actions à utiliser dans CodePipeline.

Utilisez cette référence lorsque vous planifiez ou travaillez avec des types d'actions gérés à l'aide d'un modèle d'intégration pris en charge dans CodePipeline.

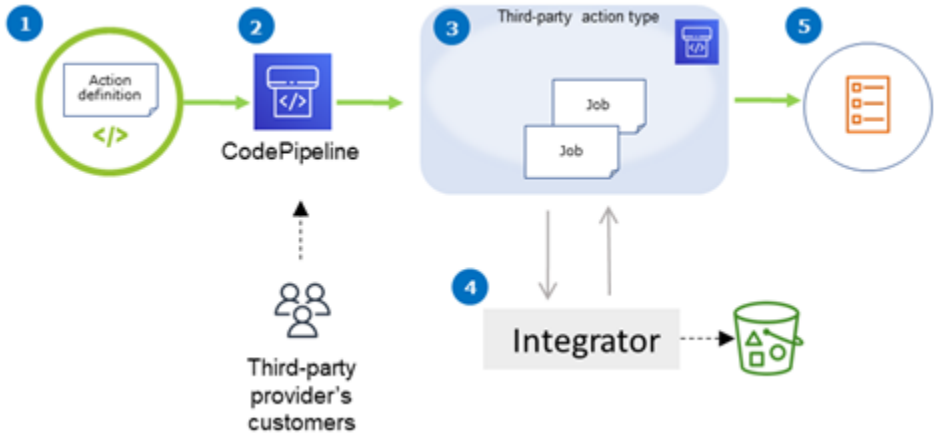
Pour certifier votre type d'action tiers en tant que partenaire d'intégration avec CodePipeline, reportez-vous au AWS Partner Network (APN). Ces informations constituent un complément à la AWS CLI Référence.

Rubriques

- Comment les types d'actions tiers fonctionnent avec l'intégrateur
- Concepts
- Modèles d'intégration pris en charge
- Modèle d'intégration Lambda
- Modèle d'intégration des travailleurs

Comment les types d'actions tiers fonctionnent avec l'intégrateur

Vous pouvez ajouter des types d'actions tiers aux pipelines clients pour effectuer des tâches sur les ressources des clients. L'intégrateur gère les demandes de travail et exécute l'action avec CodePipeline. Le schéma suivant montre un type d'action tiers créé pour que les clients puissent l'utiliser dans leur pipeline. Une fois que le client a configuré l'action, celle-ci s'exécute et crée des demandes de travail qui sont gérées par le moteur d'action de l'intégrateur.



Le schéma montre les étapes suivantes :

1. La définition de l'action est enregistrée et mise à disposition dans CodePipeline. L'action tierce est disponible pour les clients du fournisseur tiers.
2. Le client du fournisseur choisit et configure l'action dans CodePipeline.
3. L'action s'exécute et les jobs sont mis en file d'attente. CodePipeline Lorsque la tâche est prête CodePipeline, elle envoie une demande d'emploi.
4. L'intégrateur (le responsable des tâches pour les API de sondage tierces ou la fonction Lambda) récupère la demande de tâche, renvoie une confirmation et travaille sur les artefacts pour les actions.
5. L'intégrateur renvoie une success/failure sortie (le job worker utilise des success/failure API ou la fonction Lambda envoie une success/failure sortie) avec un résultat de tâche et un jeton de continuation.

Pour plus d'informations sur les étapes que vous pouvez suivre pour demander, afficher et mettre à jour un type d'action, consultezUtilisation des types d'actions.

Concepts

Cette section utilise les termes suivants pour les types d'actions tiers :

Type d'action

> Un processus reproductible qui peut être réutilisé dans des pipelines qui exécutent les mêmes charges de travail de livraison continue. Les types d'action sont identifiés par un `OwnerCategory`,`Provider`, et`Version`. Par exemple :

```
        {

            "Category": "Deploy",
            "Owner": "AWS",
            "Provider": "CodeDeploy",
            "Version": "1"
        },
```

Toutes les actions du même type partagent la même implémentation.

Action

Une instance unique d'un type d'action, l'un des processus discrets qui se produisent au cours d'une étape d'un pipeline. Cela inclut généralement les valeurs utilisateur spécifiques au pipeline dans lequel cette action s'exécute.

Définition de l'action

Schéma d'un type d'action qui définit les propriétés requises pour configurer l'action et les input/output artefacts.

Exécution de l'action

Un ensemble de tâches qui ont été exécutées pour déterminer si l'action sur le pipeline du client a été couronnée de succès ou non.

Moteur d'exécution d'actions

Propriété de la configuration d'exécution des actions qui définit le type d'intégration utilisé par un type d'action. Les valeurs valides sont `JobWorker` et `Lambda`.

Integration

Décrit un logiciel exécuté par un intégrateur pour implémenter un type d'action. CodePipeline prend en charge deux types d'intégration correspondant aux deux moteurs d'action pris en charge `JobWorker` et`Lambda`.

Intégrateur

Personne responsable de la mise en œuvre d'un type d'action.

Tâche

Un travail tenant compte du contexte du pipeline et du client pour exécuter une intégration. L'exécution d'une action est composée d'une ou de plusieurs tâches.

Travailleur

Service qui traite les données saisies par le client et exécute une tâche.

Modèles d'intégration pris en charge

CodePipeline dispose de deux modèles d'intégration :

- Modèle d'intégration Lambda : ce modèle d'intégration est la méthode préférée pour utiliser les types d'actions dans CodePipeline. Le modèle d'intégration Lambda utilise une fonction Lambda pour traiter les demandes de travail lorsque votre action est exécutée.
- Modèle d'intégration des travailleurs : Le modèle d'intégration des travailleurs est le modèle précédemment utilisé pour les intégrations tierces. Le modèle d'intégration des assistants de travail utilise un assistant de travail configuré pour contacter les CodePipeline API afin de traiter les demandes de travail lorsque votre action est exécutée.

À titre de comparaison, le tableau suivant décrit les caractéristiques des deux modèles :

	Modèle d'intégration Lambda	Modèle d'intégration des travailleurs
Description	L'intégrateur écrit l'intégration sous la forme d'une fonction Lambda, qui est invoquée CodePipeline chaque fois qu'une tâche est disponible pour l'action. La fonction Lambda n'interroge pas les offres d'emploi disponibles, mais attend la réception de la prochaine demande de tâche.	L'intégrateur écrit l'intégration en tant que collaborateur qui interroge en permanence les offres d'emploi sur les pipelines du client. L'assistant exécute ensuite la tâche et renvoie le résultat de la tâche à l'aide CodePipeline d' CodePipeline API.
Infrastructures	AWS Lambda	Déployez du code Job Worker sur l'infrastructure de l'intégrateur, comme les instances Amazon EC2.

	Modèle d'intégration Lambda	Modèle d'intégration des travailleurs
Effort de développement	L'intégration contient uniquement la logique métier.	L'intégration doit interagir avec les CodePipeline API en plus de contenir la logique métier.
Effort opérationnel	Effort opérationnel moindre puisque l'infrastructure n'est que AWS des ressources.	Effort opérationnel plus important car le travailleur a besoin de son matériel autonome.
Durée maximale d'exécution des tâches	Si l'intégration doit s'exécuter activement pendant plus de 15 minutes, ce modèle ne peut pas être utilisé. Cette action est destinée aux intégrateurs qui doivent démarrer un processus (par exemple, lancer une compilation à partir de l'artefact de code du client) et renvoyer un résultat lorsqu'il est terminé. Nous ne recommandons pas à l'intégrateur de continuer à attendre la fin de la compilation. Renvoie plutôt une suite. CodePipelinecrée une nouvelle tâche en 30 secondes supplémentaires si une suite est reçue du code de l'intégrateur pour vérifier la tâche jusqu'à sa fin.	Les tâches de très longue durée (hours/days) peuvent être maintenues à l'aide de ce modèle.

Modèle d'intégration Lambda

Le modèle d'intégration Lambda pris en charge inclut la création de la fonction Lambda et la définition de la sortie pour le type d'action tiers.

Mettez à jour votre fonction Lambda pour gérer les entrées de CodePipeline

Vous pouvez créer une nouvelle fonction Lambda. Vous pouvez ajouter une logique métier à votre fonction Lambda qui est exécutée chaque fois qu'une tâche est disponible sur votre pipeline pour votre type d'action. Par exemple, compte tenu du contexte du client et du pipeline, vous souhaiterez peut-être commencer à intégrer votre service au client.

Utilisez les paramètres suivants pour mettre à jour votre fonction Lambda afin de gérer l'entrée de CodePipeline.

Format :

- `jobId`:
  - L'identifiant unique de la tâche généré par le système.
  - Type : Chaîne
  - Motif : [0-9a-f] {8} - [0-9a-f] {4} - [0-9a-f] {4} - [0-9a-f] {4} - [0-9a-f] {12}
- `accountId`:
  - L'identifiant du AWS compte du client à utiliser lors de l'exécution de la tâche.
  - Type : Chaîne
  - Motif : [0-9] {12}
- `data`:
  - Autres informations relatives à une tâche qu'une intégration utilise pour terminer la tâche.
  - Contient une carte des éléments suivants :
    - `actionConfiguration`:
      - Les données de configuration de l'action. Les champs de configuration des actions sont un mappage de paires clé-valeur permettant à votre client de saisir des valeurs. Les clés sont déterminées par les paramètres clés contenus dans le fichier de définition du type d'action lorsque vous configurez votre action. Dans cet exemple, les valeurs sont déterminées par l'utilisateur de l'action en spécifiant les informations dans les `Password` champs `Username` et.
      - Type : mappage chaîne à chaîne, éventuellement présent

        Exemple :

```
    "configuration": {
```

```
        "Username": "MyUser",
        "Password": "MyPassword"
    },
```

- `encryptionKey`:
  - Représente les informations relatives à la clé utilisée pour chiffrer les données du magasin d'artefacts, telle qu'une AWS KMS clé.
  - Contenu : Type du type de données`encryptionKey`, éventuellement présent
- `inputArtifacts`:
  - Liste des informations relatives à un artefact sur lequel travailler, comme un artefact de test ou de construction.
  - Contenu : Liste des types de données`Artifact`, éventuellement présents
- `outputArtifacts`:
  - Liste des informations relatives au résultat d'une action.
  - Contenu : Liste des types de données`Artifact`, éventuellement présents
- `actionCredentials`:
  - Représente un objet d'informations d'identification de AWS session. Ces informations d'identification sont des informations d'identification temporaires émises par AWS STS. Ils peuvent être utilisés pour accéder aux artefacts d'entrée et de sortie dans le compartiment S3 utilisé pour stocker les artefacts du pipeline CodePipeline.

    Ces informations d'identification disposent également des mêmes autorisations que le modèle de déclaration de politique spécifié dans le fichier de définition du type d'action.
  - Contenu : Type du type de données`AWSSessionCredentials`, éventuellement présent
- `actionExecutionId`:
  - L'ID externe de l'exécution de l'action.
  - Type : Chaîne
- `continuationToken`:
  - Un jeton généré par le système, tel qu'un ID de déploiement, requis par une tâche pour poursuivre la tâche de manière asynchrone.
  - Type : chaîne, éventuellement présente

Types de données :

- `encryptionKey`:
  - `id`:
    - ID utilisé pour identifier la clé. Pour chaque AWS KMS clé, vous pouvez utiliser l'ID de clé, l'ARN de clé ou l'alias ARN.
    - Type : Chaîne
  - `type`:
    - Type de clé de chiffrement, telle qu'une AWS KMS clé.
    - Type : Chaîne
    - Valeurs valides : `KMS`
- `Artifact`:
  - `name`:
    - Le nom de l'artefact.
    - Type : chaîne, éventuellement présente
  - `revision`:
    - L'ID de révision de l'artefact. Selon le type d'objet, il peut s'agir d'un ID de validation (GitHub) ou d'un ID de révision (Amazon S3).
    - Type : chaîne, éventuellement présente
  - `location`:
    - L'emplacement d'un artefact.
    - Contenu : Type du type de données`ArtifactLocation`, éventuellement présent
- `ArtifactLocation`:
  - `type`:
    - Type d'artefact présent dans le lieu.
    - Type : chaîne, éventuellement présente
    - Valeurs valides : `S3`
  - `s3Location`:
    - Emplacement du compartiment S3 qui contient une révision.
    - Contenu : Type du type de données`S3Location`, éventuellement présent
- `S3Location`:
  - `bucketName`:
    - Le nom du compartiment S3.

    - Type : Chaîne
  - `objectKey`:
    - La clé de l'objet dans le compartiment S3, qui identifie de manière unique l'objet dans le compartiment.
    - Type : Chaîne
- `AWSSessionCredentials`:
  - `accessKeyId`:
    - La clé d'accès à la session.
    - Type : Chaîne
  - `secretAccessKey`:
    - La clé d'accès secrète pour la session.
    - Type : Chaîne
  - `sessionToken`:
    - Le jeton de la session.
    - Type : Chaîne

Exemple :

```
{
    "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"
    }
}
```

Renvoie les résultats de votre fonction Lambda à CodePipeline

La ressource Job Worker de l'intégrateur doit renvoyer une charge utile valide en cas de succès, d'échec ou de poursuite.

Format :

- `result`: Le résultat du travail.
  - Obligatoire
  - Valeurs valides (sans distinction majuscules/minuscules) :
    - `Success`: indique qu'une tâche est réussie et terminée.
    - `Continue`: indique qu'une tâche est réussie et doit se poursuivre, par exemple si le collaborateur chargé du travail est réinvoqué pour exécuter la même action.
    - `Fail`: indique qu'une tâche a échoué et qu'elle est terminée.

- `failureType`: type d'échec à associer à une tâche ayant échoué.

  La `failureType` catégorie des actions des partenaires décrit le type d'échec rencontré lors de l'exécution de la tâche. Les intégrateurs définissent le type ainsi que le message d'échec lorsqu'ils renvoient le résultat d'un échec de tâche à CodePipeline.

  - Facultatif. Obligatoire si le résultat l'est`Fail`.
  - Doit être nul si `result` c'est `Success` ou `Continue`
  - Valeurs valides :
    - ConfigurationError
    - JobFailed
    - PermissionsError
    - RevisionOutOfSync
    - RevisionUnavailable
    - SystemUnavailable
- `continuation`: état de continuation à transmettre à la tâche suivante au cours de l'exécution de l'action en cours.
  - Facultatif. Obligatoire si le résultat l'est`Continue`.
  - Doit être nul si `result` c'est `Success` ou`Fail`.
  - Propriétés :
    - `State`: Un hash de l'état à transmettre.
- `status`: État de l'exécution de l'action.
  - Facultatif.
  - Propriétés :
    - `ExternalExecutionId`: ID d'exécution externe ou ID de validation facultatif à associer à la tâche.
    - `Summary`: résumé facultatif de ce qui s'est passé. Dans les scénarios de défaillance, cela devient le message d'échec que l'utilisateur voit.
- `outputVariables`: ensemble de key/value paires à transmettre à la prochaine exécution de l'action.
  - Facultatif.
  - Doit être nul si `result` c'est `Continue` ou`Fail`.

Exemple :

```
{
    "result": "success",
    "failureType": null,
    "continuation": null,
    "status": {
        "externalExecutionId": "my-commit-id-123",
        "summary": "everything is dandy"
    },
    "outputVariables": {
        "FirstOne": "Nice",
        "SecondOne": "Nicest",
        ...
    }
}
```

Utilisez des jetons de continuation pour attendre les résultats d'un processus asynchrone

Le `continuation` jeton fait partie de la charge utile et du résultat de votre fonction Lambda. C'est un moyen de transmettre l'état du poste CodePipeline et d'indiquer que le travail doit être poursuivi. Par exemple, une fois qu'un intégrateur a démarré une compilation pour le client sur sa ressource, il n'attend pas que la génération soit terminée, mais indique CodePipeline qu'il n'a pas de résultat final en renvoyant le `result` as `continue` et en renvoyant l'identifiant unique de la construction CodePipeline sous forme de `continuation` jeton.

> **Note**
>
> Les fonctions Lambda ne peuvent être exécutées que pendant 15 minutes. Si la tâche doit être exécutée plus longtemps, vous pouvez utiliser des jetons de continuation.

L' CodePipeline équipe invoque l'intégrateur au bout de 30 secondes avec le même `continuation` jeton dans sa charge utile afin qu'il puisse vérifier qu'il est terminé. Si la compilation est terminée, l'intégrateur renvoie le success/fail résultat du terminal, sinon il continue.

Fournissez CodePipeline les autorisations nécessaires pour invoquer la fonction Lambda de l'intégrateur lors de l'exécution

Vous ajoutez des autorisations à la fonction Lambda de votre intégrateur pour fournir au CodePipeline service les autorisations nécessaires pour l'invoquer à l'aide du principal de CodePipeline service :. `codepipeline.amazonaws.com` Vous pouvez ajouter des autorisations à l'aide de la ligne de commande CloudFormation ou à l'aide de la ligne de commande. Pour obtenir un exemple, consultez Utilisation des types d'actions.

Modèle d'intégration des travailleurs

Après avoir conçu votre flux de travail de haut niveau, vous pouvez créer votre assistant de travail. Bien que les spécificités de l'action tierce déterminent ce dont le collaborateur a besoin, la plupart des collaborateurs chargés des actions tierces incluent les fonctionnalités suivantes :

- Sondage pour les offres d'emploi liées à CodePipeline l'utilisation`PollForThirdPartyJobs`.
- Reconnaître les emplois et renvoyer les résultats à CodePipeline l'utilisation de `AcknowledgeThirdPartyJob`PutThirdPartyJobSuccessResult, et`PutThirdPartyJobFailureResult`.
- Récupération d'artefacts en les and/or plaçant dans le compartiment Amazon S3 pour le pipeline. Pour télécharger des artefacts depuis le compartiment Amazon S3, vous devez créer un client Amazon S3 qui utilise la signature Signature Version 4 (Sig V4). Le sig V4 est requis pour AWS KMS.

  Pour charger des artefacts dans le compartiment Amazon S3, vous devez également configurer la `PutObject` demande Amazon S3 de manière à utiliser le chiffrement via AWS Key Management Service (AWS KMS). AWS KMS utilisations AWS KMS keys. Afin de savoir s'il doit utiliser la clé gérée par le client Clé gérée par AWS ou une clé gérée par le client pour télécharger des artefacts, votre collaborateur doit examiner les données de la tâche et vérifier la propriété de la clé de cryptage. Si la propriété est définie, vous devez utiliser cet ID de clé géré par le client lors de la configuration AWS KMS. Si la propriété key est nulle, vous utilisez le Clé gérée par AWS. CodePipeline utilise le Clé gérée par AWS sauf configuration contraire.

  Pour un exemple qui montre comment créer les AWS KMS paramètres en Java ou .NET, voir Spécifier les paramètres AWS Key Management Service dans Amazon S3 à l'aide AWS des kits SDK. Pour plus d'informations sur le compartiment Amazon S3 pour CodePipeline, consultezCodePipeline concepts.

Choix et configuration d'une stratégie de gestion des autorisations pour votre exécutant de tâches

Pour développer un assistant de travail pour votre action tierce CodePipeline, vous avez besoin d'une stratégie d'intégration de la gestion des utilisateurs et des autorisations.

La stratégie la plus simple consiste à ajouter l'infrastructure dont vous avez besoin pour votre collaborateur en créant des instances Amazon EC2 dotées d'un rôle d'instance Gestion des identités et des accès AWS (IAM), ce qui vous permet d'augmenter facilement les ressources dont vous avez besoin pour votre intégration. Vous pouvez utiliser l'intégration intégrée AWS pour simplifier l'interaction entre votre collaborateur et CodePipeline.

Apprenez-en plus sur Amazon EC2 et déterminez s'il s'agit du bon choix pour votre intégration. Pour plus d'informations, consultez Amazon EC2 - Hébergement de serveurs virtuels. Pour plus d'informations sur la configuration d'une instance Amazon EC2, consultez la section Démarrage avec les instances Linux Amazon EC2.

Une autre stratégie à envisager consiste à utiliser la fédération d'identités avec IAM pour intégrer le système et les ressources de votre fournisseur d'identité existants. Cette stratégie est utile si vous disposez déjà d'un fournisseur d'identité d'entreprise ou si vous êtes déjà configuré pour aider les utilisateurs utilisant des fournisseurs d'identité Web. La fédération d'identités vous permet d'accorder un accès sécurisé aux AWS ressources CodePipeline, notamment sans avoir à créer ou à gérer des utilisateurs IAM. Vous pouvez utiliser ces fonctionnalités et ces stratégies pour mettre en place des mots de passe à des fins de sécurité et pour créer une rotation des informations d'identification. Vous pouvez vous appuyer sur des modèles d'application pour créer votre propre modèle. Pour plus d'informations, consultez Gestion de fédération.

Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

- Utilisateurs et groupes dans AWS IAM Identity Center :

  Créez un jeu d'autorisations. Suivez les instructions de la rubrique Création d'un jeu d'autorisations du Guide de l'utilisateur AWS IAM Identity Center .
- Utilisateurs gérés dans IAM par un fournisseur d'identité :

  Créez un rôle pour la fédération d'identité. Suivez les instructions de la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM.
- Utilisateurs IAM :

- Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM dans le Guide de l'utilisateur IAM.
- (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

Voici un exemple de politique que vous pourriez créer pour une utilisation avec votre assistant de travail tiers. Cette stratégie n'est fournie qu'à titre d'exemple, et « telle quelle ».

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/"

      ]
    }
  ]
}
```

Référence pour les fichiers de définitions d'image

Cette section est fournie à des fins de référence uniquement. Pour de plus amples informations sur la création d'un pipeline avec des actions source ou de déploiement pour les conteneurs, veuillez consulter Création d'un pipeline, d'étapes et d'actions.

AWS CodePipeline les collaborateurs chargés des actions de conteneur, telles qu'une action source Amazon ECR ou des actions de déploiement Amazon ECS, utilisent des fichiers de définitions pour mapper l'URI de l'image et le nom du conteneur à la définition de la tâche. Chaque fichier de définitions est un JSON-formatted fichier utilisé par le fournisseur d'actions comme suit :

- Les déploiements standard d'Amazon ECS nécessitent un `imagedefinitions.json` fichier comme entrée pour l'action de déploiement. Pour un didacticiel utilisant l'action de déploiement standard d'Amazon ECS dans CodePipeline, consultezTutoriel : Déploiement standard d'Amazon ECS avec CodePipeline. Pour un autre exemple de didacticiel qui utilise l'action de déploiement standard Amazon ECS CodePipeline en même temps que l' ECRBuildAndPublish action, consultezTutoriel : Créez et transférez une image Docker vers Amazon ECR avec CodePipeline (type V2).
- Les blue/green déploiements Amazon ECS nécessitent un `imageDetail.json` fichier comme entrée pour l'action de déploiement. Pour un didacticiel présentant un exemple blue/green de déploiement, consultezTutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR.
  - Les actions source Amazon ECR génèrent un fichier `imageDetail.json` qui est fourni en tant que sortie de l'action source.

Rubriques

- fichier imagedefinitions.json pour les actions de déploiement standard d'Amazon ECS
- Detail.json fichier image pour les actions de blue/green déploiement d'Amazon ECS

fichier imagedefinitions.json pour les actions de déploiement standard d'Amazon ECS

Un document de définitions d'images est un fichier JSON qui décrit le nom de votre conteneur Amazon ECS, ainsi que l'image et la balise. Si vous déployez des applications basées sur des conteneurs, vous devez générer un fichier de définitions d'images pour fournir au collaborateur

CodePipeline le conteneur Amazon ECS et l'identification de l'image à récupérer dans le référentiel d'images, tel qu'Amazon ECR.

> **Note**
>
> Le nom de fichier par défaut pour le fichier est `imagedefinitions.json`. Si vous choisissez d'utiliser un autre nom de fichier, vous devez l'indiquer lorsque vous créez la phase de déploiement du pipeline.

Créez le fichier `imagedefinitions.json` en respectant les considérations suivantes :

- Le fichier doit utiliser un UTF-8 encodage.
- La taille maximum du fichier de définitions d'image est de 100 Ko.
- Vous devez créer le fichier en tant que source ou artefact de génération afin qu'il soit un artefact d'entrée pour l'action de déploiement. En d'autres termes, assurez-vous que le fichier est soit téléchargé vers votre emplacement source, tel que votre CodeCommit référentiel, soit généré en tant qu'artefact de sortie intégré.

Le fichier `imagedefinitions.json` fournit le nom du conteneur et l'URI de l'image. Il doit être construit avec l'ensemble de paires clé-valeur suivant.

Clé	Valeur
name	*container_name*
imageUri	*imageUri*

> **Note**
>
> Le champ name est utilisé pour le nom de l'image du conteneur, c'est-à-dire le nom de l'image Docker.

Voici la structure JSON, dans laquelle le nom du conteneur est `sample-app`, l'URI d'image est `ecs-repo` et la balise est `latest` :

```
[
  {
    "name": "sample-app",
    "imageUri": "11111EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest"
  }
]
```

Vous pouvez également construire le fichier de manière à répertorier plusieurs paires image-conteneur.

Structure JSON :

```
[
  {
    "name": "simple-app",
    "imageUri": "httpd:2.4"
  },
  {
    "name": "simple-app-1",
    "imageUri": "mysql"
  },
  {
    "name": "simple-app-2",
    "imageUri": "java1.8"
  }
]
```

Avant de créer votre pipeline, suivez les étapes suivantes pour configurer le fichier `imagedefinitions.json`.

1. Dans le cadre de la planification du déploiement d'application basé sur conteneur pour votre pipeline, planifiez l'étape source et l'étape de build, le cas échéant.
2. Sélectionnez l'une des méthodes suivantes :

    a. Si votre pipeline est créé de telle sorte qu'il ignore l'étape de génération, vous devez créer manuellement le fichier JSON et le télécharger dans votre référentiel source afin que l'action source puisse fournir l'artefact. Créez le fichier à l'aide de l'éditeur de texte et nommez-le ou utilisez le nom de fichier `imagedefinitions.json` par défaut. Disposez le fichier de définitions d'image dans votre référentiel source.

> **Note**
>
> Si votre référentiel source est un compartiment Amazon S3, pensez à compresser le fichier JSON.

b. Si votre pipeline comporte une phase de construction, ajoutez une commande à votre fichier buildspec qui affiche le fichier de définitions d'images dans votre référentiel source pendant la phase de génération. L'exemple suivant utilise la commande printf pour créer un fichier `imagedefinitions.json`. Répertoriez cette commande dans la section `post_build` du fichier `buildspec.yml` :

```
printf '[{"name":"container_name","imageUri":"image_URI"}]' >
imagedefinitions.json
```

Vous devez inclure le fichier de définitions d'image en tant qu'artefact de sortie dans le fichier `buildspec.yml`.

3. Lors de la création de votre pipeline dans la console, vous devez saisir le nom du fichier de définitions d'image dans le champ Nom du fichier image de la page Déploiement de l'assistant Créer un pipeline.

Pour un didacticiel étape par étape sur la création d'un pipeline utilisant Amazon ECS comme fournisseur de déploiement, consultez Tutoriel : Déploiement continu avec CodePipeline.

Detail.json fichier image pour les actions de blue/green déploiement d'Amazon ECS

Un `imageDetail.json` document est un fichier JSON qui décrit l'URI de votre image Amazon ECS. Si vous déployez des applications basées sur des conteneurs pour un blue/green déploiement, vous devez générer le `imageDetail.json` fichier pour fournir à Amazon ECS et à l'assistant CodeDeploy technique l'identification de l'image à récupérer dans le référentiel d'images, tel qu'Amazon ECR.

> **Note**
>
> Le nom du fichier doit être `imageDetail.json`.

Pour une description de l'action et de ses paramètres, consultezAmazon Elastic Container Service et référence des actions de déploiement CodeDeploy bleu-vert.

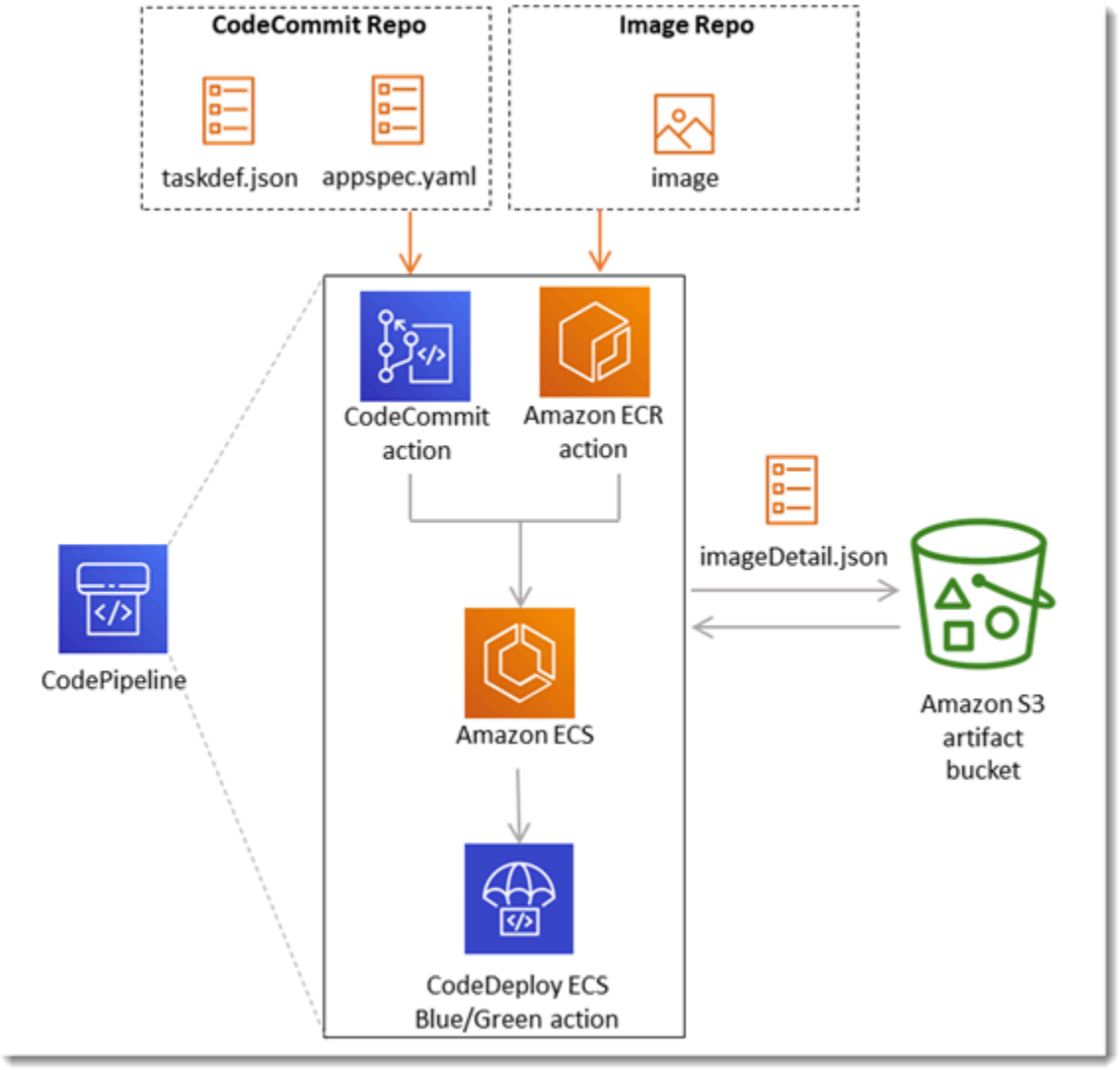
Vous devez créer le fichier `imageDetail.json` en tant que source ou artefact de génération afin qu'il soit un artefact d'entrée pour l'action de déploiement. Vous pouvez utiliser l'une de ces méthodes pour fournir le fichier `imageDetail.json` dans le pipeline :

- Incluez le `imageDetail.json` fichier dans votre emplacement source afin qu'il soit inclus dans le pipeline en tant qu'entrée pour votre action de blue/green déploiement Amazon ECS.

  > **Note**
  >
  > Si votre référentiel source est un compartiment Amazon S3, pensez à compresser le fichier JSON.

- Les actions de source Amazon ECR génèrent automatiquement un `imageDetail.json` fichier en tant qu'artefact d'entrée pour l'action suivante.

  > **Note**
  >
  > Étant donné que l'action source Amazon ECR crée ce fichier, les pipelines comportant une action source Amazon ECR n'ont pas besoin de fournir manuellement un `imageDetail.json` fichier.
  > Pour un didacticiel sur la création d'un pipeline qui inclut un stage source Amazon ECR, consultezTutoriel : Création d'un pipeline avec une source et ECS-to-CodeDeploy un déploiement Amazon ECR.



Le fichier `imageDetail.json` fournit l'URI de l'image. Il doit être construit avec la paire clé-valeur suivante.

Clé	Valeur
ImageURI	*image_URI*

imageDetail.json

Voici la structure JSON, dans laquelle l'URI de l'image est `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)

Un `imageDetail.json` fichier est généré automatiquement par l'action source Amazon ECR chaque fois qu'une modification est transmise au référentiel d'images. Les actions source `imageDetail.json` générées par Amazon ECR sont fournies sous forme d'artefact de sortie entre l'action source et l'action suivante dans le pipeline.

Voici la structure JSON, dans laquelle le nom du référentiel est `dk-image-repo`, l'URI de l'image est `ecs-repo` et la balise de l'image est `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"
    ]
}
```

Le `imageDetail.json` fichier associe l'URI de l'image et le nom du conteneur à la définition de tâche Amazon ECS comme suit :

- `ImageSizeInBytes` : taille, en octets, de l'image dans le référentiel.
- `ImageDigest` : hachage `sha256` du manifeste de l'image.
- `Version` : version de l'image.
- `ImagePushedAt` : date et heure de la transmission de la dernière image dans le référentiel.
- `RegistryId`: ID de AWS compte associé au registre qui contient le référentiel.
- `RepositoryName`: nom du référentiel Amazon ECR dans lequel l'image a été envoyée.
- `ImageURI` : URI de l'image.

- `ImageTags` : balise utilisée pour l'image.

Avant de créer votre pipeline, suivez les étapes suivantes pour configurer le fichier `imageDetail.json`.

1. Dans le cadre de la planification du blue/green déploiement d'applications basées sur des conteneurs pour votre pipeline, planifiez la phase source et la phase de génération, le cas échéant.
2. Sélectionnez l'une des méthodes suivantes :

   a. Si votre pipeline a ignoré l'étape de création, vous devez créer manuellement le fichier JSON et le télécharger dans votre référentiel source, par exemple CodeCommit, afin que l'action source puisse fournir l'artefact. Créez le fichier à l'aide de l'éditeur de texte et nommez-le ou utilisez le nom de fichier `imageDetail.json` par défaut. Procédez à la transmission du fichier `imageDetail.json` dans votre référentiel source.

   b. Si votre pipeline comporte une étape de génération, procédez comme suit :

      i. Ajoutez une commande à votre fichier buildspec qui affiche le fichier de définitions d'image dans votre référentiel source pendant la phase de génération. L'exemple suivant utilise la commande printf pour créer un fichier `imageDetail.json`. Répertoriez cette commande dans la section `post_build` du fichier buildspec.yml :

      ```
      printf '{"ImageURI":"image_URI"}' > imageDetail.json
      ```

      Vous devez inclure le fichier `imageDetail.json` en tant qu'artefact de sortie dans le fichier `buildspec.yml`.

      ii. Ajoutez le fichier `imageDetail.json` en tant qu'artefact dans le fichier `buildspec.yml`.

      ```
      artifacts:
        files:
          - imageDetail.json
      ```

Référence des variables

Cette section est fournie à des fins de référence uniquement. Pour plus d'informations sur la création de variables, consultez Utilisation des variables.

Les variables vous permettent de configurer vos actions de pipeline avec des valeurs déterminées au moment de l'exécution du pipeline ou de l'exécution de l'action.

Certains fournisseurs d'actions produisent un ensemble défini de variables. Vous choisissez parmi des clés de variable par défaut pour ce fournisseur d'action, telles que l'ID de validation.

> **Important**
>
> Lorsque vous transmettez des paramètres secrets, n'entrez pas directement la valeur. La valeur est rendue en texte brut ; elle est donc lisible. Pour des raisons de sécurité, n'utilisez pas de texte en clair avec des secrets. Nous vous recommandons vivement de l'utiliser AWS Secrets Manager pour stocker des secrets.

Pour voir des exemples étape par étape d'utilisation de variables :

- Pour un didacticiel avec une variable au niveau du pipeline transmise au moment de l'exécution du pipeline, consultez. Tutoriel : Utiliser des variables au niveau du pipeline
- Pour un didacticiel présentant une action Lambda qui utilise des variables issues d'une action en amont (CodeCommit) et génère des variables de sortie, consultezTutoriel : Utilisation de variables avec des actions d'invocation Lambda.
- Pour un didacticiel présentant une CloudFormation action qui fait référence à des variables de sortie de pile issues d'une CloudFormation action en amont, consultezTutoriel : Création d'un pipeline qui utilise des variables provenant de AWS CloudFormation actions de déploiement.
- Pour un exemple d'action d'approbation manuelle avec un texte de message faisant référence à des variables de sortie qui renvoient à l'ID de CodeCommit validation et au message de validation, consultezExemple : Utiliser des variables dans les approbations manuelles.
- Pour un exemple CodeBuild d'action avec une variable d'environnement dont la résolution est le nom de la GitHub branche, consultezExemple : utilisation d'une BranchName variable avec des variables d' CodeBuild environnement.

- CodeBuild les actions produisent en tant que variables toutes les variables d'environnement qui ont été exportées dans le cadre de la construction. Pour de plus amples informations, veuillez consulter CodeBuild variables de sortie d'action.

Limites de variable

Pour plus d'informations sur les limites, consultez Quotas dans AWS CodePipeline.

> **Note**
>
> Lorsque vous entrez une syntaxe variable dans les champs de configuration des actions, ne dépassez pas la limite de 1 000 caractères pour les champs de configuration. Une erreur de validation est renvoyée lorsque cette limite est dépassée.

Rubriques

- Concepts
- Cas d'utilisation des variables
- Configuration des variables
- Résolution des variables
- Règles pour les variables
- Variables disponibles pour les actions de pipeline

Concepts

Cette section répertorie les termes et concepts clés liés aux variables et aux espaces de noms.

Variables

Les variables sont des paires clé-valeur qui peuvent être utilisées pour configurer dynamiquement des actions dans votre pipeline. Ces variables sont actuellement mises à disposition de trois manières :

- Il existe un ensemble de variables qui sont implicitement disponibles au début de chaque exécution de pipeline. Cet ensemble comprend actuellement `PipelineExecutionId`, l'ID de l'exécution du pipeline en cours.

- Les variables au niveau du pipeline sont définies lorsque le pipeline est créé et résolues au moment de l'exécution du pipeline.

  Vous spécifiez des variables au niveau du pipeline lorsque celui-ci est créé et vous pouvez fournir des valeurs au moment de l'exécution du pipeline.

- Il existe des types d'action qui produisent des ensembles de variables lorsqu'ils sont exécutés. Vous pouvez voir les variables produites par une action en inspectant le `outputVariables` champ qui fait partie de l'ListActionExecutionsAPI. Pour obtenir la liste des noms de clé disponibles par fournisseur d'actions, consultez Variables disponibles pour les actions de pipeline. Pour connaître les variables produites par chaque type d'action, consultez le CodePipeline Référence sur la structure des actions.

Pour référencer ces variables dans votre configuration d'action, vous devez utiliser la syntaxe de référence de variable avec l'espace de noms correct.

Pour un exemple de flux de travail de variable, consultez Configuration des variables.

Espaces de noms

Pour que les variables puissent être référencées de manière unique, elles doivent être affectées à un espace de noms. Une fois qu'un ensemble de variables a été affecté à un espace de noms, elles peuvent être référencées dans une configuration d'action à l'aide de l'espace de noms et d'une clé de variable avec la syntaxe suivante :

```
#{namespace.variable_key}
```

Il existe trois types d'espaces de noms dans lesquels les variables peuvent être attribuées :

- L' espace de noms réservé codepipeline

  Il s'agit de l'espace de noms affecté à l'ensemble des variables implicites disponibles au début de chaque exécution du pipeline. Cet espace de noms est `codepipeline`. Exemple de référence de variable :

  ```
  #{codepipeline.PipelineExecutionId}
  ```

- L'espace de noms des variables au niveau du pipeline

Il s'agit de l'espace de noms attribué aux variables au niveau du pipeline. L'espace de noms pour toutes les variables au niveau du pipeline est`variables`. Exemple de référence de variable :

```
#{variables.variable_name}
```

- Espace de noms affecté à l'action

  Il s'agit d'un espace de noms que vous affectez à une action. Toutes les variables produites par l'action relèvent de cet espace de noms. Pour que les variables produites par une action puissent être utilisées dans une configuration d'action en aval, vous devez configurer l'action de production avec un espace de noms. Les espaces de noms doivent être uniques au sein de la définition du pipeline et ne peuvent pas entrer en conflit avec les noms d'artefact. Voici un exemple de référence de variable pour une action configurée avec l'espace de noms `SourceVariables`.

  ```
  #{SourceVariables.VersionId}
  ```

Cas d'utilisation des variables

Voici quelques-uns des cas d'utilisation les plus courants des variables au niveau du pipeline, afin de vous aider à déterminer comment vous pouvez utiliser les variables pour répondre à vos besoins spécifiques.

- Les variables au niveau du pipeline sont destinées CodePipeline aux clients qui souhaitent utiliser le même pipeline à chaque fois avec de légères variations dans les entrées de la configuration des actions. Tout développeur qui démarre un pipeline ajoute la valeur de la variable dans l'interface utilisateur au démarrage du pipeline. Avec cette configuration, vous ne transmettez des paramètres que pour cette exécution.
- Les variables au niveau du pipeline vous permettent de transmettre des entrées dynamiques aux actions du pipeline. Vous pouvez migrer vos pipelines paramétrés CodePipeline sans avoir à gérer différentes versions du même pipeline, ou créer des pipelines complexes.
- Vous pouvez utiliser des variables au niveau du pipeline pour transmettre des paramètres d'entrée qui vous permettent de réutiliser un pipeline à chaque exécution, par exemple lorsque vous souhaitez spécifier la version que vous souhaitez déployer dans un environnement de production, afin de ne pas avoir à dupliquer les pipelines.
- Vous pouvez utiliser un pipeline unique pour déployer des ressources vers plusieurs environnements de création et de déploiement. Par exemple, pour un pipeline doté d'un

CodeCommit référentiel, le déploiement à partir d'une branche spécifiée et d'un environnement de déploiement cible peut être effectué CodeBuild et CodeDeploy les paramètres transmis au niveau du pipeline.

> **Note**
>
> Pour Amazon ECR, Amazon S3 ou les CodeCommit sources, vous pouvez également créer un remplacement de source à l'aide d'une entrée de transformation en entrée EventBridge pour utiliser l'`revisionValue`entrée de votre événement de pipeline, où le `revisionValue` est dérivé de la variable d'événement source pour votre clé d'objet, votre commit ou votre ID d'image. Pour plus d'informations, consultez l'étape facultative pour la saisie de la transformation d'entrée incluse dans les procédures sous Actions et ressources relatives aux sources Amazon ECR EventBridgeConnexion aux actions de source Amazon S3 avec une source activée pour les événements, ouCodeCommit actions à la source et EventBridge.

Configuration des variables

Vous pouvez configurer des variables au niveau du pipeline ou au niveau de l'action dans la structure du pipeline.

Configuration des variables au niveau du pipeline

Vous pouvez ajouter une ou plusieurs variables au niveau du pipeline. Vous pouvez référencer cette valeur dans la configuration des CodePipeline actions. Vous pouvez ajouter les noms des variables, les valeurs par défaut et les descriptions lorsque vous créez le pipeline. Les variables sont résolues au moment de l'exécution.

> **Note**
>
> Si aucune valeur par défaut n'est définie pour une variable au niveau du pipeline, la variable est considérée comme obligatoire. Vous devez spécifier des remplacements pour toutes les variables requises lorsque vous démarrez un pipeline, sinon l'exécution du pipeline échouera avec une erreur de validation.

Vous fournissez des variables au niveau du pipeline à l'aide de l'attribut variables dans la structure du pipeline. Dans l'exemple suivant, la variable `Variable1` a une valeur de`Value1`.

```
"variables": [
     {
         "name": "Variable1",
         "defaultValue": "Value1",
         "description": "description"
     }
 ]
```

Pour un exemple de structure JSON du pipeline, consultezCréation d'un pipeline, d'étapes et d'actions.

Pour un didacticiel avec une variable au niveau du pipeline transmise au moment de l'exécution du pipeline, consultez. Tutoriel : Utiliser des variables au niveau du pipeline

Notez que l'utilisation de variables au niveau du pipeline dans tout type d'action Source n'est pas prise en charge.

> **Note**
>
> Si l'`variables`espace de noms est déjà utilisé dans certaines actions du pipeline, vous devez mettre à jour la définition de l'action et choisir un autre espace de noms pour l'action en conflit.

Configuration des variables au niveau de l'action

Vous configurez une action pour produire des variables en déclarant un espace de noms pour l'action. L'action doit déjà être un des fournisseurs d'actions qui génère des variables. Sinon, les variables disponibles sont des variables de niveau pipeline.

Vous pouvez déclarer l'espace de noms comme suit :

- Sur la page Modifier l'action de la console, entrez un espace de noms dans Espace de noms variable.
- En entrant un espace de noms dans le champ du paramètre `namespace` de la structure du pipeline JSON.

Dans cet exemple, vous ajoutez le `namespace` paramètre à l'action CodeCommit source avec le nom`SourceVariables`. Cela configure l'action pour produire les variables disponibles pour ce fournisseur d'actions, telles que `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
        }
    ]
},
```

Ensuite, vous configurez l'action en aval pour utiliser les variables produites par l'action précédente. Pour ce faire, procédez comme suit :

- Dans la page Edit action (Modifier l'action) de la console, saisissez la syntaxe de variable (pour l'action en aval) dans les champs de configuration de l'action.
- Saisie de la syntaxe de variable (pour l'action en aval) dans les champs de configuration de l'action dans la structure du pipeline JSON

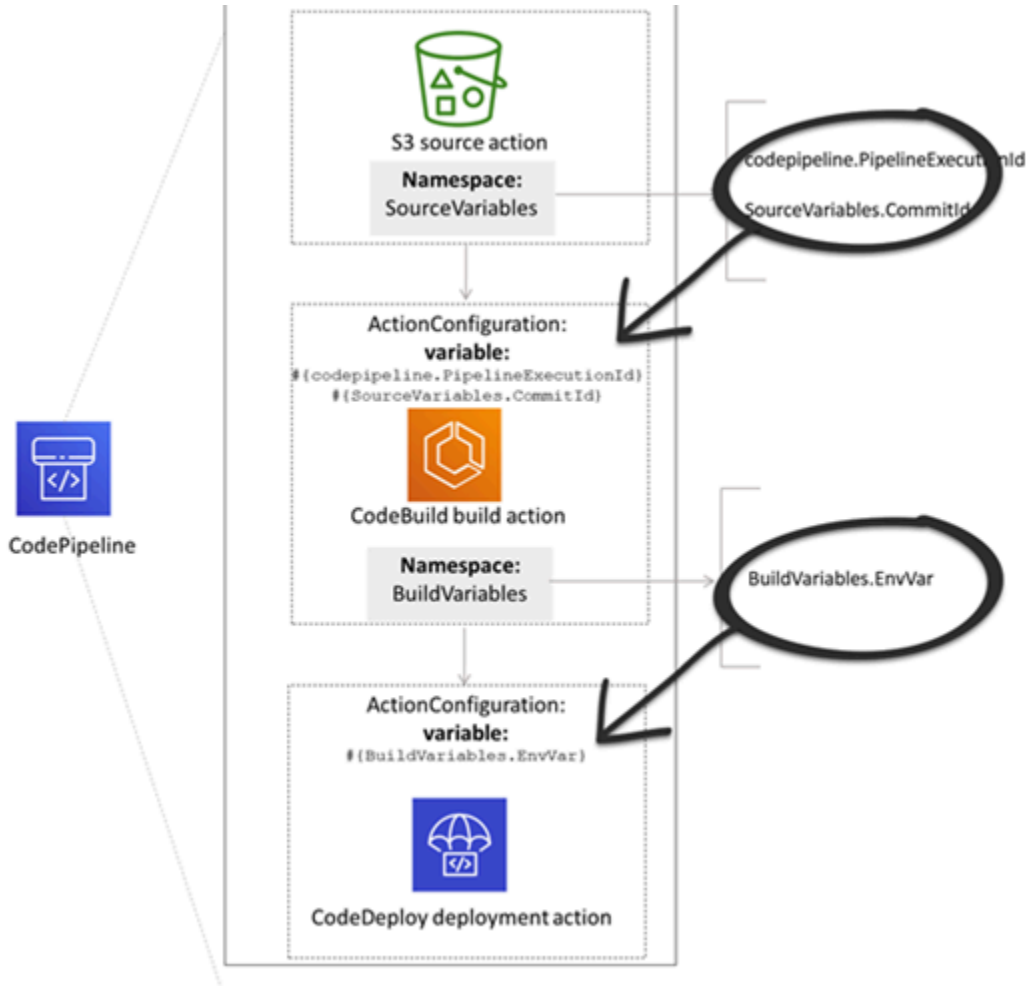
Dans cet exemple, le champ de configuration de l'action de build affiche les variables d'environnement mises à jour lors de l'exécution de l'action. L'exemple spécifie l'espace de noms et la variable pour l'ID d'exécution avec `#{codepipeline.PipelineExecutionId}` et l'espace de noms et la variable pour l'ID de validation avec **`#{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
        }
    ]
},
```

Résolution des variables

Chaque fois qu'une action est exécutée dans le cadre d'une exécution de pipeline, les variables produites peuvent être utilisées dans toute action dont l'exécution est garantie après l'action de

production. Pour utiliser ces variables dans une action de consommation, vous pouvez les ajouter à la configuration de l'action de consommation en utilisant la syntaxe indiquée dans l'exemple précédent. Avant d'effectuer une action consommatrice, CodePipeline résout toutes les références de variables présentes dans la configuration avant de lancer l'exécution de l'action.



Règles pour les variables

Les règles suivantes vous aident à configurer les variables :

- Vous spécifiez l'espace de noms et la variable pour une action via une nouvelle propriété d'action ou en modifiant une action.
- Lorsque vous utilisez l'assistant de création de pipeline, la console génère un espace de noms pour chaque action créée avec l'assistant.
- Si l'espace de noms n'est pas spécifié, les variables produites par cette action ne peuvent être référencées dans aucune configuration d'action.

- Pour référencer des variables produites par une action, l'action de référencement doit se produire après l'action qui produit les variables. Cela signifie qu'elle doit se produire soit au cours d'une étape ultérieure à l'action produisant les variables, soit au cours de la même étape mais dans un ordre d'exécution supérieur.

Variables disponibles pour les actions de pipeline

Le fournisseur d'actions détermine les variables qui peuvent être générées par l'action.

Pour des procédures pas à pas de gestion des variables, consultez Utilisation des variables.

Actions avec touches variables définies

Contrairement à un espace de noms que vous pouvez choisir, les actions suivantes utilisent des clés variables qui ne peuvent pas être modifiées. Par exemple, pour le fournisseur d'actions Amazon S3, seules les clés `VersionId` variables `ETag` et sont disponibles.

Chaque exécution possède également un ensemble de variables de CodePipeline-generated pipeline contenant des données relatives à l'exécution, telles que l'ID de version du pipeline. Ces variables peuvent être utilisées par n'importe quelle action du pipeline.

Rubriques

- CodePipeline variable d'ID d'exécution
- Variables de sortie des actions Amazon ECR
- CloudFormation StackSets variables de sortie d'action
- CodeCommit variables de sortie d'action
- CodeStarSourceConnection variables de sortie d'action
- GitHub action variables de sortie (GitHub (via l'application OAuth) action)
- Variables de sortie de l'action S3

CodePipeline variable d'ID d'exécution

CodePipeline variable d'ID d'exécution

Fournisseur	Clé de variable	Exemple de valeur	Exemple de syntaxe de variable
codepipeline	`PipelineExecutionId`	8abc75f0-fbf8-4f4c-bfEXAMPLE	`#{codepipeline.PipelineExecutionId}`

Variables de sortie des actions Amazon ECR

Variables Amazon ECR

Clé de variable	Exemple de valeur	Exemple de syntaxe de variable
`ImageDigest`	sha256:EXAMPLE1122334455	`#{SourceVariables.ImageDigest}`
`ImageTag`	dernières	`#{SourceVariables.ImageTag}`
`ImageURI`	11111 EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest	`#{SourceVariables.ImageURI}`
`RegistryId`	EXAMPLE12233	`#{SourceVariables.RegistryId}`
`RepositoryName`	my-image-repo	`#{SourceVariables.RepositoryName}`

CloudFormation StackSets variables de sortie d'action

CloudFormation StackSets variables

Clé de variable	Exemple de valeur	Exemple de syntaxe de variable
`OperationId`	11111111-2bbb-111-2bbb-11111 exemple	`#{DeployVariables.OperationId}`
`StackSetId`	exemple my-stackset:1111aaaa-1111-2222-2bbb-11111	`#{DeployVariables.StackSetId}`

CodeCommit variables de sortie d'action

CodeCommit variables

Clé de variable	Exemple de valeur	Exemple de syntaxe de variable
`AuthorDate`	29/10/2019 T03:32:21Z	`#{SourceVariables.AuthorDate}`
`BranchName`	développement	`#{SourceVariables.BranchName}`
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Correction d'un bug (taille maximale de 100 Ko)	`#{SourceVariables.CommitMessage}`
`CommitterDate`	29/10/2019 T03:32:21Z	`#{SourceVariables.CommitterDate}`
`RepositoryName`	mon CodeCommitRepo	`#{SourceVariables.RepositoryName}`

CodeStarSourceConnection variables de sortie d'action

Les variables de sortie disponibles dépendent du type de déclencheur (push, pull request ou tag). Les variables du tableau suivant marquées comme Toutes sont produites pour chaque type de déclencheur. Les variables marquées d'un type de déclencheur spécifique ne sont produites que lorsque l'exécution du pipeline est lancée par ce déclencheur.

`CodeStarSourceConnection`variables (Bitbucket Cloud GitHub, GitHub Enterprise Repository et GitLab.com)

Clé de variable	Exemple de valeur	Exemple de syntaxe de variable	Type de déclencheur
`AuthorDate`	29/10/2019 T03:32:21Z	`#{SourceVariables.AuthorDate}`	Tous
`AuthorDisplayName`	Jane Doe	`#{SourceVariables.AuthorDisplayName}`	Tous
`AuthorEmail`	jane.doe@example.com	`#{SourceVariables.AuthorEmail}`	Tous
`AuthorId`	janedoe	`#{SourceVariables.AuthorId}`	Tous
`BranchName`	développement	`#{SourceVariables.BranchName}`	Appuyez, étiquetez
`CommitId`	exampleb01f91b31	`#{SourceVariables.CommitId}`	Tous

Clé de variable	Exemple de valeur	Exemple de syntaxe de variable	Type de déclencheur
`CommitMessage`	Correction d'un bug (taille maximale de 100 Ko)	`#{SourceVariables.CommitMessage}`	Tous
`ConnectionArn`	arn:aws:codestar-connections:region ::connection/ *account-id connection-id*	`#{SourceVariables.ConnectionArn}`	Tous
`DestinationBranchName`	principal	`#{SourceVariables.DestinationBranchName}`	Demande d'extraction
`FullRepositoryName`	username/GitHubRepo	`#{SourceVariables.FullRepositoryName}`	Tous
`ProviderType`	GitHub	`#{SourceVariables.ProviderType}`	Tous
`PullRequestId`	1	`#{SourceVariables.PullRequestId}`	Demande d'extraction
`PullRequestTitle`	Ajouter une nouvelle fonctionnalité	`#{SourceVariables.PullRequestTitle}`	Demande d'extraction

Clé de variable	Exemple de valeur	Exemple de syntaxe de variable	Type de déclencheur
SourceBranchName	branche de fonctionnalités	#{SourceVariables.SourceBranchName}	Demande d'extraction
TagName	v1.0.0	#{SourceVariables.TagName}	Balise

GitHub action variables de sortie (GitHub (via l'application OAuth) action)

GitHub action sur les variables GitHub (via l'application OAuth))

Clé de variable	Exemple de valeur	Exemple de syntaxe de variable
AuthorDate	29/10/2019 T03:32:21Z	#{SourceVariables.AuthorDate}
BranchName	principal	#{SourceVariables.BranchName}
CommitId	exampleb01f91b31	#{SourceVariables.CommitId}
CommitMessage	Correction d'un bug (taille maximale de 100 Ko)	#{SourceVariables.CommitMessage}
CommitterDate	29/10/2019 T03:32:21Z	#{SourceVariables.CommitterDate}
CommitUrl		#{SourceVariables.CommitUrl}

Clé de variable	Exemple de valeur	Exemple de syntaxe de variable
`RepositoryName`	mon GitHubRepo	`#{SourceVariables.RepositoryName}`

Variables de sortie de l'action S3

Variables S3

Clé de variable	Exemple de valeur	Exemple de syntaxe de variable
`ETag`	example28be1c3	`#{SourceVariables.ETag}`
`VersionId`	exampleta_IUQCv	`#{SourceVariables.VersionId}`

Actions à l'aide de touches variables configurées par l'utilisateur

Pour les actions CodeBuild CloudFormation,, et Lambda, les clés variables sont configurées par l'utilisateur.

Rubriques

- CloudFormation variables de sortie d'action
- CodeBuild variables de sortie d'action
- Variables de sortie d'action Lambda

CloudFormation variables de sortie d'action

CloudFormation variables

Clé de variable	Exemple de syntaxe de variable
Pour les CloudFormation actions, les variables sont produites à partir de toutes les valeurs désignées dans la `Outputs`	`#{DeployVariables.StackName}`

Clé de variable	Exemple de syntaxe de variable
section d'un modèle de pile. Notez que les seuls modes CloudFormation d'action qui génèrent des sorties sont ceux qui entraînent la création ou la mise à jour d'une pile, tels que la création de pile, les mises à jour de pile et l'exécution d'un ensemble de modifications. Les modes d'action correspon dants qui génèrent des variables sont les suivants : • CREATE_UPDATE • CHANGE_SET_EXECUTE • CHANGE_SET_REPLACE • REPLACE_ON_FAILURE Pour plus d'informations sur ces modes d'action, consultez CloudFormation référence d'action de déploiement. Pour un didacticiel qui explique comment créer un pipeline avec une action de CloudFormation déploiement dans un pipeline qui utilise des variables CloudFormation de sortie, consultez Tutoriel : Création d'un pipeline qui utilise des variables provenant de AWS CloudFormation actions de déploiement.	

CodeBuild variables de sortie d'action

CodeBuild variables

Clé de variable	Exemple de syntaxe de variable
Pour les CodeBuild actions, les variables sont produites à partir des valeurs générées par les variables d'environnement exportées. Configurez une variable d' CodeBuild environne ment en modifiant votre CodeBuild action dans CodePipeline ou en ajoutant la variable d'environnement à la spécification de construction. Ajoutez des instructions à votre CodeBuild buildspec pour ajouter la variable d'environnement dans la section des	`#{BuildVariables.EnvVar}`

Clé de variable	Exemple de syntaxe de variable
variables exportées. Voir env/exported -variables dans le Guide de l'AWS CodeBuild utilisateur.	

Variables de sortie d'action Lambda

Variables lambda

Clé de variable	Exemple de syntaxe de variable
L'action Lambda produira en tant que variables toutes les paires clé-valeur incluses dans la `outputVariables` section de la PutJobSuccessResult demande d'API. Pour un didacticiel présentant une action Lambda qui utilise des variables issues d'une action en amont (CodeCommit) et génère des variables de sortie, consultezTutoriel : Utilisation de variables avec des actions d'invocation Lambda.	`#{TestVariables.testRunId}`

Utilisation de modèles globulaires dans la syntaxe

Lorsque vous spécifiez les fichiers ou les chemins utilisés dans les artefacts du pipeline ou les emplacements des sources, vous pouvez spécifier l'artefact en fonction du type d'action. Par exemple, pour l'action S3, vous spécifiez la clé de l'objet S3.

Pour les déclencheurs, vous pouvez définir des filtres. Vous pouvez utiliser des modèles globulaires pour définir des filtres. Voici quelques exemples.

Lorsque la syntaxe est « globale », la représentation sous forme de chaîne du chemin est mise en correspondance à l'aide d'un langage de modèles limité avec une syntaxe qui ressemble à des expressions régulières. Par exemple :

- `*.java`Spécifie un chemin qui représente un nom de fichier se terminant par .java
- `*.*`Spécifie les noms de fichiers contenant un point
- `*.{java,class}`Spécifie les noms de fichiers se terminant par .java ou .class
- `foo.?`Spécifie les noms de fichiers commençant par foo. et une extension à un seul caractère

Les règles suivantes sont utilisées pour interpréter les modèles globulaires :

- Pour spécifier zéro ou plusieurs caractères d'un composant de nom dans les limites du répertoire, utilisez`*`.
- Pour spécifier zéro ou plusieurs caractères d'un composant de nom dépassant les limites du répertoire, utilisez`**`.
- Pour spécifier un caractère d'un composant de nom, utilisez`?`.
- Pour éviter les caractères qui seraient autrement interprétés comme des caractères spéciaux, utilisez la barre oblique inverse ()`\`.
- Pour spécifier un seul caractère parmi un ensemble de caractères, utilisez`[ ]`.
- Pour spécifier un seul fichier situé à la racine de l'emplacement de votre build ou de votre référentiel source, utilisez`my-file.jar`.
- Pour spécifier un seul fichier dans un sous-répertoire, utilisez `directory/my-file.jar` ou`directory/subdirectory/my-file.jar`.
- Pour spécifier tous les fichiers, utilisez`"**"`. Le modèle `**` glob indique qu'il doit correspondre à un nombre quelconque de sous-répertoires.

- Pour spécifier tous les fichiers et répertoires d'un répertoire nommé`directory`, utilisez`"directory/**"`. Le modèle `**` glob indique qu'il doit correspondre à un nombre quelconque de sous-répertoires.
- Pour spécifier tous les fichiers d'un répertoire nommé`directory`, mais aucun de ses sous-répertoires, utilisez`"directory/*"`.
- Dans une expression entre crochets`*`, les `\` caractères `?` et correspondent à eux-mêmes. Le caractère (-) correspond à lui-même s'il `!` s'agit du premier caractère entre crochets ou du premier caractère après la négation.
- Les `{ }` caractères sont un groupe de sous-modèles, où le groupe correspond si l'un des sous-modèles du groupe correspond. Le `","` caractère est utilisé pour séparer les sous-modèles. Les groupes ne peuvent pas être imbriqués.

Mise à jour de pipelines d'interrogation vers la méthode recommandée de détection des modifications

Si vous disposez d'un pipeline qui utilise le sondage pour réagir aux modifications apportées à la source, vous pouvez le mettre à jour pour utiliser la méthode de détection recommandée. Pour obtenir un guide de migration contenant des instructions pour mettre à jour vos pipelines de sondage afin d'utiliser la méthode recommandée de détection des modifications basée sur les événements, consultez. Migrez les pipelines de sondage pour utiliser la détection des modifications basée sur les événements

Mettre à jour une action source GitHub (via l'application OAuth) en une action source GitHub (via GitHub l'application)

Dans AWS CodePipeline, il existe deux versions prises en charge de l'action GitHub source :

- Recommandé : L'action GitHub (via GitHub l'application) utilise une authentification GitHub basée sur l'application et soutenue par une CodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées ressource. Il installe une application AWS CodeStar Connections dans votre GitHub organisation afin que vous puissiez gérer l'accès dans GitHub.
- Déconseillé : l'action GitHub (via l'application OAuth) utilise des jetons OAuth pour s'authentifier GitHub et utilise un webhook distinct pour détecter les modifications. Cette méthode n'est plus recommandée.

> **Note**
>
> Les connexions ne sont pas disponibles dans les régions Asie-Pacifique (Hong Kong), Asie-Pacifique (Hyderabad), Asie-Pacifique (Jakarta), Asie-Pacifique (Melbourne), Asie-Pacifique (Osaka), Afrique (Le Cap), Moyen-Orient (Bahreïn), Moyen-Orient (Émirats arabes unis), Europe (Espagne), Europe (Zurich), Israël (Tel Aviv) ou AWS GovCloud (US-West). Pour consulter les autres actions disponibles, consultezIntégrations de produits et de services avec CodePipeline. Pour les considérations relatives à cette action dans la région Europe (Milan), voir la note figurant dansCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.

L'utilisation de l'action GitHub (via l' GitHub application) au lieu de l'action GitHub (via l'application OAuth) présente certains avantages importants :

- Avec les connexions, vous CodePipeline n'avez plus besoin d'applications OAuth ni de jetons d'accès personnels pour accéder à votre référentiel. Lorsque vous créez une connexion, vous installez une GitHub application qui gère l'authentification auprès de votre GitHub référentiel et accorde des autorisations au niveau de l'organisation. Vous devez autoriser les jetons OAuth en tant qu'utilisateur pour accéder au référentiel. Pour plus d'informations sur OAuth-based

GitHub l'accès par opposition à App-based GitHub l'accès, consultezhttps://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps.

- Lorsque vous gérez GitHub (via GitHub App) des actions dans l'interface de ligne de commande ou CloudFormation que vous n'avez plus besoin de stocker votre jeton d'accès personnel en tant que secret dans Secrets Manager. Il n'est plus nécessaire de référencer dynamiquement le secret stocké dans la configuration de vos CodePipeline actions. Vous ajoutez plutôt l'ARN de connexion à votre configuration d'action. Pour un exemple de configuration d'action, consultezCodeStarSourceConnection pour Bitbucket Cloud GitHub, GitHub Enterprise Server et GitLab.com les actions GitLab autogérées.
- Lorsque vous créez une ressource de connexion à utiliser avec votre action GitHub (via GitHub App) dans CodePipeline, vous pouvez utiliser la même ressource de connexion pour associer d'autres services pris en charge, tels que CodeGuru Reviewer, à votre référentiel.
- Dans GitHub (via GitHub App), vous pouvez cloner des référentiels pour accéder aux métadonnées git lors d' CodeBuild actions ultérieures, tandis que dans GitHub (via l'application OAuth), vous ne pouvez télécharger que la source.
- Un administrateur installe l'application pour les référentiels de votre organisation. Vous n'avez plus besoin de suivre les jetons OAuth qui dépendent de la personne qui les a créés.

Toutes les applications installées dans une organisation ont accès au même ensemble de référentiels. Pour modifier les personnes autorisées à accéder à chaque référentiel, modifiez la politique IAM pour chaque connexion. Pour un exemple, voir Exemple : une politique limitée pour l'utilisation de connexions avec un référentiel spécifié.

Vous pouvez suivre les étapes décrites dans cette rubrique pour supprimer votre action source GitHub (via l'application OAuth) et ajouter une action source GitHub (via GitHub l'application) depuis la CodePipeline console.

Rubriques

- Étape 1 : remplacez votre action (via l'application OAuth) GitHub
- Étape 2 : créer une connexion avec GitHub
- Étape 3 : Enregistrez votre action GitHub source

Étape 1 : remplacez votre action (via l'application OAuth) GitHub

Utilisez la page de modification du pipeline pour remplacer votre action (via l'application OAuth) par une GitHub action GitHub (via GitHub l'application).

Pour remplacer votre action (via l'application OAuth) GitHub

1. Connectez-vous à la CodePipeline console.
2. Choisissez votre pipeline, puis sélectionnez Modifier. Choisissez Modifier l'étape sur votre scène source. Un message s'affiche pour vous recommander de mettre à jour votre action.
3. Dans le fournisseur d'actions, choisissez GitHub (via GitHub l'application).
4. Effectuez l'une des actions suivantes :

   - Sous Connexion, si vous n'avez pas encore créé de connexion avec votre fournisseur, choisissez Se connecter à GitHub. Passez à l'étape 2 : créez une connexion avec GitHub.
   - Sous Connexion, si vous avez déjà créé une connexion avec votre fournisseur, choisissez-la. Passez à l'étape 3 : enregistrez l'action source pour votre connexion.

Étape 2 : créer une connexion avec GitHub

Une fois que vous avez choisi de créer la connexion, la GitHub page Se connecter à s'affiche.

Pour créer une connexion avec GitHub

1. Dans les paramètres de GitHub connexion, le nom de votre connexion apparaît dans Nom de la connexion.

   Sous GitHub Applications, choisissez l'installation d'une application ou choisissez Installer une nouvelle application pour en créer une.

   > **Note**
   >
   > Installez une application pour toutes vos connexions à un fournisseur particulier. Si vous avez déjà installé l' GitHub application, choisissez-la et ignorez cette étape.

2. Si la page d'autorisation GitHub s'affiche, connectez-vous avec vos informations d'identification, puis choisissez de continuer.

3. Sur la page d'installation de l'application, un message indique que l' AWS CodeStar application essaie de se connecter à votre GitHub compte.

   > **Note**
   >
   > Vous n'installez l'application qu'une seule fois pour chaque GitHub compte. Si vous avez déjà installé l'application, vous pouvez choisir Configure (Configurer) pour passer à une page de modification pour l'installation de votre application, ou vous pouvez utiliser le bouton Précédent pour revenir à la console.

4. Sur la AWS CodeStar page Installer, choisissez Installer.
5. Sur la GitHub page Se connecter à, l'ID de connexion de votre nouvelle installation s'affiche. Choisissez Se connecter.

Étape 3 : Enregistrez votre action GitHub source

Effectuez vos mises à jour sur la page Modifier l'action pour enregistrer votre nouvelle action source.

Pour enregistrer votre action GitHub source

1. Dans Référentiel, entrez le nom de votre référentiel tiers. Dans Branche, entrez la branche dans laquelle vous souhaitez que votre pipeline détecte les changements de source.

   > **Note**
   >
   > Dans Repository, tapez `owner-name/repository-name` comme indiqué dans cet exemple :
   >
   > ```
   > my-account/my-repository
   > ```

2. Dans Format d'artefact de sortie, choisissez le format de vos artefacts.

   - Pour stocker les artefacts de sortie de l' GitHub action à l'aide de la méthode par défaut, choisissez CodePipeline default. L'action accède aux fichiers depuis le GitHub référentiel et stocke les artefacts dans un fichier ZIP dans le magasin d'artefacts du pipeline.

- Pour stocker un fichier JSON contenant une référence d'URL au référentiel afin que les actions en aval puissent exécuter directement les commandes Git, choisissez Full clone (Clone complet). Cette option ne peut être utilisée que par les actions CodeBuild en aval.

  Si vous choisissez cette option, vous devrez mettre à jour les autorisations pour votre rôle de service de CodeBuild projet, comme indiqué dansAjoutez CodeBuild GitClone des autorisations pour les connexions à Bitbucket GitHub, GitHub Enterprise Server ou GitLab.com. Pour un didacticiel qui explique comment utiliser l'option de clonage complet, consultezTutoriel : Utiliser un clone complet avec une source de GitHub pipeline.

3. Dans Artefacts de sortie, vous pouvez conserver le nom de l'artefact de sortie pour cette action, par exemple`SourceArtifact`. Choisissez Terminé pour fermer la page Modifier l'action.
4. Choisissez Terminé pour fermer la page d'édition de la scène. Choisissez Enregistrer pour fermer la page de modification du pipeline.

Quotas dans AWS CodePipeline

CodePipeline dispose de quotas pour le nombre de pipelines, d'étapes, d'actions et de webhooks qu'un AWS compte peut avoir dans chaque AWS région.

Ces quotas s'appliquent par région et peuvent être augmentés. Les demandes d'augmentation de quota sont traitées dans un délai de deux semaines maximum.

Ressource	Par défaut
Durée avant l'expiration d'une action (Il s'agit de délais d'expiration configurables. (Consultez le tableau suivant pour les délais d'expiration non configurables)	CloudFormation action de déploiement : 3 jours CodeDeploy actions de déploiement et CodeDeploy ECS (blue/green) : 5 jours AWS Lambda action d'invocation : 24 heures **Note** Pendant l'exécution de l'action, contacte CodePipeline régulière ment Lambda pour obtenir un état. La fonction Lambda répond avec un statut indiquant que l'exécution de l'action est réussie, échouée ou en cours. Si la fonction Lambda n'a envoyé aucune réponse au bout de 20 minutes, l'action expire. Si, pendant les 20 minutes, la fonction Lambda a répondu que l'action était toujours en cours, CodePipeline redémarre le chronomètre de 20 minutes et réessaie. En cas d'échec au bout de 24 heures, CodePipel ine définit l'état de l'action d'appel Lambda sur Echec.

Ressource	Par défaut
	Lambda dispose d'un délai d'expiration distinct pour les fonctions Lambda qui n'est pas lié au délai d'expiration de l'action. CodePipeline Action de déploiement d'Amazon S3 : 90 minutes **Note** Si le chargement vers S3 expire pendant le déploiement d'un fichier ZIP volumineux, l'action échoue avec une erreur de temporisation. Essayez de diviser le fichier ZIP en fichiers plus petits. Action d'approbation manuelle, délai d'expiration par défaut du compte : 7 jours **Note** Le délai d'expiration par défaut pour l'action d'approbation manuelle peut être remplacé pour une action spécifique dans le pipeline, et il est configurable jusqu'à 86 400 minutes (60 jours) avec une valeur minimale de 5 minutes. Pour plus d'informations, consultez ActionDeclaration dans la Référence d'API CodePipeline .

Ressource	Par défaut
	Une fois configuré, ce délai est appliqué à l'action. Dans le cas contraire, c'est le niveau de compte par défaut qui est utilisé. Toutes les autres actions : 1 heure **Note** Le délai d'expiration de l'action de déploiement Amazon ECS est configurable jusqu'à une heure (délai d'expiration par défaut).
Nombre maximum de pipelines par région dans un AWS compte	1 000 **Note** Les pipelines configurés pour l'interrogation ou la détection des modifications basées sur des événements sont comptabilisés dans ce quota.

Ressource	Par défaut
Nombre maximum de pipelines configurés pour interroger les changements de source, par AWS région	300 **Note** Ce quota est fixe et ne peut pas être modifié. Si vous atteignez la limite pour les pipelines d'interrogation, vous pouvez toujours configurer des pipelines supplémentaires qui utilisent la détection des modificat ions basée sur les événement s. Pour de plus amples informati ons, veuillez consulter Actions à la source et méthodes de détection des modifications.[1]
Nombre maximum de webhooks par région dans un compte AWS	300
Nombre d'actions personnalisées par région dans un AWS compte	50

[1]En fonction de votre fournisseur source, utilisez les instructions suivantes pour mettre à jour vos pipelines d'interrogation afin d'utiliser la détection des modifications basée sur les événements :

- Pour mettre à jour une action CodeCommit source, voirMigrer les pipelines de sondage (CodeCommit ou la source Amazon S3) (console).
- Pour mettre à jour une action source Amazon S3, consultezMigrer les pipelines de sondage (CodeCommit ou la source Amazon S3) (console).
- Pour mettre à jour une action GitHub source, voirMigrer les pipelines de sondage vers les webhooks GitHub (via l'application OAuth) (actions source) (console).

Les quotas suivants AWS CodePipeline s'appliquent à la disponibilité des régions, aux contraintes de dénomination et aux tailles d'artefacts autorisées. Ces quotas sont fixes et ne peuvent pas être modifiés.

Pour obtenir la liste des points de terminaison de CodePipeline service pour chaque région, voir AWS CodePipeline Points de terminaison et quotas dans la référence AWS générale.

Pour plus d'informations sur les exigences de structure, consultez CodePipeline référence de la structure du pipeline.

AWS Régions dans lesquelles vous pouvez créer un pipeline	USA Est (Virginie du Nord) USA Est (Ohio) USA Ouest (Californie du Nord) US West (Oregon) Canada (Centre) Europe (Francfort) Europe (Zurich) * Europe (Stockholm) Europe (Milan) * Europe (Espagne) * Europe (Irlande) Europe (Londres) Europe (Paris) Israël (Tel Aviv) * Afrique (Le Cap) * Asie-Pacifique (Hong Kong) * Asie-Pacifique (Mumbai)

	Asie-Pacifique (Hyderabad) *
	Asie-Pacifique (Tokyo)
	Asie-Pacifique (Séoul)
	Asie-Pacifique (Osaka)
	Asie-Pacifique (Singapour)
	Asie-Pacifique (Sydney)
	Asie-Pacifique (Jakarta) *
	Asie-Pacifique (Melbourne) *
	Asie-Pacifique (Malaisie) *
	Asie-Pacifique (Nouvelle Zélande) *
	Amérique du Sud (São Paulo)
	Moyen-Orient (Bahreïn) *
	Moyen-Orient (Émirats arabes unis) *
	AWS GovCloud (US-West)
	AWS GovCloud (US-East)

Caractères autorisés dans un nom d'action	Les noms d'action ne peuvent pas dépasser 100 caractères. Caractères autorisés : Lettres minuscules a à z, bornes incluses. Lettres majuscules A à Z, bornes incluses. Chiffres 0 à 9 inclus. Caractères spéciaux . (point), @ (signe arobase), - (signe moins) et _ (trait de soulignement). Tous les autres caractères, tels que des espaces, ne sont pas autorisés.
Caractères autorisés dans les types d'action	Les noms de types d'action ne peuvent pas dépasser 25 caractères. Caractères autorisés : Lettres minuscules a à z, bornes incluses. Lettres majuscules A à Z, bornes incluses. Chiffres 0 à 9, bornes incluses. Caractères spéciaux . (point), @ (signe arobase), - (signe moins) et _ (trait de soulignement). Tous les autres caractères, tels que des espaces, ne sont pas autorisés.

Caractères autorisés dans les noms d'artefacts	Les noms des artifacts ne peuvent pas dépasser 100 caractères. Caractères autorisés : Lettres minuscules a à z, bornes incluses. Lettres majuscules A à Z, bornes incluses. Chiffres 0 à 9 inclus. Caractères spéciaux - (signe moins) et _ (trait de soulignement). Tous les autres caractères, tels que des espaces, ne sont pas autorisés.
Caractères autorisés dans les noms d'actions partenaires	Les noms des actions partenaires doivent respecter les mêmes conventions de dénomination et les mêmes restrictions que les autres noms d'action dans l' CodePipeline. En particulier, ils ne peuvent pas dépasser 100 caractères. Caractères autorisés : Lettres minuscules a à z, bornes incluses. Lettres majuscules A à Z, bornes incluses. Chiffres 0 à 9, bornes incluses. Caractères spéciaux . (point), @ (signe arobase), - (signe moins) et _ (trait de soulignement). Tous les autres caractères, tels que des espaces, ne sont pas autorisés.

Caractères autorisés dans un nom de pipeline	Les noms de pipeline ne peuvent pas excéder 100 caractères. Caractères autorisés : Lettres minuscules a à z, bornes incluses. Lettres majuscules A à Z, bornes incluses. Chiffres 0 à 9, bornes incluses. Caractères spéciaux . (point), @ (signe arobase), - (signe moins) et _ (trait de soulignement). Tous les autres caractères, tels que des espaces, ne sont pas autorisés.
Caractères autorisés dans un nom d'étape	Les noms des étapes ne peuvent pas excéder 100 caractères. Caractères autorisés : Lettres minuscules a à z, bornes incluses. Lettres majuscules A à Z, bornes incluses. Chiffres 0 à 9, bornes incluses. Caractères spéciaux . (point), @ (signe arobase), - (signe moins) et _ (trait de soulignement). Tous les autres caractères, tels que des espaces, ne sont pas autorisés.

Durée avant l'expiration d'une action	CodeBuild action de construction : 36 heures Action de test : 8 heures Actions personnalisées : 24 heures Step Functions invoque une action : 7 jours Délai de génération pour l'action Commandes : 55 minutes
Longueur maximale de la clé de configuration d'action (par exemple, les clés de configuration CodeBuild sont `ProjectName` , `PrimarySource` et `EnvironmentVariables`)	50 caractères
Longueur maximale de la valeur de configuration d'action (par exemple, la valeur de la `RepositoryName` configuration dans la configuration CodeCommit d'action doit être inférieure à 1 000 caractères) : `"RepositoryName": "my-repo-name-less-than-1000-characters"`)	1 000 caractères
Nombre maximal d'actions par pipeline	1 000
Nombre maximum d'exécutions de pipeline simultanées par pipeline (mode QUEUED|PARALLEL)	50
Nombre maximal d'exécutions d'actions simultanées par exécution de pipeline en mode PARALLÈLE	5
Nombre maximum de fichiers pour un objet Amazon S3	100 000
Nombre maximum d'actions parallèles dans une étape	100

Nombre maximum d'actions séquentielles dans une étape	100
Taille maximum des artéfacts dans une étape source	Artefacts stockés dans des compartiments Amazon S3 : 7 Go Artefacts stockés dans CodeCommit nos GitHub référentiels : 1 Go Exception : si vous utilisez AWS Elastic Beanstalk pour déployer des applications, la taille maximale de l'artefact est toujours de 512 Mo. Exception : si vous utilisez CloudFormation pour déployer des applications, la taille maximale de l'artefact est toujours de 256 Mo. Exception : si vous utilisez l'action `CodeDeployToECS` pour déployer des applications, la taille maximale de l'artéfact est toujours de 3 Mo.
Taille maximale du fichier JSON de définitions d'images utilisé dans les pipelines déployant des conteneurs et des images Amazon ECS	100 Ko
Taille maximale des artefacts d'entrée pour les CloudFormation actions	256 Mo
Taille maximale des artefacts d'entrée pour l'action `CodeDeployToECS`	3 Mo
Taille maximale des artefacts d'entrée pour l'action `Step Functions`	L'action Step Functions s'exécute sur Lambda, et ses quotas de taille d'artefact sont donc les mêmes que ceux des fonctions Lambda. Pour plus d'informations, consultez la section Quotas Lambda dans le guide du développeur Lambda.

Taille maximum de l'objet JSON qui peut être stocké dans la propriété `ParameterOverrides`	Pour une action de CodePipeline déploiement avec CloudFormation comme fournisseur, la `ParameterOverrides` propriété est utilisée pour stocker un objet JSON qui spécifie les valeurs du fichier de configuration du CloudFormation modèle. Taille maximum de 1 kilo-octets pour l'objet JSON pouvant être stocké dans la propriété `ParameterOverrides` .
Nombre d'actions dans une étape	1 minimum, 50 maximum
Nombre d'artefacts autorisés pour chaque action	Pour connaître le nombre d'artefacts d'entrée et de sortie autorisés pour chaque action, veuillez consulter Artefacts d'entrée et de sortie valides pour chaque type d'action.
Nombre de mois pendant lesquels les informations sur l'historique d'exécution du pipeline sont conservées	12
Nombre d'étapes dans un pipeline	2 minimum, 50 maximum
Balises de pipeline	Les balises sont sensibles à la casse. Nombre maximal de 50 par ressource.

Noms de clé de balise de pipeline	Toute combinaison de lettres Unicode, de chiffres, d'espaces et de caractères autorisés d'une longueur UTF-8 comprise entre 1 et 128 caractères. Les caractères autorisés sont : + - = . _ : / @ Les noms de clé de balise doivent être uniques et chaque clé peut avoir une seule valeur. Une balise ne peut pas : • commencez par AWS : • être composée uniquement d'espaces • se terminer par un espace • contenir des émoticônes ou un des caractères suivants : ? ^ * [\ ~ ! # $ % & * ( ) > < | " '
Valeurs de balise de pipeline	Toute combinaison de lettres Unicode, de chiffres, d'espaces et de caractères autorisés d'une longueur UTF-8 comprise entre 1 et 256 caractères. Les caractères autorisés sont : + - = . _ : / @ Une clé ne peut avoir qu'une seule valeur, mais plusieurs clés peuvent avoir la même valeur. Une balise ne peut pas : • commencez par AWS : • être composée uniquement d'espaces • se terminer par un espace • contenir des émoticônes ou un des caractères suivants : ? ^ * [\ ~ ! # $ % & * ( ) > < | " '

Triggers	Il y a un maximum de 50 déclencheurs dans une définition de pipeline dans la `pull request` configuration `push` et. Il y a un maximum de trois filtres par déclenche ur push et par déclencheur pull request. **Note** Les doublons pour les filtres d'un même tableau de types d'événements ne sont pas autorisés. Vous pouvez ajouter jusqu'à 8 modèles d'inclusion et 8 modèles d'exclusion, branches et chemins de fichiers pour chaque type d'événement (push, pull request). Les caractères autorisés dans les valeurs du modèle incluent tous les types de caractères. Pour les modèles d'inclusion et d'exclusion, la longueur maximale est de 255 caractères. Pour les noms de balises, la longueur maximale est de 255 caractères. La taille maximale de la `triggers` baie ne doit pas dépasser 200 Ko

Filtres déclencheurs	Chemins de fichiers : • Nombre de modèles : vous pouvez ajouter jusqu'à 8 modèles d'inclusion et 8 modèles d'exclusion. • Taille du motif : La taille de chaque motif d'inclusion ou d'exclusion peut comporter jusqu'à 255 caractères. Succursales : • Nombre de modèles : vous pouvez ajouter jusqu'à 8 modèles d'inclusion et 8 modèles d'exclusion. • Taille du motif : La taille de chaque motif d'inclusion ou d'exclusion peut comporter jusqu'à 255 caractères. Demandes d'extraction : Succursales : • Nombre de modèles : vous pouvez ajouter jusqu'à 8 modèles d'inclusion et 8 modèles d'exclusion. • Taille du motif : La taille de chaque motif d'inclusion ou d'exclusion peut comporter jusqu'à 255 caractères.

Unicité des noms	Au sein d'un même AWS compte, chaque pipeline que vous créez dans une AWS région doit porter un nom unique. Vous pouvez réutiliser les noms des pipelines dans différent es AWS régions. Les noms des étapes doivent être uniques au sein d'un pipeline. Les noms des actions doivent être uniques au sein d'une étape.
Quotas pour les variables de sortie et les espaces de noms	Il existe une limite de taille maximale de 122 880 octets pour toutes les variables de sortie combinées pour une action particulière. Il existe une limite de taille maximale de 100 Ko pour la configuration totale de l'action résolue pour une action particulière. Les noms de variables en sortie sont sensibles à la casse. Les espaces de noms sont sensibles à la casse. Caractères autorisés : • Lettres minuscules a à z, bornes incluses. • Lettres majuscules A à Z, bornes incluses. • Chiffres 0 à 9, bornes incluses. • Caractères spéciaux ^ (caret), @ (signe arobase), - (signe moins), _ (trait de soulignement), [(crochet gauche),] (crochet droit), * (astérisque), $ (signe dollar). Tous les autres caractères, tels que des espaces, ne sont pas autorisés.

Quotas pour les variables au niveau du pipeline	Il existe un maximum de 50 variables au niveau du pipeline par pipeline. Les noms des variables au niveau du pipeline doivent être les suivants : • Longueur maximale de 128 caractères • Lettres minuscules a à z, bornes incluses. • Lettres majuscules A à Z, bornes incluses. • Chiffres 0 à 9, bornes incluses. • Caractères spéciaux `@\-_]+` Tous les autres caractères, tels que des espaces, ne sont pas autorisés. Pour les valeurs variables, la longueur maximale est de 1 000 caractères Pour les valeurs variables, tous les caractères sont autorisés. Pour les descriptions de variables, la longueur maximale est de 200 caractères.

* Vous devez activer cette région avant de pouvoir l'utiliser.

Annexe A : actions relatives aux sources GitHub (via l'application OAuth)

Cette annexe fournit des informations sur l'action GitHub (via l'application OAuth) dans. CodePipeline

> **Note**
>
> Bien que nous ne recommandons pas d'utiliser l'action GitHub (via l'application OAuth), les pipelines existants dotés de l'action GitHub (via l'application OAuth) continueront de fonctionner sans aucun impact. Pour un pipeline avec une action GitHub (via l'application OAuth), CodePipeline utilise des OAuth-based jetons pour se connecter à votre GitHub référentiel. En revanche, l' GitHub action (via GitHub App) utilise une ressource de connexion pour associer AWS des ressources à votre GitHub référentiel. La ressource de connexion utilise des jetons basés sur des applications pour se connecter. Pour plus d'informations sur la mise à jour de votre pipeline selon l' GitHubaction recommandée utilisant une connexion, consultezMettre à jour une action source GitHub (via l'application OAuth) en une action source GitHub (via GitHub l'application). Pour plus d'informations sur OAuth-based GitHub l'accès par opposition à l' GitHub accès basé sur une application, consultezhttps://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps.

Pour intégrer GitHub, CodePipeline utilise une application GitHub OAuth pour votre pipeline. CodePipelineutilise des webhooks pour gérer la détection des modifications pour votre pipeline à l'aide de l'action GitHub source (via l'application OAuth).

> **Note**
>
> Lorsque vous configurez une action source GitHub (via GitHub App) dans CloudFormation, vous n'incluez aucune information sur les GitHub jetons et vous n'ajoutez aucune ressource Webhook. Vous configurez une ressource de connexions comme indiqué dans AWS::CodeStarConnections : :Connection du Guide de l' CloudFormation utilisateur.

Cette référence contient les sections suivantes relatives à l'action GitHub (via l'application OAuth) :

- Pour plus d'informations sur la façon d'ajouter une action source GitHub (via l'application OAuth) et un webhook à un pipeline, consultez. Ajouter une action GitHub source (via l'application OAuth)

- Pour plus d'informations sur les paramètres de configuration et des exemples d' YAML/JSON extraits pour une action source GitHub (via l'application OAuth), consultez. GitHub (via l'application OAuth) référence de l'action source

> **Important**
>
> Lorsque vous créez des CodePipeline webhooks, n'utilisez pas vos propres informations d'identification et ne réutilisez pas le même jeton secret sur plusieurs webhooks. Pour une sécurité optimale, générez un jeton secret unique pour chaque webhook que vous créez. Le jeton secret est une chaîne arbitraire que vous fournissez, qui est GitHub utilisée pour calculer et signer les charges utiles du webhook envoyées CodePipeline, afin de protéger l'intégrité et l'authenticité des charges utiles du webhook. L'utilisation de vos propres informations d'identification ou la réutilisation du même jeton sur plusieurs webhooks peut entraîner des failles de sécurité.

> **Note**
>
> Si un jeton secret a été fourni, il sera supprimé dans la réponse.

Rubriques

- Ajouter une action GitHub source (via l'application OAuth)
- GitHub (via l'application OAuth) référence de l'action source

Ajouter une action GitHub source (via l'application OAuth)

Vous ajoutez GitHub (via l'application OAuth) des actions source pour CodePipeline :

- À l'aide de l'assistant de création de pipeline (Création d'un pipeline personnalisé (console)) de la CodePipeline console ou de la page Modifier l'action pour choisir l'option du GitHub fournisseur. La console crée un webhook qui démarre votre pipeline lorsque la source change.
- Utilisation de l'interface de ligne de commande pour ajouter la configuration de l'`GitHub`action et créer des ressources supplémentaires comme suit :

- Utilisation de l'`GitHub`exemple de configuration d'action GitHub (via l'application OAuth) référence de l'action source pour créer l'action comme indiqué dansCréation d'un pipeline (interface de ligne de commande).
- Désactivation des contrôles périodiques et création manuelle de la détection des modifications, car la méthode de détection des modifications démarre par défaut le pipeline en interrogeant la source. Vous migrez votre pipeline de sondages vers des webhooks pour les actions GitHub (via l'application OAuth).

GitHub (via l'application OAuth) référence de l'action source

> **Note**
>
> Bien que nous ne recommandons pas d'utiliser l'action GitHub (via l'application OAuth), les pipelines existants dotés de l'action GitHub (via l'application OAuth) continueront de fonctionner sans aucun impact. Pour un pipeline avec une action source GitHub (via l'application OAuth), CodePipeline utilise des OAuth-based jetons pour se connecter à votre GitHub référentiel. En revanche, la nouvelle GitHub action (via GitHub App) utilise une ressource de connexion pour associer AWS des ressources à votre GitHub référentiel. La ressource de connexion utilise des jetons basés sur des applications pour se connecter. Pour plus d'informations sur la mise à jour de votre pipeline selon l' GitHub action recommandée utilisant une connexion, consultezMettre à jour une action source GitHub (via l'application OAuth) en une action source GitHub (via GitHub l'application).

Déclenche le pipeline lorsqu'un nouveau commit est effectué sur le GitHub référentiel et la branche configurés.

Pour l'intégrer GitHub, CodePipeline utilise une application OAuth ou un jeton d'accès personnel pour votre pipeline. Si vous utilisez la console pour créer ou modifier votre pipeline, CodePipeline crée un GitHub webhook qui démarre votre pipeline lorsqu'une modification se produit dans le référentiel.

Vous devez déjà avoir créé un GitHub compte et un référentiel avant de connecter le pipeline par le biais d'une GitHub action.

Si vous souhaitez limiter l'accès aux CodePipeline référentiels, créez un GitHub compte et accordez-lui l'accès uniquement aux référentiels auxquels vous souhaitez vous intégrer. CodePipeline Utilisez

ce compte lorsque vous configurez CodePipeline l'utilisation de GitHub référentiels pour les étapes source dans les pipelines.

Pour plus d'informations, consultez la documentation GitHub destinée aux développeurs sur le GitHub site Web.

Rubriques

- Type d'action
- Paramètres de configuration
- Artefacts d'entrée
- Artefacts de sortie
- Variables de sortie
- Déclaration d'action (GitHub exemple)
- Connexion à GitHub (OAuth)
- Consultez aussi

Type d'action

- Catégorie : `Source`
- Propriétaire : `ThirdParty`
- Fournisseur : `GitHub`
- Version : `1`

Paramètres de configuration

Propriétaire

Obligatoire : oui

Le nom de l' GitHub utilisateur ou de l'organisation propriétaire du GitHub référentiel.

Repo

Obligatoire : oui

Nom du référentiel où les modifications de la source doivent être détectées.

Branche

Obligatoire : oui

Nom de la branche où les modifications de la source doivent être détectées.

OAuthToken

Obligatoire : oui

Représente le jeton GitHub d'authentification qui permet CodePipeline d'effectuer des opérations sur votre GitHub référentiel. L'entrée est toujours affichée sous la forme d'un masque de quatre astérisques. Celui-ci représente l'une des valeurs suivantes :

- Lorsque vous utilisez la console pour créer le pipeline, CodePipeline utilise un jeton OAuth pour enregistrer la GitHub connexion.
- Lorsque vous utilisez le AWS CLI pour créer le pipeline, vous pouvez transmettre votre jeton d'accès GitHub personnel dans ce champ. Remplacez les astérisques (****) par votre jeton d'accès personnel copié depuis. GitHub Lorsque vous exécutez `get-pipeline` pour afficher la configuration de l'action, le masque à quatre astérisques s'affiche pour cette valeur.
- Lorsque vous utilisez un CloudFormation modèle pour créer le pipeline, vous devez d'abord stocker le jeton en tant que secret dans AWS Secrets Manager. Vous incluez la valeur de ce champ en tant que référence dynamique au secret stocké dans Secrets Manager, tel que`{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}`.

Pour plus d'informations sur GitHub les scopes, consultez le GitHub Developer API Reference sur le GitHub site Web.

PollForSourceChanges

Obligatoire : non

`PollForSourceChanges`contrôle si CodePipeline les modifications apportées à la source sont interrogées dans le GitHub référentiel. Nous vous recommandons plutôt d'utiliser des webhooks pour détecter les modifications de la source. Pour plus d'informations sur la configuration des webhooks, consultez Migrer les pipelines de sondage vers les webhooks GitHub (via l'application OAuth) (actions source) (CLI) ou Mettre à jour les pipelines pour les actions source des événements push GitHub (via l'application OAuth)) (CloudFormation modèle).

> **Important**
>
> Si vous avez l'intention de configurer des webhooks, vous devez définir `PollForSourceChanges` sur `false` pour éviter les exécutions de pipeline en double.

Valeurs valides pour ce paramètre :

- `True`: si cette option est définie, CodePipeline interroge votre référentiel pour détecter les modifications apportées à la source.

  > **Note**
  >
  > Si vous l'omettez`PollForSourceChanges`, la CodePipeline valeur par défaut est interrogée dans votre référentiel pour détecter les modifications apportées à la source. Ce comportement est le même que si `PollForSourceChanges` est défini sur `true`.

- `False`: Si cette option est définie, CodePipeline n'interroge pas votre référentiel pour détecter les modifications apportées à la source. Utilisez ce paramètre si vous avez l'intention de configurer un webhook pour détecter les modifications de la source.

Artefacts d'entrée

- Nombre d'objets : `0`
- Description : Les artefacts d'entrée ne s'appliquent pas à ce type d'action.

Artefacts de sortie

- Nombre d'objets : `1`
- Description : l'artefact de sortie de cette action est un fichier ZIP qui regroupe le contenu du référentiel et de la branche configurés au moment de la validation spécifiée comme révision source pour l'exécution du pipeline. Les artefacts générés à partir du référentiel sont les artefacts de sortie de l' GitHub action. L'ID de validation du code source est affiché en CodePipeline tant que révision source pour l'exécution du pipeline déclenchée.

Variables de sortie

Lorsque cette action est configurée, elle produit des variables qui peuvent être référencées par la configuration d'action d'une action en aval dans le pipeline. Cette action produit des variables qui peuvent être visualisées en tant que variables de sortie, même si l'action n'a pas d'espace de noms. Vous configurez une action avec un espace de noms pour rendre ces variables disponibles pour la configuration des actions en aval.

Pour plus d'informations sur les variables dans CodePipeline, consultezRéférence des variables.

CommitId

L'ID de GitHub validation qui a déclenché l'exécution du pipeline. Les ID de validation sont le SHA complet de la validation.

CommitMessage

Message de description, le cas échéant, associé à la validation ayant déclenché l'exécution du pipeline.

CommitUrl

Adresse URL de la validation ayant déclenché le pipeline.

RepositoryName

Le nom du GitHub référentiel dans lequel le commit qui a déclenché le pipeline a été effectué.

BranchName

Nom de la branche du GitHub référentiel dans lequel la modification de source a été effectuée.

AuthorDate

Date à laquelle la validation a été créée, au format horodatage.

CommitterDate

Date à laquelle la validation a été validée, au format horodatage.

Déclaration d'action (GitHub exemple)

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"
        }
    ]
},
```

Connexion à GitHub (OAuth)

La première fois que vous utilisez la console pour ajouter un GitHub référentiel à un pipeline, il vous est demandé d'autoriser CodePipeline l'accès à vos référentiels. Le jeton nécessite les GitHub étendues suivantes :

- La portée `repo`, qui est utilisée pour contrôler entièrement la lecture et l'extraction des artefacts dans un pipeline à partir de référentiels publics et privés.
- La portée `admin:repo_hook`, qui est utilisée pour contrôler entièrement les hooks de référentiel.

Lorsque vous utilisez l'interface de ligne de commande ou un CloudFormation modèle, vous devez fournir la valeur d'un jeton d'accès personnel que vous avez déjà créé dans GitHub.

Consultez aussi

Les ressources connexes suivantes peuvent s'avérer utiles dans le cadre de l'utilisation de cette action.

- Référence de ressources pour le Guide de AWS CloudFormation l'utilisateur AWS::CodePipeline : :Webhook — Cela inclut des définitions de champs, des exemples et des extraits de code pour la ressource dans. CloudFormation
- Référence des ressources pour le Guide de AWS CloudFormation l'utilisateur AWS::CodeStar : : GitHubRepository — Cela inclut des définitions de champs, des exemples et des extraits de code pour la ressource contenue dans. CloudFormation
- Tutoriel : Créez un pipeline qui crée et teste votre application Android avec AWS Device Farm— Ce didacticiel fournit un exemple de fichier buildspec et un exemple d'application pour créer un pipeline avec une GitHub source. Il crée et teste une application Android avec CodeBuild et AWS Device Farm.

AWS CodePipeline Historique du document du guide de l'utilisateur

Le tableau suivant décrit les modifications importantes apportées à chaque version du guide de l' CodePipeline utilisateur. Pour recevoir les notifications de mise à jour de cette documentation, abonnez-vous à un flux RSS.

- Version de l'API : 2015-07-09
- Dernière mise à jour de la documentation : 23 mai 2025

Modification	Description	Date
Documentation corrigée concernant le comportement du filtre de déclenchement du pipeline	Correction de la documentation pour les exemples de filtres de déclenchement afin de décrire avec précision la relation entre les inclusions et les exclusions. Les exclusions ont priorité sur les inclusions pour les filtres de branches et de chemins de fichiers. Pour plus d'informations, consultez la section Exemples de filtres de déclenchement.	16 mars 2026
Support ajouté pour Azure DevOps	Support ajouté pour la configuration des connexions pour que les AWS ressources interagissent avec Azure DevOps. Pour plus d'informations, consultez Connections for Azure DevOps.	5 août 2025
Documentation mise à jour pour le fichier de spécifications	Ajout de nouvelles informations sur l'utilisation du fichier	16 mai 2025

de déploiement de l'action de déploiement EC2	de spécifications de déploiement pour l'action de déploiement EC2 avec des exemples et des options de configuration mis à jour. Consultez la page de référence des actions sur Amazon EC2 Action Reference.	
Documentation ajoutée pour l'action de déploiement Lambda	Ajout de la prise en charge de la nouvelle action de déploiement Lambda avec des exemples et des options de configuration mis à jour. Consultez la page de référence des actions dans la section Référence AWS Lambda des actions de déploiement.	16 mai 2025
Ajouter des champs à l'action Commandes pour le type de calcul, le type d'environnement et le type de variable d'environnement	Ajout d'informations sur les nouveaux champs qui vous permettent de spécifier un environnement de génération et un type de calcul pour l'action Commandes. Un nouveau champ de type pour les variables d'environnement a également été ajouté pour l'action Commandes. Consultez la page de référence des actions sur CodePipeline Comands action.	8 mai 2025

Nouvelle rubrique pour documenter la politique de rôle de service par défaut	Ajout d'informations sur la politique de rôle de service minimum et mise à jour du tableau des liens vers des autorisations de rôle de service supplémentaires pour chaque action dans CodePipeline. Consultez la nouvelle page de référence des règles sur la politique des rôles de CodePipeline service.	26 mars 2025
Nouvelle action `CodePipeline` d'appel	Ajout d'informations sur la nouvelle `CodePipeline` action d'appel. Consultez la page de référence des actions sur CodePipeline Invoke Action Reference.	14 mars 2025
Nouvelle `CodeBuild` règle	Ajout d'informations sur la nouvelle `CodeBuild` règle disponible pour exécuter des projets de construction en tant que règle pour les conditions d'étape. Consultez la nouvelle page de référence des règles sous la rubrique Référence des CodeBuild règles.	14 mars 2025

Partage de connexions dans AWS CodeConnections	Pour les pipelines dont les connexions sont utilisées AWS CodeConnections, vous pouvez utiliser des connexions configurées pour le partage entre elles Comptes AWS. Vous pouvez configurer des connexion s partagées dans AWS Resource Access Manager. Pour plus d'informations, voir Utiliser une connexion partagée avec une autre personne Compte AWS.	6 mars 2025
EventBridge entrée de transformation pour les révisions de source	Pour les pipelines contenant CodeCommit des sources Amazon ECR et Amazon S3, vous pouvez utiliser l'entrée de transformation EventBrid ge d'entrée pour les révisions de source, où elle `revisionV alue` est dérivée de la variable d'événement source pour votre clé d'objet (S3), votre commit (CodeCommit) ou votre identifiant d'image (ECR). Consultez les actions et EventBridge ressource s source Amazon ECR, la connexion aux actions source Amazon S3 avec une source activée pour les événement s, et les actions CodeCommit source et EventBridge.	3 mars 2025

Remplacer les spécifications de construction pour l'action AWS CodeBuild	Vous pouvez choisir de remplacer les spécifications de construction de l' CodeBuild action tout en saisissant directement les commandes . Consultez la page de référence des actions sur la page de référence des actions de AWS CodeBuild génération et de test. Consultez également la section Création d'un pipeline, d'étapes et d'actions.	3 mars 2025
Nouvelle action de EC2 déploiement	Ajout d'informations sur la nouvelle action de EC2 déploiement. Consultez la page de référence des actions sur Amazon EC2 Action Reference. Pour un didacticiel, voir Tutoriel : Déployer sur des instances EC2 avec CodePipeline.	21 février 2025
Nouvelle action de EKS déploiement	Ajout d'informations sur la nouvelle action de EKS déploiement. Consultez la page de référence des actions sur EKS deploy action. Pour un didacticiel, consultez Tutoriel : Déployer sur Amazon EKS avec CodePipeline.	20 février 2025

Nouvelles CloudWatch mesures et dimensions pour CodePipeline	Ajout de métriques pour les pipelines dans CloudWatch les métriques. Voir CodePipeline CloudWatch les métriques.	13 février 2025
Nouvelle `Commands` règle	Ajout d'informations sur la nouvelle `Commands` règle disponible pour exécuter des commandes shell en tant que règle pour les conditions d'étape. Consultez la nouvelle page de référence des règles dans la section Référence des règles des commandes.	17 décembre 2024
Exemples détaillés et informations de référence pour les déclencheurs	Ajout de descriptions pour les filtres d'événements de pull request par fournisseur dans les événements de pull request pour les déclencheurs par fournisseur. Ajout d'exemples étendus pour les déclencheurs avec des informations plus détaillées sur les inclusions et les exclusions dans les événements push et sur les événements de pull request dans les événements de demande d'extraction pour les déclencheurs par fournisseur. Ajout d'informations de référence JSON supplémentaires sur les inclusions et les exclusions dans les déclencheurs.	17 décembre 2024

Nouvelles actions dans le catalogue d'actions	Vous pouvez désormais utiliser les `InspectorScan` actions `ECRBuildAndPublish` et. Pour plus d'informations, consultez les pages de référence ECRBuildAndPublishet les InspectorScanactions.	22 novembre 2024
Nouvelle configuration automatique pour une nouvelle tentative d'étape en cas d'échec	Vous pouvez configurer une étape pour réessayer automatiquement une étape ayant échoué ou des actions ayant échoué pendant la phase. Pour plus d'informations, voir Configurer une étape pour une nouvelle tentative automatique en cas d'échec.	15 octobre 2024

Nouveau `Skip` résultat pour les conditions d'entrée	Ajout d'informations sur le `Skip` résultat disponibles pour les conditions d'entrée. Vous pouvez utiliser les `LambdaInvoke` règles `VariableCheck` et avec cette configuration. Consultez les étapes de la section Création de conditions de saisie avec le résultat Ignorer. Pour obtenir la liste des considérations relatives aux conditions dans lesquelles les résultats sont ignorés, voir Considérations relatives aux résultats configurés pour les conditions d'étape.	15 octobre 2024
Nouvelles étapes de console pour créer des pipelines à partir de modèles statiques	Dans la CodePipeline console, vous pouvez utiliser un nouvel assistant de création de pipeline pour choisir parmi plusieurs modèles statiques dans lesquels générer des ressources de pipeline CloudFormation. Pour plus d'informations, voir Création d'un pipeline à partir de modèles statiques.	9 octobre 2024

Nouvelle `Commands` action	Ajout d'informations sur la nouvelle `Commands` action disponible pour exécuter des commandes shell en tant qu'action dans votre pipeline. Consultez la nouvelle page de référence des actions dans Action des commandes et les autorisations du rôle de service dans Ajouter des autorisations au rôle CodePipeline de service. Pour un didacticiel, voir Tutoriel : Création d'un pipeline qui exécute des commandes avec compute.	3 octobre 2024
Nouvelle `VariableCheck` règle pour les conditions	Ajout d'informations sur la `VariableCheck` règle relative aux conditions d'étape. Consultez la nouvelle page de référence des règles à l'adresse VariableCheck. Pour un didacticiel, voir Tutoriel : créer une règle de vérification des variables pour un pipeline en tant que condition d'entrée.	27 septembre 2024

Mises à jour des connexions pour GitHub	Ajout d'informations sur l'utilisation du jeton GitHub d'accès utilisateur avec des connexions à GitHub (actions GitHub V2). Les jetons d'accès utilisateur sont utilisés avec les CodeBuild projets. Voir GitHub connexions et didacticiel : Utiliser un clone complet avec une source de GitHub pipeline.	16 septembre 2024
Mises à jour pour restructurer la référence JSON du pipeline et la table des matières du guide	Le guide a été restructuré, y compris certaines modifications du titre des sections, afin d'améliorer l'utilisabilité des sections de référence et de tâches.	16 août 2024
Mise à jour du champ du jeton secret dans la réponse `PutWebhook` et les `ListWebhooks` actions	Le champ du jeton secret pour les `ListWebhooks` actions `PutWebhook` et a été mis à jour. Si un jeton secret a été fourni, il sera supprimé dans la réponse. Voir les notes ajoutées à l'annexe A : actions source de GitHub la version 1. Pour les mises à jour associées dans le guide de CodePipeline l'API, consultez PutWebhooket ListWebhooks.	6 août 2024

Ajout de nouveau contenu pour les conditions et les règles de la scène	Vous pouvez désormais configurer les conditions et les règles d'étape pour les pipelines de type V2. Voir Concepts, Comment fonctionnent les conditions scéniques ? , et Configurer les conditions d'une étape. Un chapitre de référence sur les règles a été ajouté pour fournir des informations de référence. Voir la référence aux CodePipeline règles.	30 juillet 2024
Ajout de nouvelles informations de référence pour les types de pipelines et les fonctionnalités associées	Un nouveau script d'analyse est disponible pour évaluer le coût du passage à des pipelines de type V2. Voir Quel type de pipeline me convient le mieux ? . Une table de référence a été ajoutée qui fournit des liens vers toute la documentation du CodePipeline service par fonctionnalité. Consultez la référence des CodePipeline fonctionnalités.	11 juillet 2024

Mises à jour de l'action S3 source pour ajouter une nouvelle option pour les remplacements de source	Une nouvelle option pour les remplacements de source nommés `S3_OBJECT_KEY` est disponible pour l'action S3 source. Un nouveau `AllowOverrideForS3ObjectKey` paramètre a été ajouté pour l'action source S3. Consultez la page de référence des actions source Amazon S3 et démarrez un pipeline avec une dérogation de révision de source.	7 juin 2024
Mises à jour de l'action S3 source pour ajouter de nouvelles variables de sortie	De nouvelles variables de sortie `ObjectKey` sont nommées `BucketName` et disponibles pour l'action S3 source. Consultez la page de référence des actions source Amazon S3.	5 juin 2024
Mises à jour `CloudFormationStackSet` des `CloudFormationStackInstances` actions et	Le `CallAs` paramètre a été ajouté pour l'`CloudFormationStackInstances` action `CloudFormationStackSet` et. Consultez la page de référence des actions.	2 mai 2024

Support pour les annulations au niveau des étapes	Vous pouvez restaurer manuellement ou automatiquement une étape à une précédente exécution de pipeline réussie pour l'étape. Consultez la section Configuration de la restauration des étapes et des concepts.	26 avril 2024
Mises à jour de la disponibilité des régions StackSets et des actions Step Functions	Les actions StackSets et Step Functions sont désormais disponibles dans toutes les régions où elles CodePipeline sont disponibles. Voir la référence des CloudFormation StackSets actions et la référence des actions AWS Step Functions.	27 mars 2024
Mises à jour de la politique gérée	La politique AWS gérée `AWSCodePipeline_FullAccess` a été mise à jour. Voir les politiques AWS gérées pour AWS CodePipeline.	15 mars 2024
Support du délai d'expiration configurable pour les actions d'approbation manuelles	Informations de quota ajoutées pour le nouveau champ de délai d'expiration configurable pour les actions d'approbation manuelles. Pour de plus amples informations, veuillez consulter Quotas .	15 février 2024

Support pour le filtrage des déclencheurs par branches et chemins de fichiers	Support ajouté pour la configuration des déclencheurs qui permet de filtrer l'état des pull requests, les branches et les chemins de fichiers pour les pipelines de type V2. Pour plus d'informations, consultez Filtrer les déclencheurs sur les requêtes push ou pull de code, les déclencheurs et le filtre sur les branches de fonctionnalités pour démarrer votre pipeline, et Quotas.	8 février 2024
Support pour les nouveaux modes d'exécution du pipeline	Support ajouté pour les modes d'exécution de pipeline PARALLEL et QUEUED. Pour plus d'informations, voir Définir le mode d'exécution du pipeline, Comment les exécutions sont traitées en mode QUEUED, Comment les exécutions sont traitées en mode PARALLÈLE et Quotas.	8 février 2024

Mises à jour des pages de console permettant d'afficher les détails des actions, de consulter les actions d'approbation manuelles et de la page des pipelines de liste	Mises à jour de la console documentées pour le nouveau bouton et la nouvelle boîte de dialogue Afficher les détails, la nouvelle boîte de dialogue d'approbation manuelle et les nouvelles colonnes pour les exécutions récentes sur la page des pipelines de liste. Pour plus d'informations, voir Afficher les pipelines (console), Afficher les détails des actions dans un pipeline et Gérer les actions d'approbation dans les pipelines.	10 janvier 2024
Support pour l' GitLab autogestion	Support ajouté pour la configuration des connexions pour que les AWS ressources puissent interagir avec GitLab l'autogestion. Pour plus d'informations, consultez la section Connexions pour l' GitLabautogestion.	28 décembre 2023
Mises à jour `CloudFormationStackSet` des `CloudFormationStackInstances` actions et	Le `ConcurrencyMode` paramètre a été ajouté pour l'`CloudFormationStackInstances` action `CloudFormationStackSet` et. Consultez la page de référence des actions.	19 décembre 2023

Mises à jour AWS Device Farm des paramètres d'action dans CodePipeline	Les paramètres de l' AWS Device Farm action dans CodePipeline ont été mis à jour. Pour plus d'informations, reportez-vous à la section Référence des AWS Device Farm actions.	18 décembre 2023
Support ajouté pour les messages d'erreur détaillés relatifs à l' CloudFormation action dans CodePipeline	CloudFormation les messages d'erreur d'action peuvent désormais faire apparaître des détails sur les ressource s défaillantes. Pour plus d'informations, reportez-vous à la section Référence des CloudFormation actions.	15 décembre 2023
Mises à jour pour démarrer un pipeline avec des remplacem ents de révisions de source dans CodePipeline	Vous pouvez désormais démarrer un pipeline avec une révision de source spécifiée. Pour plus d'informations, voir Démarrer un pipeline avec une modification de version source.	17 novembre 2023

Nouvelles régions prises en charge	CodePipeline est désormais disponible dans les régions Asie-Pacifique (Hyderabad), Asie-Pacifique (Jakarta), Asie-Pacifique (Melbourne), Asie-Pacifique (Osaka), Moyen-Orient (Émirats arabes unis), Europe (Espagne) et Israël (Tel Aviv). Les rubriques de référence relatives aux compartiments réservés aux événements et aux Service AWS points de terminaison ont été mises à jour.	13 novembre 2023
Mises à jour pour les champs d'événements sur Amazon EventBridge	Vous pouvez désormais consulter les champs d'événements mis à jour sur Amazon EventBridge. Pour plus d'informations, consultez la section Surveillance CodePipeline des événements.	9 novembre 2023

Mises à jour pour les nouveaux pipelines de type V2, les déclencheurs sur les balises Git et les variables de pipeline dans CodePipeline	Vous pouvez désormais choisir un type de pipeline dans CodePipeline. Pour un pipeline de type V2, vous pouvez désormais utiliser une configuration de déclencheur pour démarrer votre pipeline sur des balises Git. Avec les pipelines de type V2, vous pouvez également utiliser des variables au niveau du pipeline pour transmettre des paramètres d'entrée pour une exécution de pipeline. Pour plus d'informations, consultez Variables, Tutoriel : utilisation de variables au niveau du pipeline et Tutoriel : utilisation des balises Git pour démarrer votre pipeline. Pour plus d'informations sur les types de conduites, consultez la section Types de conduites.	24 octobre 2023
CodePipeline permet de réessayer toutes les actions en cas d'échec	En cas d'échec d'une étape dans CodePipeline, vous pouvez réessayer l'étape sans relancer le pipeline. Pour ce faire, réessayez les actions qui ont échoué dans une étape ou réessayez toutes les actions de la phase en commençant par la première action de la phase. Pour plus d'informations, consultez.	17 octobre 2023

Support aux GitLab groupes	Support ajouté pour configurer les connexions pour que les AWS ressources interagissent avec GitLab les groupes. Pour plus d'informations, consultez la section GitLab Connexions.	15 septembre 2023
CodePipeline prend en charge les connexions à GitLab .com	Vous pouvez utiliser les connexions pour configurer les AWS ressources afin d'interagir avec GitLab .com. Vous pouvez également choisir l'option de clonage complet pour utiliser les commandes et les métadonnées Git pour les actions en aval. Pour plus d'informations, consultez la rubrique de référence sur les GitLab connexions et la structure CodeStarSourceConnection d'action.	10 août 2023
Mise à jour de l'`CloudFormationStackInstances` action	Le `RegionConcurrencyType` paramètre a été ajouté pour l'`CloudFormationStackInstances` action. Consultez la page de référence de l'action pour l'`CloudFormationStackInstances` action.	08 août 2023

Mise à jour de l'`CloudFormationStackSet` action	Le `RegionConcurrencyType` paramètre a été ajouté pour l'`CloudFormationStackSet` action. Consultez la page de référence de l'action pour l'`CloudFormationStackSet` action.	24 juillet 2023
Mises à jour de la politique gérée	La politique AWS gérée `AWSCodePipeline_FullAccess` a été mise à jour. Voir les politiques AWS gérées pour AWS CodePipeline.	21 juin 2023
Mises à jour des procédures de migration pour les pipelines de sondage	Les procédures de migration (mise à jour) des pipelines d'interrogation afin d'utiliser la détection des modifications basée sur les événements ont été mises à jour avec les étapes pour les pipelines qui utilisent un compartiment Amazon S3 activé pour les notifications destinées à. EventBridge Pour plus d'informations, voir Migrer les pipelines de sondage pour utiliser la détection des modifications basée sur les événements.	12 juin 2023

Mises à jour des politiques gérées	Les politiques AWS `AWSCodePipeline_FullAccess` gérées `AWSCodePipeline_ReadOnlyAccess` ont été mises à jour avec une autorisation supplémentaire. Pour plus d'informations, voir les AWS CodePipeline mises à jour des politiques AWS gérées.	16 mai 2023
Mises à jour des politiques gérées	Les politiques AWS `AWSCodePipelineFullAccess` gérées `AWSCodePipelineReadOnlyAccess` sont obsolètes. Utilisez les `AWSCodePipeline_ReadOnlyAccess` politiques `AWSCodePipeline_FullAccess` et. Consultez les AWS CodePipeline mises à jour des politiques AWS gérées.	17 novembre 2022

Mises à jour des procédures qui utilisent CloudTrail	Toutes les procédures de console, les exemples de commandes CLI et les exemples d' CloudFormation extraits et de modèles pour un pipeline avec une source S3 ont été mis à jour avec l'option de choisir Write et de sélectionner false pour les événements de gestion dans. CloudTrail Consultez les exemples mis à jour dans Démarrage d'un pipeline, Tutoriel : créer un pipeline avec CloudFormation, Modifier les pipelines pour utiliser les événements push et Mettre à jour les pipelines de sondage.	27 avril 2022
Nouvelle intégration prise en charge avec Snyk	Vous pouvez utiliser l'action Snyk Invoke CodePipeline pour automatiser l'analyse de sécurité de votre code open source. Pour plus d'informations, reportez-vous à la référence des actions Snyk et aux intégrations.	10 juin 2021
Nouvelle région Europe prise en charge (Milan)	CodePipeline est désormais disponible en Europe (Milan). Les rubriques Limites et Service AWS points de terminaison ont été mises à jour.	27 janvier 2021

La détection des modifications peut être désactivée pour les actions source associées à des connexions	Vous pouvez utiliser la CLI ou le SDK pour mettre à jour une action `CodeStarSourceConnection` source afin de désactiver la détection automatique des modifications pour le référentiel source. La rubrique de référence de la structure d'CodeStarSourceConnection action a été mise à jour avec une description du `DetectChanges` paramètre.	8 janvier 2021
CodePipeline prend désormais en charge les actions CloudFormation StackSets de déploiement	Un nouveau didacticiel, Tutoriel : Création d'un pipeline utilisé CloudFormation StackSets comme fournisseur de déploiement, décrit les étapes à suivre pour créer et mettre CloudFormation StackSets à jour vos ensembles de piles et vos instances de pile avec votre pipeline. La rubrique de référence de la structure d'CloudFormation StackSets action a également été ajoutée.	30 décembre 2020
Nouvelle région prise en charge Asie-Pacifique (Hong Kong)	CodePipeline est désormais disponible en Asie-Pacifique (Hong Kong). Les rubriques Limites et Service AWS points de terminaison ont été mises à jour.	22 décembre 2020

Afficher les modèles EventBridge d'événements mis à jour dans CodePipeline	Les modèles d'événements et les statuts mis à jour pour les événements du pipeline, de l'étape et du niveau d'action ont été ajoutés à la section Surveillance des CodePipeline événements.	21 décembre 2020
Afficher les exécutions du pipeline entrant dans CodePipeline	Vous pouvez utiliser la console ou la CLI pour visualiser les exécutions entrantes . Pour plus d'informations, voir Afficher une exécution entrante (console) et Afficher le statut d'exécution entrante (CLI).	16 novembre 2020
L'action CodeCommit source dans CodePipeline prend en charge l'option de clonage complet	Lorsque vous utilisez une action CodeCommit source, vous pouvez choisir l'option de clonage complet pour utiliser les commandes et les métadonnées Git pour les CodeBuild actions en aval. Pour plus d'informations, consultez la référence CodeCommit d'action et le didacticiel : Utiliser un clone complet avec une source de CodeCommit pipeline.	11 novembre 2020

CodePipeline prend en charge les connexions à GitHub et GitHub Enterprise Server	Vous pouvez utiliser les connexions pour configure r les AWS ressources avec lesquelles interagir avec GitHub GitHub Enterprise Cloud et GitHub Enterpris e Server. Vous pouvez également choisir l'option de clonage complet pour utiliser les commandes et les métadonnées Git pour les actions en aval. Pour plus d'informations, consultez GitHub les sections Connexions, Connexions aux serveurs GitHub d'entreprise et Tutoriel : Utiliser un clone complet avec une source de GitHub pipeline. Si vous avez un pipeline existant avec une action GitHub source, voir Mettre à jour une action source GitHub (via une OAuth application) en une action source GitHub (via une GitHub application).	30 septembre 2020

L' CodeBuild action permet d'activer les intégrations par lots dans AWS CodePipeline	Pour les CodeBuild actions de votre pipeline, vous pouvez activer les builds par lots afin d'exécuter plusieurs builds en une seule exécution . Pour plus d'informations, consultez les CodeBuild sections Référence de structure d'action et Création d'un pipeline (console).	30 juillet 2020
AWS CodePipeline prend désormais en charge les actions AWS AppConfig de déploiement	Un nouveau didacticiel, Tutoriel : Création d'un pipeline utilisé AWS AppConfig comme fournisseur de déploiement, décrit les étapes à suivre AWS AppConfig pour déployer des fichiers de configuration avec votre pipeline. La rubrique de référence de la structure d'AWS AppConfig action a également été ajoutée.	25 juin 2020
AWS CodePipeline prend désormais en charge Amazon VPC en AWS GovCloud (ouest des États-Unis)	Vous pouvez désormais vous connecter directement AWS CodePipeline via un point de terminaison Amazon VPC privé en AWS GovCloud (ouest des États-Unis). Pour plus d'informations, consultez Utiliser CodePipeline avec Amazon Virtual Private Cloud.	2 juin 2020

AWS CodePipeline prend désormais en charge les actions AWS Step Functions d'invocation	Vous pouvez désormais créer un pipeline CodePipeline qui l'utilise AWS Step Functions comme fournisseur d'actions d'appel. Un nouveau didacticiel, Tutoriel : Utiliser une action AWS Step Functions d'appel dans un pipeline, décrit les étapes à suivre pour démarrer une exécution par machine à états à partir de votre pipeline. La rubrique Référence de la structure d'action AWS Step Functions a également été ajoutée.	28 mai 2020
Afficher, répertorier et mettre à jour les connexions	Vous pouvez répertorier, supprimer et mettre à jour les connexions dans la console. Voir Répertorier les connexions dans CodePipeline.	21 mai 2020
Les connexions prennent en charge le balisage des ressources de connexion dans la CLI	Les ressources de connexion prennent désormais en charge le balisage dans la AWS CLI. Les connexions s'intègrent désormais à AWS CodeGuru. Veuillez consulter Référence des autorisations IAM pour les connexions.	6 mai 2020
CodePipeline est désormais disponible en AWS GovCloud (ouest des États-Unis)	Vous pouvez désormais l'utiliser CodePipeline dans AWS GovCloud (US-West). Pour de plus amples informations, veuillez consulter Quotas .	8 avril 2020

La rubrique relative aux quotas indique quels quotas CodePipeline de service sont configurables.	La rubrique sur CodePipeline les quotas a été reformatée. La documentation indique quels quotas de service sont configurables et quels quotas ne sont pas configurables. Voir Quotas dans AWS CodePipeline.	12 mars 2020
Le délai d'expiration de l'action de déploiement d'Amazon ECS est configurable	Le délai d'expiration de l'action de déploiement Amazon ECS est configurable jusqu'à une heure (délai d'expiration par défaut). Veuillez consulter Quotas dans AWS CodePipeline.	5 février 2020
De nouvelles rubriques décrivent comment arrêter l'exécution d'un pipeline	Vous pouvez arrêter l'exécution d'un pipeline dans CodePipeline. Vous pouvez spécifier que l'exécution s'arrête une fois que les actions en cours sont autorisées à se terminer, ou vous pouvez spécifier que l'exécution s'arrête immédiatement et que les actions en cours soient abandonnées. Veuillez consulter Arrêt des exécutions de pipeline et Arrêt de l'exécution d'un pipeline dans CodePipeline.	21 janvier 2020

CodePipeline prend en charge les connexions	Vous pouvez utiliser les connexions pour configure r les AWS ressources afin d'interagir avec des référentiels de code externes. Chaque connexion est une ressource qui peut être utilisée par des services, par exemple CodePipeline pour se connecter à un référenti el tiers, tel que Bitbucket Cloud. Pour plus d'informa tions, consultez la section Utilisation des connexions dans CodePipeline.	18 décembre 2019
Rubriques mises à jour sur la sécurité, l'authentification et le contrôle d'accès	Les informations relatives à la sécurité, à l'authentification et au contrôle d'accès ont CodePipeline été organisées dans un nouveau chapitre sur la sécurité. Pour plus d'informa tions, consultez Sécurité.	17 décembre 2019
De nouvelles rubriques décrivent comment utiliser des variables dans vos pipelines	Vous pouvez désormais configurer des espaces de noms pour les actions et générer des variables à chaque fois que l'exécution de l'action est terminée. Vous pouvez configurer des actions en aval pour référencer ces espaces de noms et variables . Veuillez consulter Utilisation des variables et Variables.	14 novembre 2019

De nouvelles rubriques décrivent le fonctionnement des exécutions de pipeline, les raisons pour lesquelles les étapes sont verrouillées pendant une exécution et les circonstances dans lesquelles les exécutions de pipeline sont remplacées	Un certain nombre de rubriques ont été ajoutées à la section Bienvenue pour décrire le fonctionnement des exécutions de pipeline, y compris les raisons pour lesquelles les étapes sont verrouillées pendant une exécution et ce qui se passe lorsque les exécutions de pipeline sont remplacée s. Ces rubriques incluent une liste de concepts, un exemple de DevOps flux de travail et des recommand ations sur la manière dont un pipeline doit être structuré . Les rubriques suivantes ont été ajoutées : termes du pipeline, exemple de DevOps pipeline et fonctionnement des exécutions de pipeline.	11 novembre 2019
CodePipeline prend en charge les règles de notification	Vous pouvez désormais utiliser des règles de notificat ion pour informer les utilisate urs des modifications importantes apportées aux pipelines. Pour de plus amples informations, veuillez consulter Création d'une règle de notification.	5 novembre 2019

CodeBuild variables d'environnement disponibles dans CodePipeline	Vous pouvez définir des variables d' CodeBuild environnement dans l'action de CodeBuild création de votre pipeline. Vous pouvez utiliser la console ou l'interface de ligne de commande pour ajouter le paramètre `EnvironmentVariables` à la structure du pipeline. La rubrique Création d'un pipeline (console) a été mise à jour. Les exemples de configuration d'action dans la référence d'actions pour CodeBuild ont également été mis à jour.	14 octobre 2019
Nouvelle région	CodePipeline est désormais disponible en Europe (Stockholm). Les rubriques Limites et Service AWS points de terminaison ont été mises à jour.	5 septembre 2019

Spécifiez le contrôle ACLs prédéfini et le contrôle du cache pour les actions de déploiement d'Amazon S3	Vous pouvez désormais spécifier des options d'ACL prédéfinies et de contrôle du cache lorsque vous créez une action de déploiement Amazon S3 dans CodePipeline. Les rubriques suivantes ont été mises à jour : Création d'un pipeline (console), Référence de structure de CodePipeline pipeline et Tutoriel : Création d'un pipeline utilisant Amazon S3 comme fournisseur de déploiement.	27 juin 2019

Vous pouvez désormais ajouter des balises aux ressources dans AWS CodePipeline	Vous pouvez désormais utiliser le balisage pour suivre et gérer AWS CodePipeline des ressources telles que les pipelines, les actions personnalisées et les webhooks. Les nouvelles rubriques suivantes ont été ajoutées : Marquage des ressources, Utilisation de balises pour contrôler l'accès aux CodePipeline ressources, Marquer un pipeline CodePipeline, Marquer une action personnalisée dans CodePipeline et Marquer un webhook dans. CodePipeline Les rubriques suivantes ont été mises à jour pour montrer comment utiliser la CLI pour baliser les ressources : créer un pipeline (CLI), créer une action personnalisée (CLI) et créer un webhook pour une GitHub source.	15 mai 2019

Vous pouvez désormais consulter l'historique d'exécution des actions dans AWS CodePipeline	Vous pouvez désormais afficher les détails relatifs aux exécutions précédentes de toutes les actions dans un pipeline. Ces détails comprennent les heures de début et de fin, la durée, les ID d'exécution d'action, le statut, les détails relatifs à l'emplacement de l'artefact d'entrée et de sortie et les détails relatifs aux ressources externes. La rubrique Affichage des détails et de l'historique d'un pipeline a été mise à jour de façon à refléter cette prise en charge.	20 mars 2019
AWS CodePipeline prend désormais en charge la publication d'applications sur AWS Serverless Application Repository	Vous pouvez désormais créer un pipeline CodePipeline qui publie votre application sans serveur dans le AWS Serverless Application Repository. Un nouveau didacticiel, Tutoriel : publier des applications sur le AWS Serverless Application Repository, décrit les étapes à suivre pour créer et configurer un pipeline afin de fournir en continu votre application sans serveur au AWS Serverless Application Repository.	8 mars 2019

AWS CodePipeline prend désormais en charge les actions interrégionales dans la console	Vous pouvez désormais gérer les actions entre régions dans la AWS CodePipeline console. Ajouter une action interrégionale a été mise à jour avec les étapes permettant d'ajouter, de modifier ou de supprimer une action située dans une AWS région différent e de celle de votre pipeline. Les rubriques Création d'un pipeline, Modification d'un pipeline et Référence sur la structure du pipeline CodePipeline ont été mises à jour.	14 février 2019
AWS CodePipeline prend désormais en charge les déploiements Amazon S3	Vous pouvez désormais créer un pipeline CodePipel ine qui utilise Amazon S3 comme fournisseur d'actions de déploiement. Un nouveau didacticiel, Tutoriel : Création d'un pipeline utilisant Amazon S3 comme fournisseur de déploiement, décrit les étapes à suivre pour déployer des exemples de fichiers dans votre compartiment Amazon S3 avec CodePipel ine. La rubrique Référence sur la structure du pipeline CodePipeline a également été mise à jour.	16 janvier 2019

AWS CodePipeline prend désormais en charge les déploiements d'Alexa Skills Kit	Vous pouvez désormais utiliser un kit CodePipeline de compétences Alexa pour le déploiement continu des compétences Alexa. Un nouveau didacticiel, Tutoriel : Créez un pipeline qui déploie une compétence Amazon Alexa, contient les étapes à suivre pour créer des informations d'identification permettant de vous connecter AWS CodePipeline à votre compte de développeur Alexa Skills Kit, puis créer un pipeline qui déploie un exemple de compétence. La rubrique Référence sur la structure du pipeline CodePipeline a été mise à jour.	19 décembre 2018
AWS CodePipeline prend désormais en charge les points de terminaison Amazon VPC alimentés par AWS PrivateLink	Vous pouvez désormais vous connecter directement AWS CodePipeline via un point de terminaison privé dans votre VPC, en conservant tout le trafic à l'intérieur de votre VPC et du réseau. AWS Pour plus d'informations, consultez Utiliser CodePipeline avec Amazon Virtual Private Cloud.	6 décembre 2018

AWS CodePipeline prend désormais en charge les actions source et les actions de ECS-to-CodeDeploy déploiement Amazon ECR	Vous pouvez désormais utiliser CodePipeline et CodeDeploy avec Amazon ECR et Amazon ECS pour le déploiement continu d'applications basées sur des conteneurs. Un nouveau didacticiel, Create a pipeline with a Amazon ECR source and ECS-to-CodeDeploy deployment, décrit les étapes d'utilisation de la console pour créer un pipeline qui déploie des applications de conteneur stockées dans un référentiel d'images vers un cluster Amazon ECS avec routage CodeDeploy du trafic. Les rubriques Création d'un pipeline et Référence sur la structure du pipeline CodePipeline ont été mises à jour.	27 novembre 2018

AWS CodePipeline soutient désormais les actions interrégionales en cours	Une nouvelle rubrique, Ajouter une action interrégionale, décrit les étapes permettant d'utiliser AWS CLI ou d'CloudFormation ajouter une action située dans une région différente de celle de votre pipeline. Les rubriques Création d'un pipeline, Modification d'un pipeline et Référence sur la structure du pipeline CodePipeline ont été mises à jour.	12 novembre 2018
AWS CodePipeline s'intègre désormais à Service Catalog	Vous pouvez désormais ajouter Service Catalog en tant qu'action de déploiement à votre pipeline. Cela vous permet de configurer un pipeline pour publier les mises à jour des produits sur Service Catalog lorsque vous apportez une modification à votre référentiel source. La rubrique Intégrations a été mise à jour pour refléter cette prise en charge de Service Catalog. Deux didacticiels Service Catalog ont été ajoutés à la section des AWS CodePipeline didacticiels.	16 octobre 2018

AWS CodePipeline s'intègre désormais à AWS Device Farm	Vous pouvez désormais ajouter AWS Device Farm une action de test à votre pipeline. Cela vous permet de configurer un pipeline pour tester des applications mobiles. La rubrique Intégrations a été mise à jour pour refléter cette prise en charge de AWS Device Farm. Deux didacticiels AWS Device Farm ont été ajoutés dans la section Didacticiels AWS CodePipeline.	19 juillet 2018
AWS CodePipeline Les notifications de mise à jour du guide de l'utilisateur sont désormais disponibles via RSS	La version HTML du guide de l' CodePipeline utilisateur prend désormais en charge un flux RSS de mises à jour documentées sur la page Historique des mises à jour de la documentation. Le flux RSS inclut les mises à jour effectuées après le 30 juin 2018. Les mises à jour annoncées précédemment sont toujours disponibles sur la page Historique de mise à jour de la documentation. Utilisez le bouton RSS dans le panneau du menu supérieur pour vous abonner au flux.	30 juin 2018

Mises à jour antérieures

Le tableau suivant décrit les modifications importantes apportées à chaque version du guide de l' CodePipeline utilisateur publiée le 30 juin 2018 et les versions antérieures.

Modifier	Description	Date de modification
Utilisez des webhooks pour détecter les changements de source dans les pipelines GitHub	Lorsque vous créez ou modifiez un pipeline dans la console, il crée CodePipeline désormais un webhook qui détecte les modifications apportées à votre référentiel GitHub source, puis démarre votre pipeline. Pour plus d'informations sur la migration de votre pipeline, voir Configurer vos GitHub pipelines pour utiliser des webhooks pour la détection des modifications. Pour plus d'informa tions, consultez Lancement de l'exécution d'un pipeline dans CodePipeline.	1er mai 2018
Mise à jour des rubriques	Lorsque vous créez ou modifiez un pipeline dans la console, vous créez CodePipeline désormais une règle Amazon CloudWatch Events et un AWS CloudTrail journal qui détecte les modifications apportées à votre compartim ent source Amazon S3, puis démarre votre pipeline. Pour obtenir des informations sur la migration de votre pipeline, consultez Actions à la source et méthodes de détection des modifications. Didacticiel : Création d'un pipeline simple (compartiment S3)Il a été mis à jour pour montrer comment la règle et le suivi Amazon CloudWatch Events sont créés lorsque vous sélectionnez une source Amazon S3. Création d'un pipeline, d'étapes et d'actionset Modifier un pipeline dans CodePipeline ont également été mis à jour. Pour de plus amples informations, veuillez consulter Démarrez un pipeline dans CodePipeline.	22 mars 2018

Modifier	Description	Date de modification
Rubrique mise à jour	CodePipeline est désormais disponible en Europe (Paris). La rubrique Quotas dans AWS CodePipeline a été mise à jour.	21 février 2018
Mise à jour des rubriques	Vous pouvez désormais utiliser CodePipeline Amazon ECS pour le déploiement continu d'applications basées sur des conteneurs. Lorsque vous créez un pipeline, vous pouvez sélectionner Amazon ECS comme fournisseur de déploiement. Une modification du code dans votre référentiel de contrôle de source incite votre pipeline à créer une nouvelle image Docker, à la transférer dans votre registre de conteneurs, puis à déployer l'image mise à jour sur un service Amazon ECS. Les rubriquesIntégrations de produits et de services avec CodePipeline,Création d'un pipeline, d'étapes et d'actions , et CodePipeline référence de la structure du pipeline ont été mises à jour pour refléter cette prise en charge d'Amazon ECS.	12 décembre 2017

Modifier	Description	Date de modification
Mise à jour des rubriques	Lorsque vous créez ou modifiez un pipeline dans la console, vous créez CodePipeline désormais une règle Amazon CloudWatch Events qui détecte les modifications apportées à votre CodeCommit référentiel, puis démarre automatiquement votre pipeline. Pour obtenir des informations sur la migration de votre pipeline existant, consultez Actions à la source et méthodes de détection des modifications. Tutoriel : Création d'un pipeline simple (CodeCommit référentiel)Il a été mis à jour pour montrer comment la règle et le rôle Amazon CloudWatch Events sont créés lorsque vous sélectionnez un CodeCommit référentiel et une branche. Création d'un pipeline, d'étapes et d'actionset Modifier un pipeline dans CodePipeline ont également été mis à jour. Pour de plus amples informations, veuillez consulter Démarrez un pipeline dans CodePipeline.	11 octobre 2017
Rubriques nouvelles et mises à jour	CodePipeline fournit désormais un support intégré pour les notifications de modification de l'état du pipeline via Amazon CloudWatch Events et Amazon Simple Notification Service (Amazon SNS). Un nouveau didacticiel Tutoriel : Configuration d'une règle d' CloudWatch événements pour recevoir des notifications par e-mail en cas de modification de l'état du pipeline a été ajouté. Pour de plus amples informations, veuillez consulter Surveillance des CodePipeline événements.	8 septembre 2017

Modifier	Description	Date de modification
Rubriques nouvelles et mises à jour	Vous pouvez désormais l'ajouter CodePipeline en tant que cible pour les actions Amazon CloudWatch Events. Les règles Amazon CloudWatch Events peuvent être configuré es pour détecter les modifications de source afin que le pipeline démarre dès que ces modifications se produisen t, ou elles peuvent être configurées pour exécuter des exécutions de pipeline planifiées. Des informations ont été ajoutées pour l'option de configuration de l'action PollForSo urceChanges source. Pour de plus amples informations, veuillez consulter Démarrez un pipeline dans CodePipel ine.	5 septembre 2017
Nouvelles régions	CodePipeline est désormais disponible en Asie-Pacifique (Séoul) et en Asie-Pacifique (Mumbai). Les rubriques Quotas dans AWS CodePipeline et Régions et points de terminaison ont été mises à jour.	27 juillet 2017
Nouvelles régions	CodePipeline est désormais disponible dans l'ouest des États-Unis (Californie du Nord), au Canada (centre) et en Europe (Londres). Les rubriques Quotas dans AWS CodePipeline et Régions et points de terminaison ont été mises à jour.	29 juin 2017
Mise à jour des rubriques	Vous pouvez désormais afficher les détails sur les précédentes exécutions d'un pipeline, et plus seulement sur la dernière exécution. Ces détails comprennent les heures de début et de fin, la durée et l'ID d'exécution. Des informations sont disponibles pour un maximum de 100 exécutions de pipeline au cours des douze derniers mois. Les rubriques Afficher les pipelines et les détails dans CodePipeline, référence des autorisations et Quotas dans AWS CodePipeline ont été mises à jour pour refléter cette prise en charge.	22 juin 2017

Modifier	Description	Date de modification
Rubrique mise à jour	Nouvola a été ajouté à la liste des actions disponibles dans Intégrations d'actions de test.	18 mai 2017
Mise à jour des rubriques	Dans l' AWS CodePipeline assistant, la page Étape 4 : Bêta a été renommée Étape 4 : Déploiement. Le nom par défaut « Bêta » de l'étape créée par cette étape a été modifié par « Intermédiaire ». De nombreuses rubriques et captures d'écran ont été mises à jour afin de refléter ces modifications.	7 avril 2017
Mise à jour des rubriques	Vous pouvez désormais l'ajouter AWS CodeBuild en tant qu'action de test à n'importe quelle étape d'un pipeline. Cela vous permet d'exécuter plus facilement AWS CodeBuild des tests unitaires sur votre code. Avant cette version, vous pouviez AWS CodeBuild exécuter des tests unitaires uniquement dans le cadre d'une action de compilation. Une action de génération implique un artefact de sortie de la génération, que les tests unitaires ne produisent généralement pas. Les rubriques Intégrations de produits et de services avec CodePipelineModifier un pipeline dans CodePipeline, et CodePipeline référence de la structure du pipeline ont été mises à jour pour refléter cette prise en charge de AWS CodeBuild.	8 mars 2017

Modifier	Description	Date de modification
Rubriques nouvelles et mises à jour	La table des matières a été réorganisée pour inclure des sections pour les pipelines, les actions et les transitions entre les étapes. Une nouvelle section a été ajoutée pour les CodePipeline didacticiels. Pour plus de commodité, Intégrations de produits et de services avec CodePipeline a été divisé en rubriques plus courtes. Une nouvelle section, Autorisation et contrôle d'accès, fournit des informations complètes sur l'utilisation Gestion des identités et des accès AWS (IAM) et CodePipeline permet de sécuriser l'accès à vos ressources grâce à l'utilisation d'informations d'identification. Ces informations d'identification fournissent les autorisations requises pour accéder aux AWS ressources, telles que le placement et la récupération d'artefacts dans les compartiments Amazon S3 et l'intégration de OpsWorks piles dans vos pipelines.	8 février 2017
Nouvelle région	CodePipeline est désormais disponible en Asie-Pacifique (Tokyo). Les rubriques Quotas dans AWS CodePipeline et Régions et points de terminaison ont été mises à jour.	14 décembre 2016
Nouvelle région	CodePipeline est désormais disponible en Amérique du Sud (São Paulo). Les rubriques Quotas dans AWS CodePipeline et Régions et points de terminaison ont été mises à jour.	7 décembre 2016

Modifier	Description	Date de modification
Mise à jour des rubriques	Vous pouvez désormais ajouter AWS CodeBuild une action de construction à n'importe quelle étape d'un pipeline. AWS CodeBuild est un service de génération entièrement géré dans le cloud qui compile votre code source, exécute des tests unitaires et produit des artefacts prêts à être déployés. Vous pouvez utiliser un projet de construction existant ou en créer un dans la CodePipeline console. Les données de sortie du projet de génération peuvent ensuite être déployées dans le cadre d'un pipeline. Les rubriques Intégrations de produits et de services avec CodePipeline Authentification et contrôle d'accès CodePipeline référence de la structure du pipeline ont été mises à jour pour refléter cette prise en charge de AWS CodeBuild. Création d'un pipeline, d'étapes et d'actions Vous pouvez désormais utiliser le modèle CodePipeline d'application AWS sans serveur CloudFormation et le modèle d'application sans serveur pour fournir en continu vos applications sans serveur. La rubrique Intégrations de produits et de services avec CodePipeline a été mise à jour pour refléter cette prise en charge. Intégrations de produits et de services avec CodePipelinea été réorganisé pour regrouper les offres AWS et celles des partenaires par type d'action.	1er décembre 2016
Nouvelle région	CodePipeline est désormais disponible en Europe (Francfort). Les rubriques Quotas dans AWS CodePipeline et Régions et points de terminaison ont été mises à jour.	16 novembre 2016

Modifier	Description	Date de modification
Mise à jour des rubriques	AWS CloudFormation peut désormais être sélectionné en tant que fournisseur de déploiement dans les pipelines, ce qui vous permet d'agir sur les CloudFormation piles et de modifier des ensembles dans le cadre de l'exécution d'un pipeline. Les rubriques Intégrations de produits et de services avec CodePipeline Authentification et contrôle d'accès CodePipeline référence de la structure du pipeline ont été mises à jour pour refléter cette prise en charge de AWS CloudFormation. Création d'un pipeline, d'étapes et d'actions	3 novembre 2016
Nouvelle région	CodePipeline est désormais disponible dans la région Asie-Pacifique (Sydney). Les rubriques Quotas dans AWS CodePipeline et Régions et points de terminaison ont été mises à jour.	26 octobre 2016
Nouvelle région	CodePipeline est désormais disponible en Asie-Pacifique (Singapour). Les rubriques Quotas dans AWS CodePipeline et Régions et points de terminaison ont été mises à jour.	le 20 octobre 2016
Nouvelle région	CodePipeline est désormais disponible dans la région USA Est (Ohio). Les rubriques Quotas dans AWS CodePipeline et Régions et points de terminaison ont été mises à jour.	17 octobre 2016
Rubrique mise à jour	Création d'un pipeline, d'étapes et d'actions a été mis à jour pour refléter la prise en charge de l'affichage des identificateurs de version d'actions personnalisées dans les listes Source provider et Build provider.	22 septembre 2016
Rubrique mise à jour	La section Ajouter une action d'approbation manuelle à une étape a été mise à jour pour refléter une amélioration, qui permet aux vérificateurs des actions d'approbation d'ouvrir le formulaire Approve or reject the revision directement à partir d'une notification par e-mail.	14 septembre 2016

Modifier	Description	Date de modification
Rubriques nouvelles et mises à jour	Une nouvelle rubrique qui décrit comment afficher les détails des modifications de code en cours dans votre pipeline de publication logicielle. Un accès rapide à ces informations peut être utile lors de la vérification des actions d'approbation manuelle ou de la résolution des problèmes dans votre pipeline. Une nouvelle section, Surveillance des pipelines, fournit un emplacement central pour toutes les rubriques liées à la surveillance de l'état et à la progression de vos pipelines.	08 septembre 2016
Rubriques nouvelles et mises à jour	Une nouvelle section, Ajouter une action d'approbation manuelle à une étape, fournit des informations sur la configuration et l'utilisation d'actions d'approbation manuelle dans les pipelines. Les rubriques de cette section fournissent des informations conceptuelles sur le processus d'approbation, des instructions pour configurer les autorisations IAM requises, créer des actions d'approbation et approuver ou rejeter des actions d'approbation, ainsi que des exemples de données JSON générées lorsqu'une action d'approbation est atteinte dans un pipeline.	06 juillet 2016
Nouvelle région	CodePipeline est désormais disponible dans la région Europe (Irlande). Les rubriques Quotas dans AWS CodePipeline et Régions et points de terminaison ont été mises à jour.	23 juin 2016
Nouvelle rubrique	Une nouvelle rubrique, , a été ajoutée pour expliquer comment retenter une action ayant échoué ou un groupe d'actions parallèles ayant échoué dans l'étape.	22 juin 2016

Modifier	Description	Date de modification
Mise à jour des rubriques	Un certain nombre de rubriquesCréation d'un pipeline, d'étapes et d'actions, notamment l'authentification et le contrôle d'accès CodePipeline référence de la structure du pipelineIntégrations de produits et de services avec CodePipeline, ont été mises à jour pour refléter la prise en charge de la configuration d'un pipeline pour déployer du code en conjonction avec des livres de recettes Chef personnalisés et des applications créées dans AWS OpsWorks. CodePipeline le support pour OpsWorks n'est actuellement disponible que dans la région USA Est (Virginie du Nord) (us-east-1).	2 juin 2016
Rubriques nouvelles et mises à jour	Une nouvelle rubrique, Tutoriel : Création d'un pipeline simple (CodeCommit référentiel), a été ajoutée. Cette rubrique fournit un exemple de procédure pas à pas expliquant comment utiliser un CodeCommit référentiel et une branche comme emplacement source pour une action source dans un pipeline. Plusieurs autres rubriques ont été mises à jour pour refléter cette intégration CodeCommit, notamment l'authentification et le contrôle d'accèsIntégrations de produits et de services avec CodePipeline,Didacticiel : Création d'un pipeline à quatre étapes, etRésolution des problèmes CodePipeline.	18 avril 2016
Nouvelle rubrique	Une nouvelle rubrique, Invoquer un AWS Lambda fonction dans un pipeline dans CodePipeline, a été ajoutée. Cette rubrique contient des exemples de AWS Lambda fonctions et des étapes pour ajouter des fonctions Lambda aux pipelines.	27 janvier 2016
Rubrique mise à jour	Une nouvelle section, Stratégies basées sur les ressources, a été ajoutée à la rubrique Authentification et contrôle d'accès.	22 janvier 2016

Modifier	Description	Date de modification
Nouvelle rubrique	Une nouvelle rubrique, Intégrations de produits et de services avec CodePipeline, a été ajoutée. Les informations sur les intégrations avec des partenaires et d'autres Services AWS entités ont été déplacées vers cette rubrique. Des liens vers des blogs et des vidéos ont également été ajoutés.	17 décembre 2015
Rubrique mise à jour	Des détails relatifs à l'intégration avec Solano CI ont été ajoutés à Intégrations de produits et de services avec CodePipeline.	17 novembre 2015
Rubrique mise à jour	Le CodePipeline plugin pour Jenkins est désormais disponible via le gestionnaire de plugins Jenkins dans le cadre de la bibliothèque de plugins pour Jenkins. La procédure d'installation du plug-in a été mise à jour dans Didacticiel : Création d'un pipeline à quatre étapes.	9 novembre 2015
Nouvelle région	CodePipeline est désormais disponible dans la région USA Ouest (Oregon). La rubrique Quotas dans AWS CodePipeline a été mise à jour. Des liens ont été ajoutés dans Régions et points de terminaison.	22 octobre 2015
Nouvelle rubrique	Deux nouvelles rubriques,Configurer le chiffrement côté serveur pour les artefacts stockés dans Amazon S3 pour CodePipeline et Créez un pipeline CodePipeline qui utilise les ressources d'un autre AWS compte, ont été ajoutées. Une nouvelle section a été ajoutée à Authentification et contrôle d'accès.	25 août 2015
Rubrique mise à jour	La rubrique Créez et ajoutez une action personnalisée dans CodePipeline a été mise à jour pour refléter les modifications de la structure, y compris `inputArtifactDetails` et `outputArtifactDetails` .	17 août 2015

Modifier	Description	Date de modification
Rubrique mise à jour	La Résolution des problèmes CodePipeline rubrique a été mise à jour avec des étapes révisées pour résoudre les problèmes liés au rôle de service et à Elastic Beanstalk.	11 août 2015
Rubrique mise à jour	La rubrique Authentification et contrôle d'accès a été mise à jour pour inclure les dernières modifications apportées au rôle de service pour CodePipeline.	6 août 2015
Nouvelle rubrique	Une rubrique Résolution des problèmes CodePipeline a été ajoutée. Des étapes mises à jour ont été ajoutées pour les rôles IAM et Jenkins dans. Didacticiel : Création d'un pipeline à quatre étapes	24 juillet 2015
Mise à jour de rubrique	De nouvelles étapes ont été ajoutées pour télécharger les modèles de fichier dans Didacticiel : Création d'un pipeline simple (compartiment S3) et Didacticiel : Création d'un pipeline à quatre étapes.	22 juillet 2015
Mise à jour de rubrique	Une solution temporaire pour les problèmes de télécharg ement des modèles de fichier a été ajoutée dans Didactici el : Création d'un pipeline simple (compartiment S3).	17 juillet 2015
Mise à jour de rubrique	Un lien a été ajouté dans Quotas dans AWS CodePipeline pour rediriger vers les informations indiquant les limites qui peuvent être modifiées.	15 juillet 2015
Mise à jour de rubrique	La section Stratégies gérées de la rubrique Authentification et contrôle d'accès a été mise à jour.	10 juillet 2015
Première version publique	Il s'agit de la première version publique du guide de CodePipeline l'utilisateur.	9 juillet 2015

CodePipeline référence de fonctionnalité

Cette section est une référence de haut niveau pour les fonctionnalités de CodePipeline la documentation. Pour obtenir une présentation conceptuelle de la structure du pipeline, veuillez consulter CodePipeline référence de la structure du pipeline.

Ce tableau est régulièrement mis à jour avec les informations de référence des fonctionnalités.

Date de mise à jour du tableau	Fonctionnalité	CodePipeline Guide de l'utilisateur	CodePipeline Guide de l'API	CloudFormation Référence	AWS CDK Référence - Constructions L1
15 octobre 202	Réessai automatique en cas d'échec de l'étape	Configurer une étape pour une nouvelle tentative automatique en cas d'échec	RetryConfiguration		
15 octobre 202	Ignorer le résultat pour les conditions BeforeEntry	Création de conditions de saisie avec Ignorer le résultat et la `VariableCheck` règle (console)	FailureConditions		
3 octobre 2024	Action des commandes avec la nouvelle catégorie de calcul	Référence des actions des commandes	ActionDeclaration		

Date de mise à jour du tableau	Fonctionnalité	CodePipeline Guide de l'utilisateur	CodePipeline Guide de l'API	CloudFormation Référence	AWS CDK Référence - Constructions L1
28 août 2024	Type BeforeEntry pour les conditions de scène	Configuration des conditions pour une étape	BeforeEntryConditions	AWS::CodePipeline::Pipeline::BeforeEntryConditions	Avant l'entrée
28 août 2024	Type OnSuccess pour les conditions de scène	Configuration des conditions pour une étape	SuccessConditions	AWS::CodePipeline::Pipeline::SuccessConditions	onSuccess
26 avril 2024	Annulations de scène et type OnFailure pour les conditions de scène	Configuration de la restauration des étapes	RollbackStage	onFailure	onFailure
08 février 2024	Modes d'exécution PARALLEL et QUEUED	Définir ou modifier le mode d'exécution du pipeline	PipelineExecution	ExecutionMode	Mode d'exécution

Date de mise à jour du tableau	Fonctionnalité	CodePipeline Guide de l'utilisateur	CodePipeline Guide de l'API	CloudFormation Référence	AWS CDK Référence - Constructions L1
17 novembre 2(	Dérogations de source	Démarrer un pipeline avec un remplacement de révision de la source	SourceRevisionOverride	N/A	N/A
24 octobre 202	Types de pipelines	Quel type de pipeline me convient le mieux ?	PipelineDeclaration	PipelineType	énumération PipelineType
24 octobre 202	Variables au niveau du pipeline	Tutoriel : Utiliser des variables au niveau du pipeline	PipelineVariable	AWS::CodePipeline::Pipeline::PipelineVariableDeclaration	Variables au niveau du pipeline

Date de mise à jour du tableau	Fonctionnalité	CodePipeline Guide de l'utilisateur	CodePipeline Guide de l'API	CloudFormation Référence	AWS CDK Référence - Constructions L1
24 octobre 202	Déclencheurs et filtrage des chemins de fichiers/branches/pull requests	Automatisez le démarrage des pipelines en utilisant des déclencheurs et des filtres Tutoriel : Filtrer les noms de branches pour les pull requests afin de démarrer votre pipeline (type V2)	ActionDeclaration	AWS::CodePipeline::Pipeline PipelineTriggerDeclaration	Déclencheur
17 octobre 202	Réessayer une étape	Configuration d'une nouvelle tentative d'étape pour une étape ayant échoué ou des actions ayant échoué	ActionDeclaration	N/A	N/A

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.