Panduan Pengguna

AWS CodePipeline

Versi API 2015-07-09

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

AWS CodePipeline: Panduan Pengguna

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu CodePipeline? 1
Pengiriman berkelanjutan dan integrasi berkelanjutan 1
Apa yang bisa saya lakukan dengan CodePipeline? 2
Sekilas melihat CodePipeline 3
Bagaimana saya memulai CodePipeline? 4
Konsep 4
Alur 5
Eksekusi pipa 6
Operasi panggung 8
Eksekusi aksi 8
Jenis eksekusi 9
Jenis tindakan 9
Artifacts 9
Revisi sumber 10
Pemicu 10
Variabel 11
Ketentuan 11
Aturan 11
DevOps contoh pipa 12
Bagaimana eksekusi pipa bekerja 14
Bagaimana eksekusi pipa dimulai 15
Bagaimana revisi sumber diproses dalam eksekusi pipa 15
Bagaimana penggantian sumber bekerja dengan transformator input EventBridge 16
Bagaimana eksekusi pipa dihentikan 17
Bagaimana eksekusi diproses dalam mode SUPERSEDED 21
Bagaimana eksekusi diproses dalam mode ANTRIAN 22
Bagaimana eksekusi diproses dalam mode PARALLEL 24
Mengelola Aliran Pipa 24
Artefak input dan output 27
Bagaimana kondisi panggung bekerja? 30
Aturan untuk kondisi panggung 33
Jenis pipa 34
Jenis pipa apa yang tepat untuk saya? 34
Memulai 40

Langkah 1: Buat pengguna Akun AWS dan administratif 40
Mendaftar untuk Akun AWS 40
Buat pengguna dengan akses administratif 41
Langkah 2: Menerapkan kebijakan terkelola untuk akses administratif CodePipeline 42
Langkah 3: Instal AWS CLI 44
Langkah 4: Buka konsol untuk CodePipeline 45
Langkah selanjutnya 45
Integrasi produk dan layanan 46
Integrasi dengan tipe CodePipeline tindakan 46
Integrasi tindakan sumber 46
Bangun integrasi tindakan 54
Integrasi tindakan uji 56
Menyebarkan integrasi tindakan 58
Integrasi tindakan persetujuan dengan Amazon Simple Notification Service 65
Memanggil integrasi tindakan 65
Integrasi umum dengan CodePipeline 67
Contoh dari komunitas 70
Unggahan blog 70
Tutorial 74
Tutorial: Terapkan ke instans Amazon EC2 dengan CodePipeline 75
Prasyarat 75
Langkah 1: Buat instans Amazon EC2 Linux 76
Langkah 2: Tambahkan izin bucket artefak ke peran instans EC2 77
Langkah 3: Tambahkan file skrip ke repositori Anda 78
Langkah 4: Membuat pipeline Anda 79
Langkah 5: Uji Pipa Anda 83
Tutorial: Buat dan dorong gambar Docker ke Amazon ECR dengan CodePipeline (tipe V2) 84
Prasyarat 85
Langkah 1: Tambahkan Dockerfile ke repositori sumber Anda 86
Langkah 2: Tambahkan file imagedefinitions.json ke repositori sumber Anda 87
Langkah 3: Membuat pipeline Anda 88
Langkah 4: Uji Pipa Anda 93
Tutorial: Terapkan ke Amazon EKS dengan CodePipeline 94
Prasyarat 94
Langkah 1: (Opsional) Buat cluster di Amazon EKS 95
Langkah 2: Konfigurasikan klaster pribadi Anda di Amazon EKS 97

Langkah 3: Perbarui kebijakan peran CodePipeline layanan di IAM 98
Langkah 4: Buat entri akses untuk peran CodePipeline layanan 101
Langkah 5: Buat repositori sumber dan tambahkan file konfigurasi `helm chart` 102
Langkah 6: Membuat pipeline Anda 102
Tutorial: Buat pipeline yang menjalankan perintah dengan komputasi (tipe V2) 104
Prasyarat 105
Langkah 1: Buat file sumber dan dorong ke GitHub repositori Anda 105
Langkah 2: Buat alur Anda 106
Langkah 3: Jalankan pipeline Anda dan verifikasi perintah build 109
Tutorial: Gunakan tag Git untuk memulai pipeline Anda 111
Prasyarat 112
Langkah 1: Buka CloudShell dan kloning repositori Anda 112
Langkah 2: Buat pipeline untuk memicu tag Git 112
Langkah 3: Tandai komit Anda untuk rilis 116
Langkah 4: Lepaskan perubahan dan lihat log 117
Tutorial: Filter nama cabang untuk permintaan tarik untuk memulai pipeline Anda (tipe V2) 118
Prasyarat 118
Langkah 1: Buat pipeline untuk memulai permintaan tarik untuk cabang tertentu 119
Langkah 2: Buat dan gabungkan permintaan tarik GitHub di.com untuk memulai eksekusi pipeline Anda 121
Tutorial: Gunakan variabel tingkat pipa 123
Prasyarat 123
Langkah 1: Buat pipeline Anda dan bangun proyek 123
Langkah 2: Lepaskan perubahan dan lihat log 127
Tutorial: Buat pipeline sederhana (ember S3) 127
Buat Bucket S3 129
Buat instans Windows Server Amazon EC2 dan instal agen CodeDeploy 131
Buat aplikasi di CodeDeploy 133
Buat pipeline pertama Anda 134
Tambahkan tahap lain 137
Nonaktifkan dan aktifkan transisi antar tahapan 144
Pembersihan sumber daya 145
Tutorial: Buat pipeline sederhana (CodeCommit repositori) 146
Buat CodeCommit repositori 147
Unduh, komit, dan dorong kode Anda 148
Buat instans Amazon EC2 Linux dan instal agen CodeDeploy 150

Buat aplikasi di CodeDeploy 152
Buat pipeline pertama Anda 154
Perbarui kode di CodeCommit repositori Anda 157
Pembersihan sumber daya 159
Sumber bacaan lebih lanjut 159
Tutorial: Buat pipeline empat tahap 160
Prasyarat lengkap 161
Buat pipeline 166
Tambahkan lebih banyak tahapan 167
Pembersihan sumber daya 171
Tutorial: Menyiapkan aturan CloudWatch Acara untuk menerima pemberitahuan email untuk perubahan status pipeline 172
Mengatur notifikasi email menggunakan Amazon SNS 172
Membuat aturan pemberitahuan CloudWatch Acara untuk CodePipeline 174
Pembersihan sumber daya 175
Tutorial: Membangun dan menguji aplikasi Android dengan AWS Device Farm 175
Konfigurasikan CodePipeline untuk menggunakan pengujian Device Farm 177
Tutorial: Uji aplikasi iOS dengan AWS Device Farm 181
Konfigurasikan CodePipeline untuk menggunakan pengujian Device Farm Anda (contoh Amazon S3) 183
Tutorial: Buat pipeline yang di-deploy ke Service Catalog 187
Opsi 1: Terapkan ke Service Catalog tanpa file konfigurasi 189
Opsi 2: Terapkan ke Service Catalog menggunakan file konfigurasi 194
Tutorial: Buat pipeline dengan AWS CloudFormation 199
Contoh 1: Buat AWS CodeCommit pipeline dengan AWS CloudFormation 199
Contoh 2: Buat pipeline Amazon S3 dengan AWS CloudFormation 201
Tutorial: Buat pipeline yang menggunakan variabel dari AWS CloudFormation tindakan penerapan 205
Prasyarat: Buat peran AWS CloudFormation layanan dan repositori CodeCommit 206
Langkah 1: Unduh, edit, dan unggah AWS CloudFormation templat sampel 207
Langkah 2: Buat alur Anda 208
Langkah 3: Tambahkan tindakan CloudFormation penerapan untuk membuat set perubahan 210
Langkah 4: Tambahkan tindakan persetujuan manual 211
Langkah 5: Tambahkan tindakan CloudFormation penerapan untuk menjalankan set perubahan 212

Langkah 6: Tambahkan tindakan CloudFormation penerapan untuk menghapus tumpukan 213
Tutorial: Penerapan Standar Amazon ECS dengan CodePipeline 213
Prasyarat 214
Langkah 1: Tambahkan File Spesifikasi Build ke Repositori Sumber Anda 217
Langkah 2: Membuat Pipeline Deployment Berkelanjutan 219
Langkah 3: Tambahkan Izin Amazon ECR ke Peran CodeBuild 221
Langkah 4: Uji Pipa Anda 221
Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR 222
Prasyarat 224
Langkah 1: Buat gambar dan dorong ke repositori Amazon ECR 224
Langkah 2: Buat definisi tugas dan file AppSpec sumber dan dorong ke CodeCommit repositori 226
Langkah 3: Buat Application Load Balancer dan grup target 230
Langkah 4: Buat kluster dan layanan Amazon ECS Anda 232
Langkah 5: Buat grup CodeDeploy aplikasi dan penyebaran Anda (platform komputasi ECS) 235
Langkah 6: Buat pipeline Anda 236
Langkah 7: Buat perubahan pada pipeline Anda dan verifikasi penerapan 240
Tutorial: Buat pipeline yang menerapkan keterampilan Amazon Alexa 240
Prasyarat 241
Langkah 1: Buat profil keamanan LWA layanan pengembang Alexa 241
Langkah 2: Buat file sumber keterampilan Alexa dan dorong ke repositori Anda CodeCommit 242
Langkah 3: Gunakan perintah ASK CLI untuk membuat token penyegaran 244
Langkah 4: Buat pipeline Anda 244
Langkah 5: Buat perubahan pada file sumber apa pun dan verifikasi penyebaran 246
Tutorial: Membuat pipeline yang menggunakan Amazon S3 sebagai penyedia penerapan 247
Opsi 1: Menyebarkan file situs web statis ke Amazon S3 248
Opsi 2: Menerapkan file arsip bawaan ke Amazon S3 dari bucket sumber S3 253
Tutorial: Publikasikan aplikasi ke AWS Serverless Application Repository 258
Sebelum kamu memulai 259
Langkah 1: Buat file buildspec.yml. 260
Langkah 2: Buat dan konfigurasikan pipeline Anda 260
Langkah 3: Menyebarkan aplikasi publikasi 262

Langkah 4: Buat tindakan publikasi 263
Tutorial: Penerapan fungsi Lambda dengan CodePipeline 264
Prasyarat 264
Langkah 1: Buat fungsi Lambda sampel Anda 265
Langkah 2: Unggah file fungsi ke repositori Anda 265
Langkah 3: Membuat pipeline Anda 266
Langkah 4: Uji Pipa Anda 269
Pelajari selengkapnya 270
Tutorial: Menggunakan variabel dengan tindakan panggilan Lambda 270
Prasyarat 271
Langkah 1: Membuat fungsi Lambda 271
Langkah 2: Tambahkan tindakan pemanggilan Lambda dan tindakan persetujuan manual ke pipeline Anda 274
Tutorial: Gunakan tindakan AWS Step Functions pemanggilan 275
Prasyarat: Buat atau pilih pipa sederhana 276
Langkah 1: Buat mesin status sampel 277
Langkah 2: Tambahkan tindakan pemanggilan Step Functions ke pipeline Anda 277
Tutorial: Buat pipeline yang digunakan AppConfig sebagai penyedia penyebaran 278
Prasyarat 279
Langkah 1: Buat AWS AppConfig sumber daya Anda 279
Langkah 2: Unggah file ke bucket sumber S3 Anda 280
Langkah 3: Buat pipeline Anda 281
Langkah 4: Buat perubahan pada file sumber apa pun dan verifikasi penyebaran 282
Tutorial: Gunakan klon lengkap dengan sumber GitHub pipa 282
Prasyarat 284
Langkah 1: Buat file README 284
Langkah 2: Buat pipeline Anda dan bangun proyek 284
Langkah 3: Perbarui kebijakan peran CodeBuild layanan untuk menggunakan koneksi 288
Langkah 4: Lihat perintah repositori dalam output build 288
Tutorial: Gunakan klon lengkap dengan sumber CodeCommit pipa 289
Prasyarat 290
Langkah 1: Buat file README 290
Langkah 2: Buat pipeline Anda dan bangun proyek 290
Langkah 3: Perbarui kebijakan peran CodeBuild layanan untuk mengkloning repositori 293
Langkah 4: Lihat perintah repositori dalam output build 293
Tutorial: Buat pipeline dengan AWS CloudFormation StackSets tindakan penerapan 294

Prasyarat 295
Langkah 1: Unggah AWS CloudFormation template sampel dan file parameter 295
Langkah 2: Buat alur Anda 297
Langkah 3: Lihat penyebaran awal 300
Langkah 4: Tambahkan CloudFormationStackInstances tindakan 300
Langkah 5: Lihat sumber daya set tumpukan untuk penerapan Anda 301
Langkah 6: Buat pembaruan ke set tumpukan Anda 302
Buat aturan pemeriksaan variabel untuk pipeline sebagai kondisi entri 302
Prasyarat 303
Langkah 1: Buat file sumber sampel dan tambahkan ke GitHub repositori Anda 304
Langkah 2: Buat alur Anda 304
Langkah 2: Edit tahap build untuk menambahkan kondisi dan aturan 307
Langkah 3: Jalankan pipeline dan lihat variabel yang diselesaikan 309
Praktik terbaik dan kasus penggunaan 312
Contoh cara menggunakan CodePipeline 312
Gunakan CodePipeline dengan Amazon S3,, dan AWS CodeCommit AWS CodeDeploy 312
Gunakan CodePipeline dengan penyedia tindakan pihak ketiga (GitHub dan Jenkins) 313
Gunakan CodePipeline untuk mengkompilasi, membangun, dan menguji kode dengan CodeBuild 314
Gunakan CodePipeline dengan Amazon ECS untuk pengiriman berkelanjutan aplikasi berbasis kontainer ke cloud 314
Gunakan CodePipeline dengan Elastic Beanstalk untuk pengiriman berkelanjutan aplikasi web ke cloud 314
Gunakan CodePipeline dengan AWS Lambda untuk pengiriman berkelanjutan aplikasi berbasis Lambda dan tanpa server 314
Gunakan CodePipeline dengan CloudFormation template untuk pengiriman berkelanjutan ke cloud 315
Gunakan CodePipeline dengan Amazon VPC 316
Ketersediaan 316
Buat titik akhir VPC untuk CodePipeline 317
Memecahkan masalah penyiapan VPC 318
Tentukan CI/CD saluran pipa dengan tahapan dan tindakan 319
Buat pipeline, tahapan, dan tindakan 320
Buat pipeline khusus (konsol) 321
Buat pipeline (CLI) 334
Buat pipeline dari template statis 340

Mengedit pipa 346
Mengedit pipeline (konsol) 347
Mengedit pipeline (AWS CLI) 350
Lihat saluran pipa dan detailnya 355
Lihat saluran pipa (konsol) 355
Melihat detail tindakan dalam pipeline (konsol) 360
Lihat ARN pipeline dan peran layanan ARN (konsol) 363
Lihat detail dan riwayat saluran pipa (CLI) 364
Lihat hasil aturan untuk kondisi tahap dalam riwayat eksekusi 365
Hapus pipa 368
Hapus pipeline (konsol) 368
Hapus pipa (CLI) 368
Buat pipeline yang menggunakan sumber daya dari akun lain 370
Prasyarat: Buat kunci enkripsi AWS KMS 372
Langkah 1: Siapkan kebijakan dan peran akun 373
Langkah 2: Edit pipa 379
Migrasi jaringan pemungutan suara untuk menggunakan deteksi perubahan berbasis peristiwa 383
Cara memigrasi jaringan pipa pemungutan suara 383
Melihat saluran pemungutan suara di akun Anda 385
Migrasi jaringan pemungutan suara dengan sumber CodeCommit 390
Migrasikan jalur pemungutan suara dengan sumber S3 yang diaktifkan untuk acara 413
Migrasi jaringan pemungutan suara dengan sumber dan jejak S3 CloudTrail 432
Migrasikan pipeline polling untuk tindakan sumber GitHub (melalui OAuth aplikasi) ke koneksi 468
Migrasikan jalur polling untuk tindakan sumber GitHub (melalui OAuth aplikasi) ke webhooks 472
Buat peran CodePipeline layanan 489
Buat peran CodePipeline layanan (konsol) 490
Buat peran CodePipeline layanan (CLI) 490
Penandaan pada sumber daya 493
Menandai pipa 494
Menandai saluran pipa (konsol) 494
Pipa Tag (CLI) 496
Membuat aturan notifikasi 498
..... 502

Connect ke penyedia sumber pihak pertama menggunakan tindakan sumber 505
Tindakan sumber Amazon ECR dan EventBridge 505
Buat EventBridge aturan untuk sumber Amazon ECR (konsol) 506
Buat EventBridge aturan untuk sumber Amazon ECR (CLI) 508
Buat EventBridge aturan untuk sumber Amazon ECR (CloudFormation template) 512
Menghubungkan ke tindakan sumber Amazon S3 yang menggunakan peristiwa EventBridge .. 517
Buat pipeline dengan sumber S3 yang diaktifkan untuk acara (CLI) 518
Buat pipeline dengan sumber S3 yang diaktifkan untuk acara (template)CloudFormation 522
Menghubungkan ke tindakan sumber Amazon S3 yang menggunakan dan EventBridge AWS CloudTrail 537
Buat EventBridge aturan untuk sumber Amazon S3 (konsol) 538
Buat EventBridge aturan untuk sumber Amazon S3 (CLI) 542
Buat EventBridge aturan untuk sumber Amazon S3 (template)CloudFormation 548
CodeCommit tindakan sumber dan EventBridge 561
Buat EventBridge aturan untuk CodeCommit sumber (konsol) 562
Buat EventBridge aturan untuk CodeCommit sumber (CLI) 565
Buat EventBridge aturan untuk CodeCommit sumber (CloudFormation template) 570
Tambahkan penyedia sumber pihak ketiga ke saluran pipa menggunakan CodeConnections 578
Koneksi Azure DevOps 578
Buat koneksi ke Azure DevOps (konsol) 580
Buat koneksi ke Azure DevOps (CLI) 584
Koneksi Bitbucket Cloud 586
Buat koneksi ke Bitbucket Cloud (konsol) 588
Buat koneksi ke Bitbucket Cloud (CLI) 591
GitHub koneksi 593
Buat koneksi ke GitHub (konsol) 594
Buat koneksi ke GitHub (CLI) 598
GitHub Koneksi Enterprise Server 600
Buat koneksi ke GitHub Enterprise Server (konsol) 601
Buat host dan koneksi ke GitHub Enterprise Server (CLI) 605
GitLabkoneksi .com 608
Buat koneksi GitLab ke.com (konsol) 610
Buat koneksi GitLab ke.com (CLI) 614
Koneksi untuk dikelola GitLab sendiri 616
Buat koneksi ke GitLab kelola sendiri (konsol) 618
Buat host dan koneksi ke GitLab self-managed (CLI) 621

Menggunakan koneksi yang dibagikan dengan akun lain 624
Otomatiskan saluran pipa awal menggunakan pemicu dan penyaringan 625
Pertimbangan untuk filter pemicu 627
Peristiwa permintaan tarik untuk pemicu oleh penyedia 628
Contoh untuk filter pemicu 629
Tambahkan pemicu pada push kode tanpa filter 637
Tambahkan pemicu dengan tipe acara permintaan push atau pull kode 638
Tambahkan filter untuk jenis acara permintaan push dan pull (konsol) 639
Tambahkan filter untuk jenis acara permintaan push dan pull (CLI) 642
Tambahkan filter untuk jenis acara permintaan push dan pull (CloudFormation template) 644
Tambahkan pemicu untuk mematikan deteksi perubahan 646
Mulai dan hentikan jaringan pipa secara manual 647
Mulai pipa di CodePipeline 647
Mulai pipa secara manual 649
Mulai pipeline sesuai jadwal 650
Mulai pipeline dengan penggantian revisi sumber 654
Hentikan eksekusi pipeline 656
Hentikan eksekusi pipeline (konsol) 657
Menghentikan Eksekusi Masuk (Konsol) 661
Hentikan eksekusi pipeline (CLI) 662
Hentikan Eksekusi Masuk (CLI) 663
Lihat riwayat dan atur mode untuk eksekusi pipeline 665
Lihat eksekusi 665
Lihat riwayat eksekusi pipeline (konsol) 665
Lihat status eksekusi (konsol) 667
Melihat eksekusi masuk (Konsol) 669
Lihat revisi sumber eksekusi pipeline (konsol) 670
Lihat eksekusi tindakan (konsol) 672
Lihat artefak aksi dan informasi penyimpanan artefak (konsol) 672
Lihat detail dan riwayat saluran pipa (CLI) 673
Mengatur atau mengubah mode eksekusi pipa 685
Pertimbangan untuk melihat mode eksekusi 685
Pertimbangan untuk beralih di antara mode eksekusi 688
Mengatur atau mengubah mode eksekusi pipeline (konsol) 689
Mengatur mode eksekusi pipa (CLI) 690
Gulung kembali atau coba lagi tahapan 694

Mengkonfigurasi percobaan ulang tahap untuk tahap yang gagal atau tindakan yang gagal 694
Pertimbangan untuk percobaan ulang tahap 695
Coba lagi tahap yang gagal secara manual 695
Konfigurasikan tahapan untuk coba lagi otomatis pada kegagalan 700
Mengkonfigurasi rollback panggung 705
Pertimbangan untuk rollback 706
Gulung kembali panggung secara manual 706
Konfigurasikan panggung untuk rollback otomatis 711
Lihat status rollback dalam daftar eksekusi 715
Lihat detail status rollback 718
Konfigurasikan kondisi untuk sebuah panggung 723
Gunakan kasus untuk kondisi panggung 724
Pertimbangan untuk hasil yang dikonfigurasi untuk kondisi panggung 724
Pertimbangan untuk aturan yang dikonfigurasi untuk kondisi panggung 725
Membuat kondisi Entri 726
Membuat kondisi Entri - contoh CloudWatchAlarm aturan (Konsol) 726
Membuat kondisi Entri dengan Lewati hasil dan `VariableCheck` aturan (konsol) 727
Membuat kondisi Entri (CLI) 729
Membuat kondisi Entri (CFN) 731
Menciptakan kondisi Kegagalan 732
Membuat kondisi On Failure (Konsol) 732
Membuat kondisi onFailure dengan contoh hasil Coba Ulang (Konsol) 733
Membuat kondisi Kegagalan (CLI) 734
Membuat kondisi Kegagalan (CFN) 736
Menciptakan kondisi Sukses 737
Membuat kondisi Sukses (Konsol) 737
Menciptakan kondisi Sukses (CLI) 739
Buat kondisi On Success (CFN) 741
Menghapus kondisi panggung 742
Kondisi panggung utama 743
Menggunakan jenis tindakan, tindakan kustom, dan tindakan persetujuan 745
Bekerja dengan tipe tindakan 745
Meminta tipe tindakan 747
Tambahkan tipe tindakan yang tersedia ke pipeline (konsol) 753
Melihat tipe tindakan 755
Memperbarui jenis tindakan 756

Buat tindakan kustom untuk pipeline 758
Buat tindakan kustom 760
Buat pekerja pekerjaan untuk tindakan kustom Anda 764
Menambahkan tindakan kustom ke pipeline 772
Menandai tindakan kustom di CodePipeline 775
Tambahkan tag ke tindakan kustom 775
Lihat tag untuk tindakan kustom 776
Mengedit tag untuk tindakan kustom 776
Hapus tag dari tindakan kustom 776
Memanggil fungsi Lambda dalam pipeline 777
Langkah 1: Buat pipeline 779
Langkah 2: Buat fungsi Lambda 780
Langkah 3: Tambahkan fungsi Lambda ke pipeline di konsol CodePipeline 785
Langkah 4: Uji pipa dengan fungsi Lambda 786
Langkah 5: Langkah selanjutnya 786
Contoh acara JSON 787
Fungsi sampel tambahan 789
Menambahkan tindakan persetujuan manual ke panggung 802
Opsi konfigurasi untuk tindakan persetujuan manual 802
Ikhtisar penyiapan dan alur kerja untuk tindakan persetujuan 803
Berikan izin persetujuan kepada pengguna IAM di CodePipeline 804
Berikan izin Amazon SNS ke peran layanan 807
Menambahkan tindakan persetujuan manual 809
Menyetujui atau menolak tindakan persetujuan 813
Format data JSON untuk pemberitahuan persetujuan manual 817
Menambahkan tindakan Lintas wilayah ke pipeline 818
Mengelola tindakan lintas wilayah dalam pipeline (konsol) 820
Tambahkan tindakan Lintas wilayah ke pipeline (CLI) 822
Menambahkan tindakan Lintas wilayah ke pipeline ()CloudFormation 828
Bekerja dengan variabel 830
Konfigurasikan tindakan untuk variabel 832
Lihat variabel keluaran 836
Contoh: Gunakan variabel dalam persetujuan manual 838
Contoh: Gunakan BranchName variabel dengan variabel CodeBuild lingkungan 839
Bekerja dengan transisi panggung 842
Nonaktifkan atau aktifkan transisi (konsol) 842

Nonaktifkan atau aktifkan transisi (CLI) 844
Memantau pipeline 846
Memantau CodePipeline peristiwa 847
Jenis detail 849
Peristiwa tingkat saluran pipa 851
Acara tingkat panggung 859
Acara tingkat aksi 863
Buat Aturan yang Mengirim Pemberitahuan pada Acara Pipeline 871
Referensi bucket placeholder acara 876
Nama bucket placeholder acara menurut Wilayah 877
Pencatatan panggilan API dengan AWS CloudTrail 880
CodePipeline informasi di CloudTrail 880
Memahami entri file CodePipeline log 881
CodePipeline CloudWatch metrik 883
PipelineDuration 884
FailedPipelineExecutions 884
Pemecahan masalah 886
Kesalahan saluran pipa: Pipeline yang dikonfigurasi dengan AWS Elastic Beanstalk menampilkan pesan kesalahan: “Penerapan gagal. Peran yang diberikan tidak memiliki izin yang memadai: Layanan: AmazonElasticLoadBalancing 887
Kesalahan penerapan: Saluran pipa yang dikonfigurasi dengan tindakan AWS Elastic Beanstalk penerapan hang alih-alih gagal jika izin "" DescribeEvents tidak ada 888
Kesalahan saluran pipa: Tindakan sumber mengembalikan pesan izin yang tidak memadai: “Tidak dapat mengakses CodeCommit `repository-name` repositori. Pastikan bahwa peran IAM pipeline memiliki izin yang cukup untuk mengakses repositori.” 888
Kesalahan saluran pipa: Tindakan build atau pengujian Jenkins berjalan untuk waktu yang lama dan kemudian gagal karena kurangnya kredensyal atau izin 889
Kesalahan pipa: Pipeline yang dibuat di satu AWS Wilayah menggunakan bucket yang dibuat di AWS Wilayah lain mengembalikan InternalError "" dengan kode "JobFailed” 889
Kesalahan penerapan: File ZIP yang berisi file WAR berhasil digunakan AWS Elastic Beanstalk, tetapi URL aplikasi melaporkan kesalahan 404 tidak ditemukan 888
Nama folder artefak pipa tampaknya terpotong 890
Tambahkan CodeBuild GitClone izin untuk koneksi ke Bitbucket,, GitHub Enterprise Server GitHub, atau .com GitLab 891
Tambahkan CodeBuild GitClone izin untuk tindakan CodeCommit sumber 893

<source artifact name>Kesalahan saluran pipa: Penerapan dengan tindakan CodeDeployTo ECS mengembalikan pesan kesalahan: “Pengecualian saat mencoba membaca file artefak definisi tugas dari:” 894
GitHub (melalui OAuth aplikasi) tindakan sumber: Daftar repositori menunjukkan repositori yang berbeda 894
GitHub (melalui GitHub Aplikasi) tindakan sumber: Tidak dapat menyelesaikan koneksi untuk repositori 895
Kesalahan Amazon S3: peran CodePipeline layanan <ARN>mendapatkan akses S3 ditolak untuk bucket S3 < > BucketName 895
Saluran pipa dengan Amazon S3, Amazon ECR, CodeCommit atau sumber tidak lagi dimulai secara otomatis 898
Kesalahan koneksi saat menghubungkan ke GitHub: “Masalah terjadi, pastikan cookie diaktifkan di browser Anda” atau “Pemilik organisasi harus menginstal GitHub aplikasi” 899
Pipelines dengan mode eksekusi diubah menjadi mode QUEUED atau PARALLEL gagal saat batas run tercapai 900
Pipelines dalam mode PARALLEL memiliki definisi pipeline yang sudah ketinggalan zaman jika diedit saat mengubah ke mode QUEUED atau SUPERSEDED 900
Pipelines diubah dari mode PARALLEL akan menampilkan mode eksekusi sebelumnya 900
Saluran pipa dengan koneksi yang menggunakan pemfilteran pemicu berdasarkan jalur file mungkin tidak dimulai pada pembuatan cabang 901
Saluran pipa dengan koneksi yang menggunakan pemfilteran pemicu berdasarkan jalur file mungkin tidak dimulai saat batas file tercapai 901
CodeCommit atau revisi sumber S3 dalam mode PARALLEL mungkin tidak cocok dengan acara EventBridge 902
Tindakan Penerapan EC2 gagal dengan pesan kesalahan `No such file` 902
Tindakan Penerapan EKS gagal dengan pesan `cluster unreachable` kesalahan 903
Butuh bantuan dengan masalah yang berbeda? 904
Keamanan 905
Perlindungan data 906
Privasi lalu lintas antarjaringan 907
Enkripsi saat diam 907
Enkripsi saat bergerak 908
Pengelolaan kunci enkripsi 908
Konfigurasikan enkripsi sisi server untuk artefak yang disimpan di Amazon S3 CodePipeline 908

Gunakan AWS Secrets Manager untuk melacak kata sandi database atau kunci API pihak ketiga ... 911
Manajemen identitas dan akses ... 912
Audiens ... 913
Mengautentikasi dengan identitas ... 913
Mengelola akses menggunakan kebijakan ... 914
Bagaimana AWS CodePipeline bekerja dengan IAM ... 916
Contoh kebijakan berbasis identitas ... 921
Contoh kebijakan berbasis sumber daya ... 950
Pemecahan masalah ... 952
CodePipeline referensi izin ... 954
Kelola peran CodePipeline layanan ... 964
Respons insiden ... 971
Validasi kepatuhan ... 971
Ketahanan ... 971
Keamanan infrastruktur ... 972
Praktik terbaik keamanan ... 972
Referensi struktur pipa ... 974
Deklarasi pipa ... 978
`name` ... 980
`roleArn` ... 981
`artifactStore`ATAU `artifactStores` ... 981
`stages` ... 982
`version` ... 983
`executionMode` ... 983
`pipelineType` ... 983
`variables` ... 984
`triggers` ... 984
`metadata` ... 988
Deklarasi tahap ... 989
`name` ... 992
`actions` ... 992
`conditions` ... 992
`rules` ... 993
Deklarasi tindakan ... 993
`name` ... 997

region ... 997
roleArn ... 998
namespace ... 998
actionTypeId ... 998
InputArtifacts ... 999
outputArtifacts ... 1000
configuration(oleh penyedia tindakan) ... 1001
runOrder ... 1003
Penyedia tindakan yang valid di CodePipeline ... 1004
Pengaturan yang valid untuk PollForSourceChanges parameter ... 1009
Artefak input dan output yang valid untuk setiap jenis tindakan ... 1011
Parameter konfigurasi yang valid untuk setiap jenis penyedia ... 1013
Referensi struktur aksi ... 1017
Referensi tindakan Amazon EC2 ... 1018
Tipe tindakan ... 1019
Parameter konfigurasi ... 1019
Artefak masukan ... 1022
Artefak keluaran ... 1023
Izin kebijakan peran layanan untuk tindakan penerapan EC2 ... 1023
Menyebarkan referensi file spesifikasi ... 1025
Deklarasi tindakan ... 1027
Deklarasi tindakan dengan contoh spesifikasi Deploy ... 1028
Lihat juga ... 1030
Referensi tindakan sumber Amazon ECR ... 1030
Tipe tindakan ... 1031
Parameter konfigurasi ... 1031
Artefak masukan ... 1031
Artefak keluaran ... 1032
Variabel keluaran ... 1032
Izin peran layanan: Tindakan Amazon ECR ... 1033
Deklarasi tindakan (contoh Amazon ECR) ... 1033
Lihat juga ... 1034
ECRBuildAndPublishmembangun referensi tindakan ... 1034
Tipe tindakan ... 1035
Parameter konfigurasi ... 1036
Artefak masukan ... 1037

Artefak keluaran 1037
Variabel keluaran 1037
Izin peran layanan: tindakan ECRBuildAndPublish 1037
Deklarasi tindakan 1039
Lihat juga 1040
Amazon ECS dan referensi CodeDeploy tindakan penerapan biru-hijau 1041
Tipe tindakan 1042
Parameter konfigurasi 1042
Artefak masukan 1043
Artefak keluaran 1044
Izin peran layanan: tindakan CodeDeployToECS 1044
Deklarasi tindakan 1047
Lihat juga 1048
Referensi tindakan penerapan Amazon Elastic Container Service 1049
Tipe tindakan 1050
Parameter konfigurasi 1050
Artefak masukan 1051
Artefak keluaran 1051
Izin peran layanan: Tindakan standar Amazon ECS 1052
Deklarasi tindakan 1053
Lihat juga 1055
Referensi tindakan penerapan Amazon Elastic Kubernetes Service EKS 1055
Tipe tindakan 1056
Parameter konfigurasi 1057
Artefak masukan 1058
Artefak keluaran 1058
Variabel-variabel lingkungan 1058
Variabel keluaran 1059
Izin kebijakan peran layanan 1059
Deklarasi tindakan 1061
Lihat juga 1062
AWS Lambda menyebarkan referensi tindakan 1063
Tipe tindakan 1063
Parameter konfigurasi 1064
Artefak masukan 1067
Artefak keluaran 1067

Variabel keluaran 1067
Izin kebijakan peran layanan untuk tindakan penerapan Lambda 1067
Deklarasi tindakan 1069
Lihat juga 1070
Referensi tindakan penerapan Amazon S3 1070
Tipe tindakan 1071
Parameter konfigurasi 1071
Artefak masukan 1073
Artefak keluaran 1073
Izin peran layanan: Tindakan penerapan S3 1073
Contoh konfigurasi tindakan 1074
Lihat juga 1077
Referensi tindakan sumber Amazon S3 1077
Tipe tindakan 1079
Parameter konfigurasi 1079
Artefak masukan 1081
Artefak keluaran 1081
Variabel keluaran 1081
Izin peran layanan: Tindakan sumber S3 1082
Deklarasi tindakan 1083
Lihat juga 1084
AWS AppConfig menyebarkan referensi tindakan 1084
Tipe tindakan 1084
Parameter konfigurasi 1085
Artefak masukan 1085
Artefak keluaran 1085
Izin peran layanan: tindakan `AppConfig` 1086
Contoh konfigurasi tindakan 1086
Lihat juga 1088
CloudFormation menyebarkan referensi tindakan 1088
Tipe tindakan 1089
Parameter konfigurasi 1089
Artefak masukan 1094
Artefak keluaran 1094
Variabel keluaran 1094
Izin peran layanan: tindakan CloudFormation 1095

Deklarasi tindakan 1097
Lihat juga 1098
CloudFormation StackSets 1098
Bagaimana CloudFormation StackSets tindakan bekerja 1099
Cara menyusun StackSets tindakan dalam pipa 1101
Tindakan `CloudFormationStackSet` 1103
CloudFormationStackInstances Tindakan 1117
Izin peran layanan: tindakan `CloudFormationStackSet` 1127
Izin peran layanan: tindakan `CloudFormationStackInstances` 1127
Model izin untuk operasi set tumpukan 1128
Jenis data parameter template 1128
Lihat juga 1098
AWS CodeBuild membangun dan menguji referensi tindakan 1130
Tipe tindakan 1131
Parameter konfigurasi 1131
Artefak masukan 1134
Artefak keluaran 1134
Variabel keluaran 1135
Izin peran layanan: tindakan CodeBuild 1135
Deklarasi tindakan (CodeBuild contoh) 1136
Lihat juga 1137
AWS CodePipeline memanggil referensi tindakan 1138
Tipe tindakan 1138
Parameter konfigurasi 1139
Artefak masukan 1141
Artefak keluaran 1142
Izin kebijakan peran layanan untuk tindakan CodePipeline pemanggilan 1142
Deklarasi tindakan 1142
Lihat juga 1143
AWS CodeCommit referensi tindakan sumber 1143
Tipe tindakan 1144
Parameter konfigurasi 1145
Artefak masukan 1146
Artefak keluaran 1146
Variabel keluaran 1147
Izin peran layanan: tindakan CodeCommit 1147

Contoh konfigurasi tindakan 1148
Lihat juga 1150
AWS CodeDeploy menyebarkan referensi tindakan 1151
Tipe tindakan 1151
Parameter konfigurasi 1151
Artefak masukan 1152
Artefak keluaran 1152
Izin peran layanan: tindakan AWS CodeDeploy 1152
Deklarasi tindakan 1153
Lihat juga 1155
CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri 1155
Tipe tindakan 1159
Parameter konfigurasi 1159
Artefak masukan 1160
Artefak keluaran 1161
Variabel keluaran 1161
Izin peran layanan: tindakan CodeConnections 1162
Deklarasi tindakan 1162
Menginstal aplikasi instalasi dan membuat koneksi 1164
Lihat juga 1164
Referensi tindakan perintah 1165
Pertimbangan untuk tindakan Perintah 1166
Izin kebijakan peran layanan 1167
Tipe tindakan 1168
Parameter konfigurasi 1168
Artefak masukan 1173
Artefak keluaran 1173
Variabel-variabel lingkungan 1173
Izin peran layanan: Tindakan perintah 1175
Deklarasi tindakan (contoh) 1175
Lihat juga 1177
AWS Device Farm referensi tindakan uji 1177
Tipe tindakan 1177
Parameter konfigurasi 1178
Artefak masukan 1182

Artefak keluaran 1182
Izin peran layanan: tindakan AWS Device Farm 1182
Deklarasi tindakan 1183
Lihat juga 1184
Referensi aksi penyebaran Elastic Beanstalk 1184
Tipe tindakan 1185
Parameter konfigurasi 1185
Artefak masukan 1185
Artefak keluaran 1186
Izin peran layanan: tindakan `ElasticBeanstalk` penerapan 1186
Deklarasi tindakan 1187
Lihat juga 1188
Amazon Inspector `InspectorScan` memanggil referensi tindakan 1188
ID tipe tindakan 1189
Parameter konfigurasi 1189
Artefak masukan 1192
Artefak keluaran 1192
Variabel keluaran 1192
Izin peran layanan: tindakan `InspectorScan` 1192
Deklarasi tindakan 1193
Lihat juga 1194
AWS Lambda memanggil referensi tindakan 1195
Tipe tindakan 1195
Parameter konfigurasi 1195
Artefak masukan 1196
Artefak keluaran 1196
Variabel keluaran 1196
Contoh konfigurasi tindakan 1196
Contoh acara JSON 1197
Lihat juga 1200
AWS OpsWorks menyebarkan referensi tindakan 1200
Tipe tindakan 1200
Parameter konfigurasi 1200
Artefak masukan 1201
Artefak keluaran 1201
Izin peran layanan: tindakan AWS OpsWorks 1201

Contoh konfigurasi tindakan 1202
Lihat juga 1202
AWS Service Catalog menyebarkan referensi tindakan 1203
Tipe tindakan 1203
Parameter konfigurasi 1203
Artefak masukan 1204
Artefak keluaran 1204
Izin peran layanan: Tindakan Service Catalog 1204
Contoh konfigurasi tindakan berdasarkan jenis file konfigurasi 1205
Lihat juga 1206
AWS Step Functions 1206
Tipe tindakan 1200
Parameter konfigurasi 1207
Artefak masukan 1208
Artefak keluaran 1209
Variabel keluaran 1209
Izin peran layanan: tindakan `StepFunctions` 1209
Contoh konfigurasi tindakan 1210
Perilaku 1213
Lihat juga 1088
Referensi struktur aturan 1216
CloudWatchAlarm 1216
Jenis aturan 1217
Parameter konfigurasi 1217
Contoh konfigurasi aturan 1217
Lihat juga 1218
CodeBuild aturan 1219
Izin kebijakan peran layanan 1219
Jenis aturan 1220
Parameter konfigurasi 1220
Contoh konfigurasi aturan 1222
Lihat juga 1223
Commands 1223
Pertimbangan untuk aturan Perintah 1223
Izin kebijakan peran layanan 1219
Jenis aturan 1220

Parameter konfigurasi ... 1220
Contoh konfigurasi aturan ... 1222
Lihat juga ... 1223
DeploymentWindow ... 1228
Jenis aturan ... 1229
Parameter konfigurasi ... 1229
Contoh konfigurasi aturan ... 1231
Lihat juga ... 1232
LambdaInvoke ... 1232
Jenis aturan ... 1217
Parameter konfigurasi ... 1233
Contoh konfigurasi aturan ... 1233
Lihat juga ... 1234
VariableCheck ... 1234
Jenis aturan ... 1235
Parameter konfigurasi ... 1235
Contoh konfigurasi aturan ... 1238
Lihat juga ... 1238
Referensi model integrasi ... 1240
Cara kerja tipe tindakan pihak ketiga dengan integrator ... 1240
Konsep ... 1241
Model integrasi yang didukung ... 1243
Model integrasi Lambda ... 1244
Perbarui fungsi Lambda Anda untuk menangani input dari CodePipeline ... 1245
Kembalikan hasil dari fungsi Lambda Anda ke CodePipeline ... 1249
Gunakan token lanjutan untuk menunggu hasil dari proses asinkron ... 1251
Berikan CodePipeline izin untuk menjalankan fungsi Lambda integrator saat runtime ... 1251
Model integrasi pekerja Job ... 1252
Pilih dan konfigurasikan strategi manajemen izin untuk pekerja kerja Anda ... 1252
Referensi file definisi gambar ... 1255
file imagedefinitions.json untuk tindakan penerapan standar Amazon ECS ... 1255
File ImageDetail.json untuk tindakan penerapan Amazon ECS blue/green ... 1258
Referensi variabel ... 1263
Konsep ... 1264
Variabel ... 1264
Namespace ... 1265

Gunakan kasus untuk variabel 1266
Mengkonfigurasi variabel 1267
Mengkonfigurasi variabel di tingkat pipa 1267
Mengkonfigurasi variabel pada tingkat tindakan 1268
Resolusi variabel 1270
Aturan untuk variabel 1271
Variabel tersedia untuk tindakan pipeline 1272
Tindakan dengan kunci variabel yang ditentukan 1272
Tindakan dengan kunci variabel yang dikonfigurasi pengguna 1276
Bekerja dengan pola glob dalam sintaks 1279
Perbarui jalur pemungutan suara ke metode deteksi perubahan yang direkomendasikan 1281
Perbarui tindakan sumber GitHub (melalui OAuth aplikasi) ke tindakan sumber GitHub (melalui GitHub Aplikasi) 1282
Langkah 1: Ganti GitHub tindakan Anda (melalui OAuth aplikasi) 1283
Langkah 2: Buat koneksi ke GitHub 1284
Langkah 3: Simpan tindakan GitHub sumber Anda 1285
Kuota 1287
Lampiran A: GitHub (melalui OAuth aplikasi) tindakan sumber 1303
Menambahkan tindakan sumber GitHub (melalui OAuth aplikasi) 1304
GitHub (melalui OAuth aplikasi) referensi tindakan sumber 1305
Tipe tindakan 1306
Parameter konfigurasi 1306
Artefak masukan 1308
Artefak keluaran 1308
Variabel keluaran 1308
Deklarasi tindakan (GitHub contoh) 1309
Menghubungkan ke GitHub (OAuth) 1310
Lihat juga 1311
Riwayat dokumen 1312
Pembaruan lebih awal 1345
CodePipeline referensi fitur 1357
..... mccclx

Apa itu AWS CodePipeline?

AWS CodePipeline adalah layanan pengiriman berkelanjutan yang dapat Anda gunakan untuk memodelkan, memvisualisasikan, dan mengotomatiskan langkah-langkah yang diperlukan untuk merilis perangkat lunak Anda. Anda dapat dengan cepat memodelkan dan mengonfigurasi berbagai tahapan proses rilis perangkat lunak. CodePipeline mengotomatiskan langkah-langkah yang diperlukan untuk merilis perubahan perangkat lunak Anda secara terus menerus. Untuk informasi tentang harga CodePipeline, lihat Harga.

Topik

- Pengiriman berkelanjutan dan integrasi berkelanjutan
- Apa yang bisa saya lakukan dengan CodePipeline?
- Sekilas melihat CodePipeline
- Bagaimana saya memulai CodePipeline?
- CodePipeline konsep
- DevOps contoh pipa
- Bagaimana eksekusi pipa bekerja
- Artefak input dan output
- Bagaimana kondisi panggung bekerja?
- Jenis pipa
- Jenis pipa apa yang tepat untuk saya?

Pengiriman berkelanjutan dan integrasi berkelanjutan

CodePipeline adalah layanan pengiriman berkelanjutan yang mengotomatiskan pembangunan, pengujian, dan penyebaran perangkat lunak Anda ke dalam produksi.

Continuous delivery adalah metodologi pengembangan perangkat lunak di mana proses rilis otomatis. Setiap perubahan perangkat lunak secara otomatis dibangun, diuji, dan digunakan untuk produksi. Sebelum dorongan akhir untuk produksi, seseorang, tes otomatis, atau aturan bisnis memutuskan kapan dorongan akhir harus terjadi. Meskipun setiap perubahan perangkat lunak yang berhasil dapat segera dirilis ke produksi dengan pengiriman berkelanjutan, tidak semua perubahan harus segera dirilis.

Integrasi berkelanjutan adalah praktik pengembangan perangkat lunak di mana anggota tim menggunakan sistem kontrol versi dan sering mengintegrasikan pekerjaan mereka ke lokasi yang sama, seperti cabang utama. Setiap perubahan dibuat dan diverifikasi untuk mendeteksi kesalahan integrasi secepat mungkin. Integrasi berkelanjutan difokuskan pada pembuatan dan pengujian kode secara otomatis, dibandingkan dengan pengiriman berkelanjutan, yang mengotomatiskan seluruh proses rilis perangkat lunak hingga produksi.

Untuk informasi selengkapnya, lihat Mempraktikkan Integrasi Berkelanjutan dan Pengiriman Berkelanjutan di AWS: Mempercepat Pengiriman Perangkat Lunak dengan DevOps.

Anda dapat menggunakan CodePipeline konsol, AWS Command Line Interface (AWS CLI), the AWS SDKs, atau kombinasi apa pun untuk membuat dan mengelola pipeline Anda.

Apa yang bisa saya lakukan dengan CodePipeline?

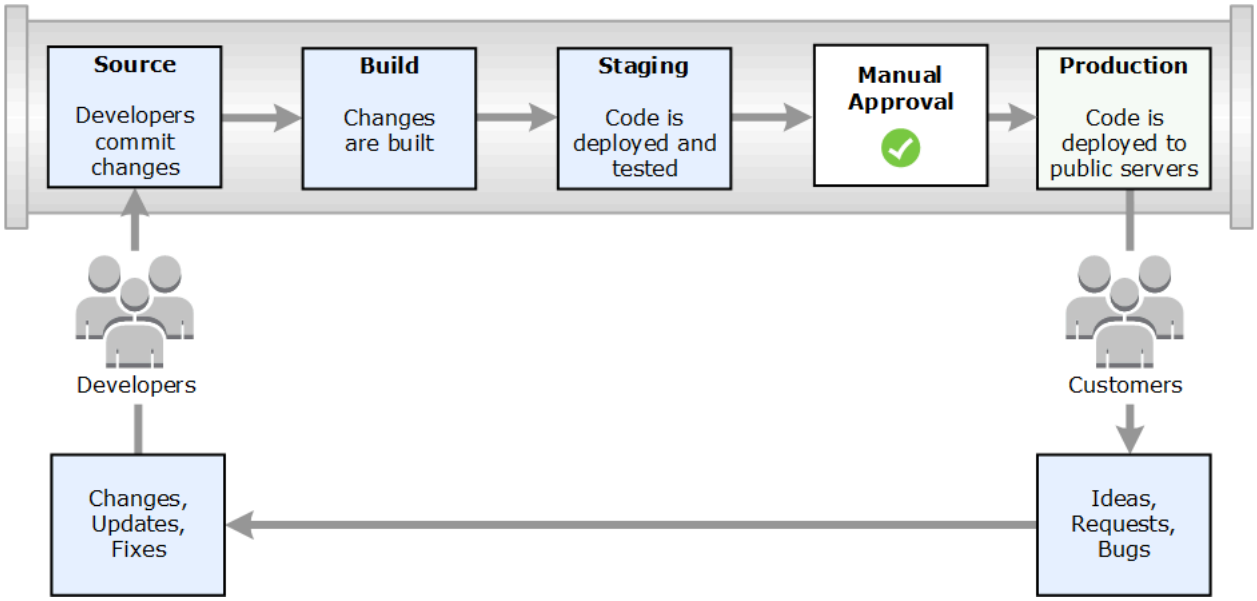
Anda dapat menggunakan CodePipeline untuk membantu Anda secara otomatis membangun, menguji, dan menyebarkan aplikasi Anda di cloud. Secara khusus, Anda dapat:

- Otomatiskan proses rilis Anda: mengotomatiskan CodePipeline sepenuhnya proses rilis Anda dari ujung ke ujung, mulai dari repositori sumber Anda hingga pembuatan, pengujian, dan penerapan. Anda dapat mencegah perubahan bergerak melalui pipeline dengan menyertakan tindakan persetujuan manual di tahap apa pun kecuali tahap Sumber. Anda dapat melepaskan kapan pun Anda inginkan, dengan cara yang Anda inginkan, pada sistem pilihan Anda, di satu contoh atau beberapa contoh.
- Tetapkan proses rilis yang konsisten: Tentukan serangkaian langkah yang konsisten untuk setiap perubahan kode. CodePipeline menjalankan setiap tahap rilis Anda sesuai dengan kriteria Anda.
- Mempercepat pengiriman sekaligus meningkatkan kualitas: Anda dapat mengotomatiskan proses rilis untuk memungkinkan pengembang menguji dan merilis kode secara bertahap dan mempercepat rilis fitur baru kepada pelanggan Anda.
- Gunakan alat favorit Anda: Anda dapat menggabungkan alat sumber, build, dan penerapan yang ada ke dalam pipeline Anda. Untuk daftar lengkap Layanan AWS dan alat pihak ketiga yang saat ini didukung oleh CodePipeline, lihatIntegrasi produk dan layanan dengan CodePipeline.
- Melihat kemajuan sekilas: Anda dapat meninjau status saluran pipa secara real-time, memeriksa detail peringatan apa pun, mencoba lagi tahapan atau tindakan yang gagal, melihat detail tentang revisi sumber yang digunakan dalam eksekusi pipeline terbaru di setiap tahap, dan menjalankan ulang saluran apa pun secara manual.

- Melihat detail riwayat pipeline: Anda dapat melihat detail tentang eksekusi pipeline, termasuk waktu mulai dan berakhir, durasi proses, dan eksekusi IDs.

Sekilas melihat CodePipeline

Diagram berikut menunjukkan contoh proses rilis menggunakan CodePipeline.



Dalam contoh ini, ketika pengembang melakukan perubahan ke repositori sumber, CodePipeline secara otomatis mendeteksi perubahan. Perubahan tersebut dibuat, dan jika ada pengujian yang dikonfigurasi, pengujian tersebut dijalankan. Setelah pengujian selesai, kode yang dibangun di-deploy ke server penahapan untuk pengujian. Dari server pementasan, CodePipeline jalankan lebih banyak tes, seperti integrasi atau tes beban. Setelah berhasil menyelesaikan pengujian tersebut, dan setelah tindakan persetujuan manual yang ditambahkan ke pipeline disetujui, CodePipeline menyebarkan kode yang diuji dan disetujui ke instance produksi.

CodePipeline dapat menyebarkan aplikasi ke instans EC2 dengan menggunakan CodeDeploy,, AWS Elastic Beanstalk atau. AWS OpsWorks Stacks CodePipeline juga dapat menyebarkan aplikasi berbasis kontainer ke layanan dengan menggunakan Amazon ECS. Pengembang juga dapat menggunakan titik integrasi yang disediakan CodePipeline untuk menyambungkan alat atau layanan lain, termasuk layanan build, penyedia pengujian, atau target atau sistem penyebaran lainnya.

Pipeline bisa sesederhana atau serumit yang dibutuhkan proses rilis Anda.

Bagaimana saya memulai CodePipeline?

Untuk memulai dengan CodePipeline:

1. Pelajari cara CodePipeline kerjanya dengan membaca CodePipeline konsep bagian ini.
2. Bersiaplah untuk digunakan CodePipeline dengan mengikuti langkah-langkah diMemulai dengan CodePipeline.
3. Bereksperimenlah CodePipeline dengan mengikuti langkah-langkah dalam CodePipeline tutorial tutorial.
4. Gunakan CodePipeline untuk proyek baru atau yang sudah ada dengan mengikuti langkah-langkahnyaBuat pipeline, tahapan, dan tindakan.

CodePipeline konsep

Memodelkan dan mengonfigurasi proses rilis otomatis Anda lebih mudah jika Anda memahami konsep dan istilah yang digunakan. AWS CodePipeline Berikut adalah beberapa konsep yang perlu diketahui saat Anda menggunakan CodePipeline.

Untuk contoh DevOps pipa, lihatDevOps contoh pipa.

Istilah-istilah berikut digunakan dalam CodePipeline:

Topik

- Alur
- Eksekusi pipa
- Operasi panggung
- Eksekusi aksi
- Jenis eksekusi
- Jenis tindakan
- Artifacts
- Revisi sumber
- Pemicu
- Variabel

- Ketentuan
- Aturan

Alur

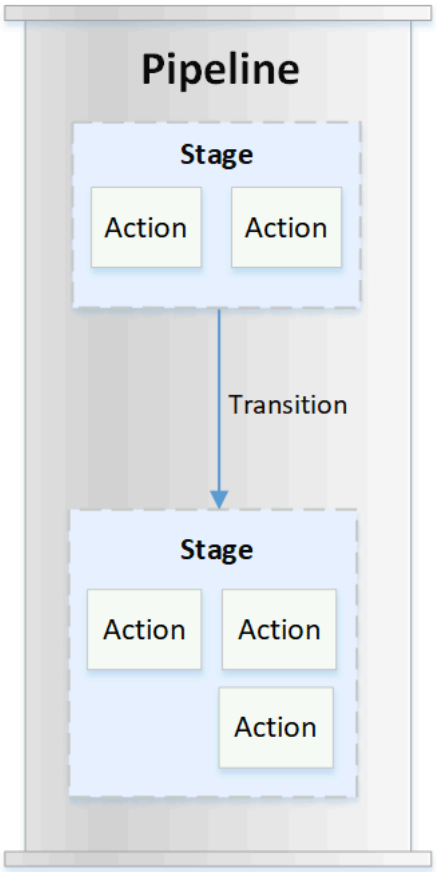
Pipeline adalah konstruksi alur kerja yang menjelaskan bagaimana perubahan perangkat lunak melalui proses rilis. Setiap pipa terdiri dari serangkaian tahapan.

Tahapan

Tahap adalah unit logis yang dapat Anda gunakan untuk mengisolasi lingkungan dan membatasi jumlah perubahan bersamaan di lingkungan itu. Setiap tahap berisi tindakan yang dilakukan pada artefak aplikasi. Kode sumber Anda adalah contoh artefak. Tahap mungkin merupakan tahap build, di mana kode sumber dibangun dan pengujian dijalankan. Ini juga bisa menjadi tahap penerapan, di mana kode diterapkan ke lingkungan runtime. Setiap tahap terdiri dari serangkaian tindakan serial atau paralel.

Transisi

Transisi adalah titik di mana eksekusi pipa bergerak ke tahap berikutnya dalam pipa. Anda dapat menonaktifkan transisi masuk tahap untuk mencegah eksekusi memasuki tahap itu, dan kemudian Anda dapat mengaktifkan transisi untuk memungkinkan eksekusi berlanjut. Ketika lebih dari satu eksekusi tiba pada transisi yang dinonaktifkan, hanya eksekusi terbaru yang berlanjut ke tahap berikutnya ketika transisi diaktifkan. Ini berarti bahwa eksekusi yang lebih baru terus menggantikan eksekusi menunggu saat transisi dinonaktifkan, dan kemudian setelah transisi diaktifkan, eksekusi yang berlanjut adalah eksekusi pengganti.



Tindakan

Tindakan adalah serangkaian operasi yang dilakukan pada kode aplikasi dan dikonfigurasi sehingga tindakan berjalan di pipeline pada titik tertentu. Ini dapat mencakup hal-hal seperti tindakan sumber dari perubahan kode, tindakan untuk menerapkan aplikasi ke instance, dan sebagainya. Misalnya, tahap penerapan mungkin berisi tindakan penerapan yang menyebarkan kode ke layanan komputasi seperti Amazon atau. EC2 AWS Lambda

Jenis CodePipeline tindakan yang valid adalah`source`,`build`,`test`,`deploy`,`approval`, dan`invoke`. Untuk daftar penyedia tindakan, lihatPenyedia tindakan yang valid di CodePipeline.

Tindakan dapat berjalan secara seri atau paralel. Untuk informasi tentang tindakan serial dan paralel dalam satu tahap, lihat `runOrder` informasi dalam persyaratan struktur tindakan.

Eksekusi pipa

Eksekusi adalah serangkaian perubahan yang dirilis oleh pipeline. Setiap eksekusi pipeline unik dan memiliki ID sendiri. Eksekusi sesuai dengan serangkaian perubahan, seperti komit gabungan atau

rilis manual dari komit terbaru. Dua eksekusi dapat melepaskan serangkaian perubahan yang sama pada waktu yang berbeda.

Sementara pipeline dapat memproses beberapa eksekusi pada saat yang sama, tahap pipeline hanya memproses satu eksekusi pada satu waktu. Untuk melakukan ini, sebuah panggung dikunci saat memproses eksekusi. Dua eksekusi pipa tidak dapat menempati tahap yang sama pada saat yang sama. Eksekusi yang menunggu untuk memasuki tahap yang diduduki disebut eksekusi masuk. Eksekusi inbound masih bisa gagal, digantikan, atau dihentikan secara manual. Untuk informasi selengkapnya tentang cara kerja eksekusi masuk, lihat. Bagaimana Eksekusi Inbound Bekerja

Eksekusi pipa melintasi tahapan pipa secara berurutan. Status yang valid untuk jaringan pipa adalah`InProgress`,,`Stopping`,, `Stopped` `Succeeded`Superseded, dan. `Failed`

Untuk informasi selengkapnya, lihat PipelineExecution.

Menghentikan eksekusi

Eksekusi pipeline dapat dihentikan secara manual sehingga eksekusi pipeline yang sedang berlangsung tidak berlanjut melalui pipeline. Jika dihentikan secara manual, eksekusi pipeline menunjukkan `Stopping` status sampai benar-benar berhenti. Kemudian menunjukkan `Stopped` status. Eksekusi `Stopped` pipeline dapat dicoba ulang.

Ada dua cara untuk menghentikan eksekusi pipeline:

- Berhenti dan tunggu
- Berhenti dan tinggalkan

Untuk informasi tentang kasus penggunaan untuk menghentikan eksekusi dan detail urutan untuk opsi ini, lihatBagaimana eksekusi pipa dihentikan.

Eksekusi gagal

Jika eksekusi gagal, itu berhenti dan tidak sepenuhnya melintasi pipa. Statusnya adalah `FAILED` status dan panggung tidak terkunci. Eksekusi yang lebih baru dapat mengejar dan memasuki tahap yang tidak terkunci dan menguncinya. Anda dapat mencoba kembali eksekusi yang gagal kecuali eksekusi yang gagal telah digantikan atau tidak dapat dicoba ulang. Anda dapat memutar kembali tahap yang gagal ke eksekusi yang berhasil sebelumnya.

Mode eksekusi

Untuk memberikan serangkaian perubahan terbaru melalui pipeline, eksekusi yang lebih baru melewati dan menggantikan eksekusi yang kurang baru yang sudah berjalan melalui pipa. Ketika ini terjadi, eksekusi yang lebih lama digantikan oleh eksekusi yang lebih baru. Eksekusi dapat digantikan oleh eksekusi yang lebih baru pada titik tertentu, yang merupakan titik antar tahapan. SUPERSEDED adalah mode eksekusi default.

Dalam mode SUPERSEDED, jika eksekusi menunggu untuk memasuki tahap terkunci, eksekusi yang lebih baru mungkin mengejar dan menggantikannya. Eksekusi yang lebih baru sekarang menunggu panggung dibuka, dan eksekusi yang digantikan berhenti dengan status. `SUPERSEDED` Ketika eksekusi pipeline digantikan, eksekusi dihentikan dan tidak sepenuhnya melintasi pipa. Anda tidak dapat lagi mencoba lagi eksekusi yang digantikan setelah diganti pada tahap ini. Mode eksekusi lain yang tersedia adalah mode PARALLEL atau QUEUED.

Untuk informasi selengkapnya tentang mode eksekusi dan tahapan terkunci, lihatBagaimana eksekusi diproses dalam mode SUPERSEDED.

Operasi panggung

Ketika eksekusi pipeline berjalan melalui suatu tahap, tahapannya sedang dalam proses menyelesaikan semua tindakan di dalamnya. Untuk informasi tentang cara kerja operasi tahapan dan informasi tentang tahapan terkunci, lihatBagaimana eksekusi diproses dalam mode SUPERSEDED.

Status yang valid untuk tahapan adalah`InProgress`,`Stopping`,, `StoppedSucceeded`, dan`Failed`. Anda dapat mencoba lagi tahap yang gagal kecuali tahap yang gagal tidak dapat dicoba kembali. Untuk informasi selengkapnya, lihat StageExecution. Anda dapat memutar kembali tahap ke eksekusi sukses sebelumnya yang ditentukan. Sebuah panggung dapat dikonfigurasi untuk memutar kembali secara otomatis pada kegagalan seperti yang dijelaskan dalamMengkonfigurasi rollback panggung. Untuk informasi selengkapnya, lihat RollbackStage.

Eksekusi aksi

Eksekusi tindakan adalah proses menyelesaikan tindakan yang dikonfigurasi yang beroperasi pada artefak yang ditunjuk. Ini bisa berupa artefak input, artefak keluaran, atau keduanya. Misalnya, tindakan build mungkin menjalankan perintah build pada artefak input, seperti mengompilasi kode sumber aplikasi. Detail eksekusi tindakan mencakup ID eksekusi tindakan, pemicu sumber eksekusi pipeline terkait, dan artefak input dan output untuk tindakan tersebut.

Status yang valid untuk tindakan adalah`InProgress`,, `Abandoned``Succeeded`, atau`Failed`. Untuk informasi selengkapnya, lihat ActionExecution.

Jenis eksekusi

Pipeline atau eksekusi tahap dapat berupa eksekusi standar atau rolled-back.

Untuk tipe standar, eksekusi memiliki ID unik dan merupakan jalur pipa penuh. Rollback pipeline memiliki tahap untuk digulung kembali dan eksekusi yang berhasil untuk tahap sebagai eksekusi target untuk memutar kembali. Eksekusi pipeline target digunakan untuk mengambil revisi sumber dan variabel untuk tahap yang akan dijalankan kembali.

Jenis tindakan

Jenis tindakan adalah tindakan yang telah dikonfigurasi sebelumnya yang tersedia untuk dipilih. CodePipeline Jenis tindakan ditentukan oleh pemilik, penyedia, versi, dan kategorinya. Jenis tindakan menyediakan parameter khusus yang digunakan untuk menyelesaikan tugas tindakan dalam pipeline.

Untuk informasi tentang produk Layanan AWS dan layanan pihak ketiga yang dapat Anda integrasikan ke dalam pipeline berdasarkan jenis tindakan, lihatIntegrasi dengan tipe CodePipeline tindakan.

Untuk informasi tentang model integrasi yang didukung untuk tipe tindakan CodePipeline, lihatReferensi model integrasi.

Untuk informasi tentang cara penyedia pihak ketiga dapat mengatur dan mengelola jenis tindakan CodePipeline, lihatBekerja dengan tipe tindakan.

Artifacts

Artefak mengacu pada pengumpulan data, seperti kode sumber aplikasi, aplikasi yang dibangun, dependensi, file definisi, templat, dan sebagainya, yang dikerjakan oleh tindakan pipeline. Artefak diproduksi oleh beberapa tindakan dan dikonsumsi oleh orang lain. Dalam sebuah pipeline, artefak dapat berupa kumpulan file yang dikerjakan oleh suatu tindakan (artefak input) atau output yang diperbarui dari tindakan yang diselesaikan (artefak keluaran).

Tindakan meneruskan output ke tindakan lain untuk diproses lebih lanjut menggunakan bucket artefak pipa. CodePipeline menyalin artefak ke toko artefak, tempat aksi mengambilnya. Untuk informasi lebih lanjut tentang artifact, lihat Artefak input dan output.

Revisi sumber

Ketika Anda membuat perubahan kode sumber, versi baru dibuat. Revisi sumber adalah versi perubahan sumber yang memicu eksekusi pipeline. Eksekusi memproses revisi sumber. Untuk GitHub dan CodeCommit repositori, ini adalah komit. Untuk bucket atau tindakan S3, ini adalah versi objek.

Anda dapat memulai eksekusi pipeline dengan revisi sumber, seperti komit, yang Anda tentukan. Eksekusi akan memproses revisi yang ditentukan dan mengganti apa yang akan menjadi revisi yang digunakan untuk eksekusi. Untuk informasi selengkapnya, lihat Mulai pipeline dengan penggantian revisi sumber.

Pemicu

Pemicu adalah peristiwa yang memulai pipeline Anda. Beberapa pemicu, seperti memulai pipeline secara manual, tersedia untuk semua penyedia aksi sumber dalam pipeline. Pemicu tertentu bergantung pada penyedia sumber untuk pipa. Misalnya, CloudWatch peristiwa harus dikonfigurasi dengan sumber daya peristiwa dari Amazon CloudWatch yang memiliki ARN pipeline yang ditambahkan sebagai target dalam aturan peristiwa. Amazon CloudWatch Events adalah pemicu yang disarankan untuk deteksi perubahan otomatis untuk saluran pipa dengan tindakan sumber CodeCommit atau S3. Webhook adalah jenis pemicu yang dikonfigurasi untuk peristiwa repositori pihak ketiga. Misalnya, WebHookv2 adalah tipe pemicu yang memungkinkan tag Git digunakan untuk memulai pipeline dengan penyedia sumber pihak ketiga GitHub seperti.com, GitHub Enterprise Server, .com, GitLab self-managed, atau GitLab Bitbucket Cloud. Dalam konfigurasi pipeline, Anda dapat menentukan filter untuk pemicu, seperti permintaan push atau pull. Anda dapat memfilter peristiwa push kode pada tag Git, cabang, atau jalur file. Anda dapat mengisi peristiwa permintaan tarik pada acara (dibuka, diperbarui, ditutup), cabang, atau jalur fille.

Untuk informasi lebih lanjut tentang menggunakan pemicu, lihat Mulai pipa di CodePipeline. Untuk tutorial yang memandu Anda menggunakan tag Git sebagai pemicu pipeline Anda, lihatTutorial: Gunakan tag Git untuk memulai pipeline Anda.

> **Important**
>
> Pipa yang tidak aktif selama lebih dari 30 hari akan menonaktifkan pemungutan suara untuk pipa. Untuk informasi lebih lanjut, lihat pollingDisabledAtdi referensi struktur pipa. Untuk langkah-langkah memigrasikan pipeline Anda dari polling ke deteksi perubahan berbasis peristiwa, lihat Mengubah Metode Deteksi.

Variabel

Variabel adalah nilai yang dapat digunakan untuk mengonfigurasi tindakan secara dinamis di pipeline Anda. Variabel dapat dideklarasikan pada tingkat pipa, atau dipancarkan oleh tindakan dalam pipa. Nilai variabel diselesaikan pada saat eksekusi pipeline dan dapat dilihat dalam riwayat eksekusi. Untuk variabel yang dideklarasikan pada tingkat pipeline, Anda dapat menentukan nilai default dalam konfigurasi pipeline, atau menggantinya untuk eksekusi tertentu. Untuk variabel yang dipancarkan oleh suatu tindakan, nilainya tersedia setelah tindakan berhasil diselesaikan. Untuk informasi selengkapnya, lihat Referensi variabel.

Ketentuan

Suatu kondisi berisi seperangkat aturan yang dievaluasi. Jika semua aturan dalam suatu kondisi berhasil, maka kondisinya terpenuhi. Anda dapat mengonfigurasi kondisi sehingga ketika kriteria tidak terpenuhi, hasil yang ditentukan, seperti gagal tahap, terlibat. Kondisi juga disebut sebagai gerbang karena memungkinkan Anda untuk menentukan kapan eksekusi akan masuk dan berjalan melalui tahap atau keluar dari panggung setelah menjalankannya. Ini analog dengan memungkinkan garis lalu lintas di jalan berkumpul di gerbang tertutup, dan kemudian menentukan pembukaan gerbang untuk memungkinkan arus lalu lintas ke suatu daerah. Hasil untuk jenis kondisi termasuk gagal panggung atau memutar kembali panggung. Kondisi membantu Anda menentukan kapan tindakan ini terjadi dalam tahap pipeline. Anda dapat mengganti kondisi saat runtime.

Ada tiga jenis kondisi. Kondisi masuk menjawab pertanyaan “Jika aturan untuk kondisi terpenuhi, maka masuki panggung.” Panggung terkunci ketika eksekusi memasuki panggung, dan kemudian aturan dijalankan. Untuk kondisi On Failure, aturan berlaku ketika sebuah tahap gagal, dengan hasil memutar kembali tahap yang gagal. Untuk kondisi On Success, aturan berlaku ketika sebuah tahap berhasil, seperti memeriksa alarm yang berhasil dijalankan sebelum melanjutkan. Misalnya, kondisi On Success akan menghasilkan rollback dari tahap yang berhasil jika CloudWatchAlarm aturan menemukan bahwa ada alarm di lingkungan penyebaran. Untuk informasi selengkapnya, lihat Bagaimana kondisi panggung bekerja?.

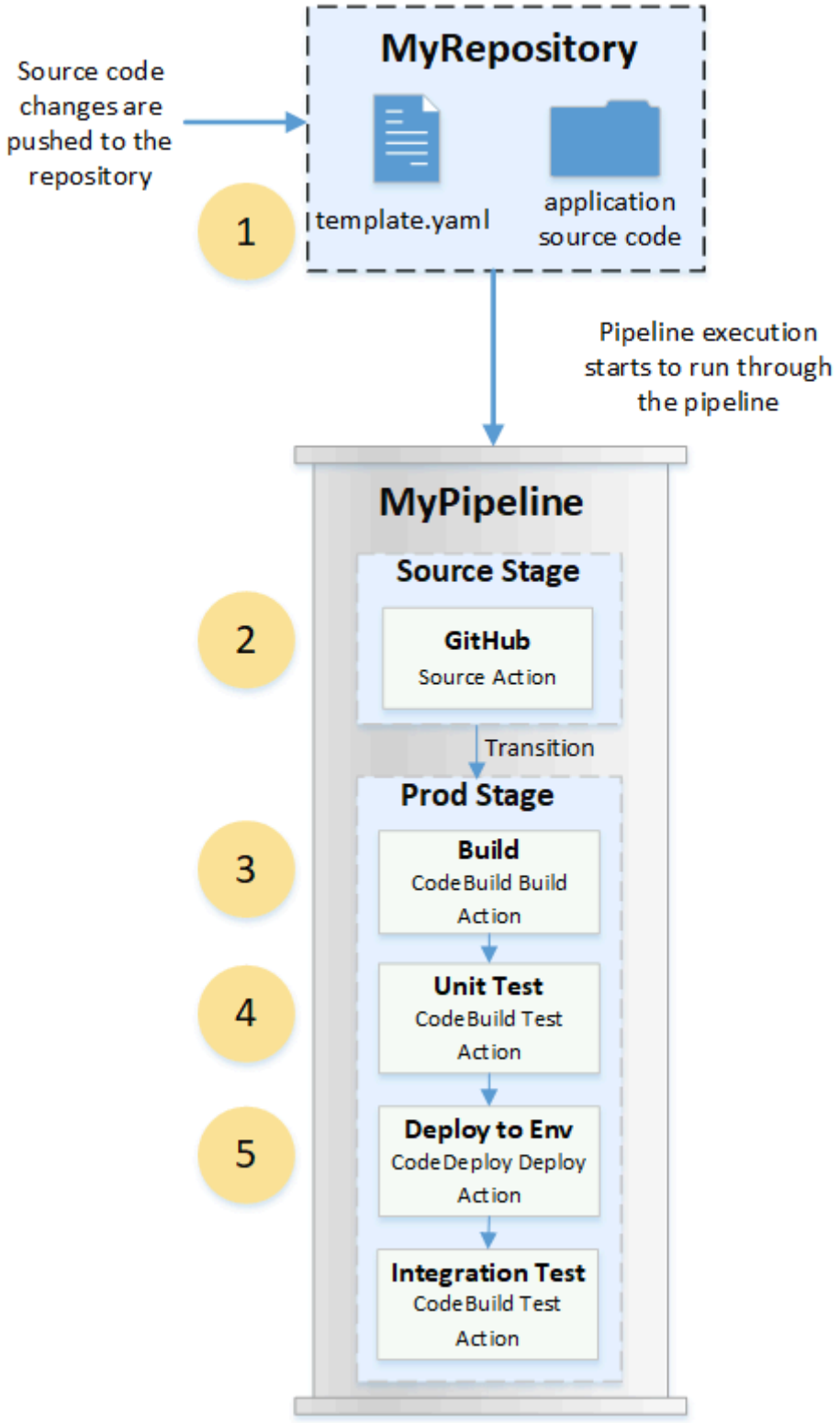
Aturan

Ketentuan menggunakan satu atau beberapa aturan yang telah dikonfigurasi sebelumnya yang menjalankan dan melakukan pemeriksaan yang kemudian akan melibatkan hasil yang dikonfigurasi ketika kondisi tidak terpenuhi. Misalnya, memenuhi semua aturan untuk aturan kondisi Entri yang memeriksa status alarm dan waktu jendela penerapan akan menerapkan tahap yang berhasil setelah

semua pemeriksaan lulus. Lihat informasi yang lebih lengkap di Bagaimana kondisi panggung bekerja?.

DevOps contoh pipa

Sebagai contoh DevOps pipa, pipa dua tahap mungkin memiliki tahap sumber yang disebut Sumber dan tahap kedua yang disebut Prod. Dalam contoh ini, pipeline memperbarui aplikasi dengan perubahan terbaru dan terus menerapkan hasil terbaru. Sebelum menyebarkan aplikasi terbaru, pipeline membangun dan menguji aplikasi web. Dalam contoh ini, sekelompok pengembang telah menyiapkan template infrastruktur dan kode sumber untuk aplikasi web dalam GitHub repositori yang disebut. MyRepository



Misalnya, pengembang mendorong perbaikan ke halaman indeks aplikasi web, dan hal berikut terjadi:

1. Kode sumber aplikasi dipertahankan dalam repositori yang dikonfigurasi sebagai tindakan GitHub sumber dalam pipeline. Saat pengembang mendorong commit ke repositori, CodePipeline mendeteksi perubahan yang didorong, dan eksekusi pipeline dimulai dari Source Stage.
2. Tindakan GitHub sumber berhasil diselesaikan (yaitu, perubahan terbaru telah diunduh dan disimpan ke ember artefak yang unik untuk eksekusi itu). Artefak keluaran yang dihasilkan oleh aksi GitHub sumber, yang merupakan file aplikasi dari repositori, kemudian digunakan sebagai artefak input untuk dikerjakan oleh tindakan pada tahap berikutnya.
3. Eksekusi pipeline bertransisi dari Source Stage ke Prod Stage. Tindakan pertama di Prod Stage menjalankan proyek build yang dibuat CodeBuild dan dikonfigurasi sebagai aksi build dalam pipeline. Tugas build menarik image lingkungan build dan membangun aplikasi web dalam wadah virtual.
4. Tindakan selanjutnya di Prod Stage adalah proyek pengujian unit yang dibuat CodeBuild dan dikonfigurasi sebagai tindakan pengujian dalam pipeline.
5. Kode unit yang diuji selanjutnya dikerjakan oleh tindakan penerapan di Tahap Prod yang menyebarkan aplikasi ke lingkungan produksi. Setelah tindakan penerapan selesai dengan sukses, tindakan terakhir dalam tahap adalah proyek pengujian integrasi yang dibuat CodeBuild dan dikonfigurasi sebagai tindakan pengujian dalam pipeline. Tindakan pengujian memanggil skrip shell yang menginstal dan menjalankan alat uji, seperti pemeriksa tautan, di aplikasi web. Setelah berhasil diselesaikan, outputnya adalah aplikasi web yang dibangun dan serangkaian hasil pengujian.

Pengembang dapat menambahkan tindakan ke pipeline yang menyebarkan atau menguji aplikasi lebih lanjut setelah dibangun dan diuji untuk setiap perubahan.

Untuk informasi selengkapnya, lihat Bagaimana eksekusi pipa bekerja.

Bagaimana eksekusi pipa bekerja

Bagian ini memberikan gambaran umum tentang cara CodePipeline memproses serangkaian perubahan. CodePipelinemelacak setiap eksekusi pipeline yang dimulai saat pipeline dimulai secara manual atau perubahan dilakukan pada kode sumber. CodePipeline menggunakan mode eksekusi berikut untuk menangani cara setiap eksekusi berlangsung melalui pipeline. Untuk informasi selengkapnya, lihat Mengatur atau mengubah mode eksekusi pipa.

- Mode DIGANTIKAN: Eksekusi yang lebih baru dapat menyalip yang lebih lama. Ini adalah opsi default.

- Mode ANTRIAN: Eksekusi diproses satu per satu dalam urutan antrian. Ini membutuhkan tipe pipa V2.
- Paralel mode: Dalam mode PARALLEL, eksekusi berjalan secara bersamaan dan independen satu sama lain. Eksekusi tidak menunggu proses lain selesai sebelum memulai atau menyelesaikan. Ini membutuhkan tipe pipa V2.

> **Important**
>
> Untuk saluran pipa dalam mode PARALLEL, rollback panggung tidak tersedia. Demikian pula, kondisi kegagalan dengan jenis hasil rollback tidak dapat ditambahkan ke pipa mode PARALLEL.

Bagaimana eksekusi pipa dimulai

Anda dapat memulai eksekusi saat mengubah kode sumber atau memulai pipeline secara manual. Anda juga dapat memicu eksekusi melalui aturan Amazon CloudWatch Events yang Anda jadwalkan. Misalnya, ketika perubahan kode sumber didorong ke repositori yang dikonfigurasi sebagai tindakan sumber pipeline, pipeline mendeteksi perubahan dan memulai eksekusi.

> **Note**
>
> Jika pipeline berisi beberapa tindakan sumber, semuanya berjalan lagi, meskipun perubahan terdeteksi hanya untuk satu tindakan sumber.

Bagaimana revisi sumber diproses dalam eksekusi pipa

Untuk setiap eksekusi pipeline yang dimulai dengan perubahan kode sumber (revisi sumber), revisi sumber ditentukan sebagai berikut.

- Untuk saluran pipa dengan CodeCommit sumber, HEAD dikloning CodePipeline pada saat komit didorong. Misalnya, komit didorong, yang memulai pipeline untuk eksekusi 1. Pada saat komit kedua didorong, ini memulai pipeline untuk eksekusi 2.

> **Note**
>
> Untuk pipeline dalam mode PARALLEL dengan CodeCommit sumber, terlepas dari komit yang memicu eksekusi pipeline, aksi sumber akan selalu mengkloning HEAD pada saat dimulai. Untuk informasi selengkapnya, lihat CodeCommit atau revisi sumber S3 dalam mode PARALLEL mungkin tidak cocok dengan acara EventBridge.

- Untuk saluran pipa dengan sumber S3, EventBridge acara untuk pembaruan bucket S3 digunakan. Misalnya, peristiwa dihasilkan saat file diperbarui di bucket sumber, yang memulai pipeline untuk eksekusi 1. Pada saat acara untuk pembaruan bucket kedua dibuat, ini memulai pipeline untuk eksekusi 2.

> **Note**
>
> Untuk pipeline dalam mode PARALLEL dengan sumber S3, terlepas dari tag gambar yang memicu eksekusi, tindakan sumber akan selalu dimulai dengan tag gambar terbaru. Untuk informasi selengkapnya, lihat CodeCommit atau revisi sumber S3 dalam mode PARALLEL mungkin tidak cocok dengan acara EventBridge.

- Untuk pipeline dengan sumber koneksi, seperti ke Bitbucket, HEAD dikloning CodePipeline pada saat komit didorong. Misalnya, untuk pipeline dalam mode PARALLEL, komit didorong, yang memulai pipeline untuk eksekusi 1, dan eksekusi pipeline kedua menggunakan komit kedua.

Bagaimana penggantian sumber bekerja dengan transformator input EventBridge

Anda dapat menggunakan penggantian untuk memulai pipeline dengan ID revisi sumber tertentu yang Anda berikan untuk eksekusi pipeline. Misalnya, jika Anda ingin memulai pipeline yang akan memproses ID komit tertentu dari CodeCommit sumber Anda, Anda dapat menambahkan ID komit sebagai penggantian saat memulai pipeline.

Ada empat jenis revisi sumber untuk`revisionType`:

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

- `S3_OBJECT_KEY`

> **Note**
>
> Untuk `COMMIT_ID` dan `IMAGE_DIGEST` jenis revisi sumber, ID revisi sumber berlaku untuk semua konten dalam repositori, di semua cabang.

> **Note**
>
> Untuk `S3_OBJECT_VERSION_ID` dan `S3_OBJECT_KEY` jenis revisi sumber, salah satu tipe dapat digunakan secara independen, atau dapat digunakan bersama untuk mengganti sumber dengan spesifik ObjectKey dan versionId. Untuk`S3_OBJECT_KEY`, parameter konfigurasi `AllowOverrideForS3ObjectKey` perlu diatur ke`true`. Untuk informasi selengkapnya tentang parameter konfigurasi sumber S3, lihatParameter konfigurasi.

Anda dapat menentukan penggantian sumber menggunakan transformator input di. EventBridge Gunakan transformator input untuk meneruskan data sebagai salah satu dari berikut ini:

- Anda dapat menggunakan transformator input untuk meneruskan data sebagai parameter JSON.
- Anda dapat menggunakan transformator input untuk melewati variabel pipa.

Untuk contoh meneruskan data sebagai parameter JSON, lihatTindakan sumber dan sumber daya Amazon ECR EventBridge, Menghubungkan ke tindakan sumber Amazon S3 yang menggunakan dan EventBridge AWS CloudTrail untuk S3, dan CodeCommit tindakan sumber dan EventBridge untuk. CodeCommit

Bagaimana eksekusi pipa dihentikan

Untuk menggunakan konsol untuk menghentikan eksekusi pipeline, Anda dapat memilih Hentikan eksekusi di halaman visualisasi pipeline, di halaman riwayat eksekusi, atau di halaman riwayat terperinci. Untuk menggunakan CLI untuk menghentikan eksekusi pipeline, Anda menggunakan perintah. `stop-pipeline-execution` Untuk informasi selengkapnya, lihat Hentikan eksekusi pipeline di CodePipeline.

Ada dua cara untuk menghentikan eksekusi pipeline:

- Berhenti dan tunggu: Semua eksekusi tindakan yang sedang berlangsung diizinkan untuk diselesaikan, dan tindakan selanjutnya tidak dimulai. Eksekusi pipa tidak berlanjut ke tahap selanjutnya. Anda tidak dapat menggunakan opsi ini pada eksekusi yang sudah dalam `Stopping` keadaan.
- Berhenti dan tinggalkan: Semua eksekusi tindakan yang sedang berlangsung ditinggalkan dan tidak selesai, dan tindakan selanjutnya tidak dimulai. Eksekusi pipa tidak berlanjut ke tahap selanjutnya. Anda dapat menggunakan opsi ini pada eksekusi yang sudah dalam `Stopping` keadaan.

> **Note**
>
> Opsi ini dapat menyebabkan tugas gagal atau tugas di luar urutan.

Setiap opsi menghasilkan urutan pipa dan fase eksekusi tindakan yang berbeda, sebagai berikut.

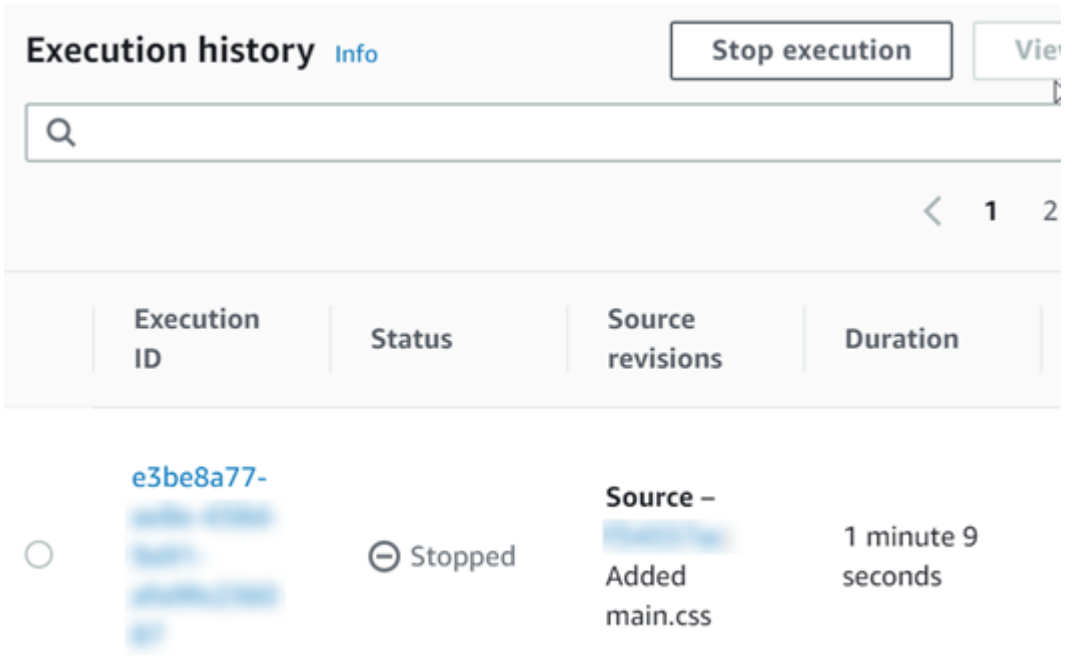
Opsi 1: Berhenti dan tunggu

Ketika Anda memilih untuk berhenti dan menunggu, eksekusi yang dipilih berlanjut hingga tindakan yang sedang berlangsung selesai. Misalnya, eksekusi pipeline berikut dihentikan saat aksi build sedang berlangsung.

1. Dalam tampilan pipeline, spanduk pesan sukses ditampilkan, dan tindakan build berlanjut hingga selesai. Status eksekusi pipeline adalah Berhenti.

   Dalam tampilan histori, status untuk tindakan yang sedang berlangsung, seperti tindakan build, sedang berlangsung hingga tindakan build selesai. Saat tindakan sedang berlangsung, status eksekusi pipeline adalah Berhenti.
2. Eksekusi berhenti ketika proses penghentian selesai. Jika tindakan build berhasil diselesaikan, statusnya Berhasil, dan eksekusi pipeline menunjukkan status Berhenti. Tindakan selanjutnya tidak dimulai. Tombol Coba lagi diaktifkan.

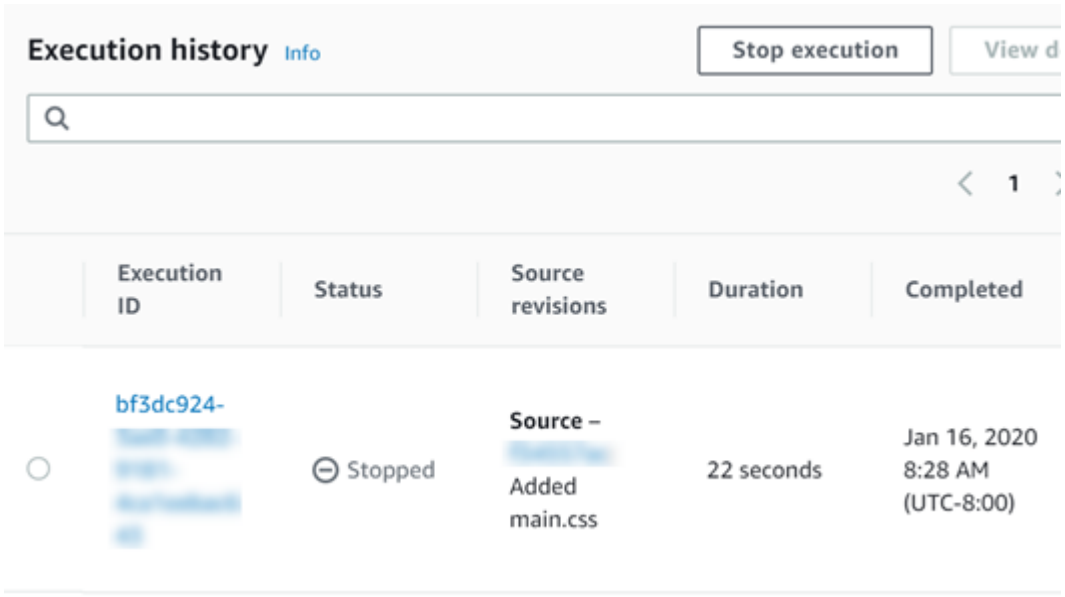
   Dalam tampilan riwayat, status eksekusi Dihentikan setelah tindakan yang sedang berlangsung selesai.



Opsi 2: Berhenti dan tinggalkan

Ketika Anda memilih untuk berhenti dan meninggalkan, eksekusi yang dipilih tidak menunggu tindakan yang sedang berlangsung selesai. Tindakan ditinggalkan. Misalnya, eksekusi pipeline berikut dihentikan dan ditinggalkan saat aksi build sedang berlangsung.

1. Dalam tampilan pipeline, pesan spanduk sukses ditampilkan, tindakan build menunjukkan status Sedang berlangsung, dan eksekusi pipeline menunjukkan status Berhenti.
2. Setelah eksekusi pipeline berhenti, aksi build menunjukkan status Abandoned, dan eksekusi pipeline menunjukkan status Stopped. Tindakan selanjutnya tidak dimulai. Tombol Coba lagi diaktifkan.
3. Dalam tampilan riwayat, status eksekusi Dihentikan.



Gunakan kasus untuk menghentikan eksekusi pipeline

Kami menyarankan Anda menggunakan opsi berhenti dan tunggu untuk menghentikan eksekusi pipeline. Opsi ini lebih aman karena menghindari kemungkinan kegagalan atau out-of-sequence tugas dalam pipeline Anda. Saat tindakan ditinggalkan CodePipeline, penyedia tindakan melanjutkan tugas apa pun yang terkait dengan tindakan tersebut. Dalam kasus tindakan, CloudFormation tindakan penerapan dalam pipeline ditinggalkan, tetapi pembaruan tumpukan mungkin berlanjut dan mengakibatkan pembaruan gagal.

Sebagai contoh tindakan terbengkalai yang dapat menghasilkan out-of-sequence tugas, jika Anda menerapkan file besar (1GB) melalui tindakan penerapan S3, dan Anda memilih untuk menghentikan dan meninggalkan tindakan saat penerapan sudah berlangsung, tindakan akan ditinggalkan di, tetapi berlanjut di CodePipeline Amazon S3. Amazon S3 tidak menemukan instruksi apa pun untuk membatalkan unggahan. Selanjutnya, jika Anda memulai eksekusi pipeline baru dengan file yang sangat kecil, sekarang ada dua penerapan yang sedang berlangsung. Karena ukuran file eksekusi baru kecil, penerapan baru selesai saat penerapan lama masih diunggah. Ketika penerapan lama selesai, file baru ditimpa oleh file lama.

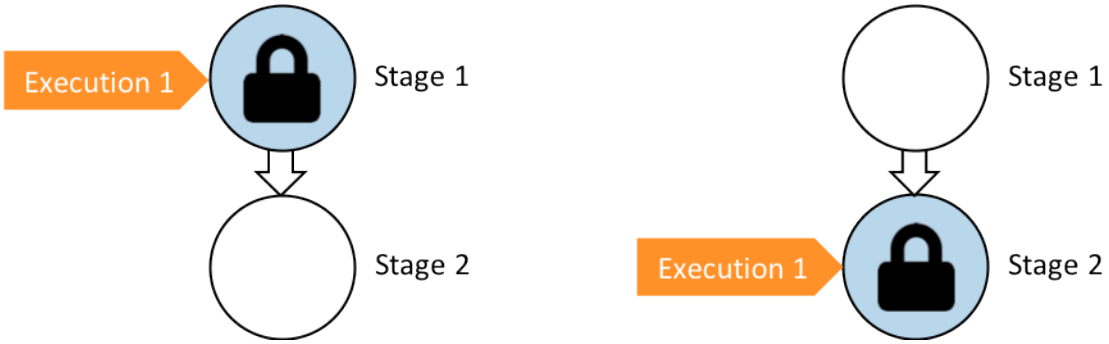
Anda mungkin ingin menggunakan opsi berhenti dan tinggalkan jika Anda memiliki tindakan khusus. Misalnya, Anda dapat meninggalkan tindakan kustom dengan pekerjaan yang tidak perlu diselesaikan sebelum memulai eksekusi baru untuk perbaikan bug.

Bagaimana eksekusi diproses dalam mode SUPERSEDED

Mode default untuk memproses eksekusi adalah mode SUPERSEDED. Eksekusi terdiri dari serangkaian perubahan yang diambil dan diproses oleh eksekusi. Pipelines dapat memproses beberapa eksekusi pada saat yang bersamaan. Setiap eksekusi dijalankan melalui pipa secara terpisah. Pipeline memproses setiap eksekusi secara berurutan dan mungkin menggantikan eksekusi sebelumnya dengan eksekusi berikutnya. Aturan berikut digunakan untuk memproses eksekusi dalam pipeline untuk mode SUPERSEDED.

Aturan 1: Tahapan dikunci saat eksekusi sedang diproses

Karena setiap tahap hanya dapat memproses satu eksekusi pada satu waktu, tahap terkunci saat sedang berlangsung. Ketika eksekusi menyelesaikan tahap, itu transisi ke tahap berikutnya dalam pipa.

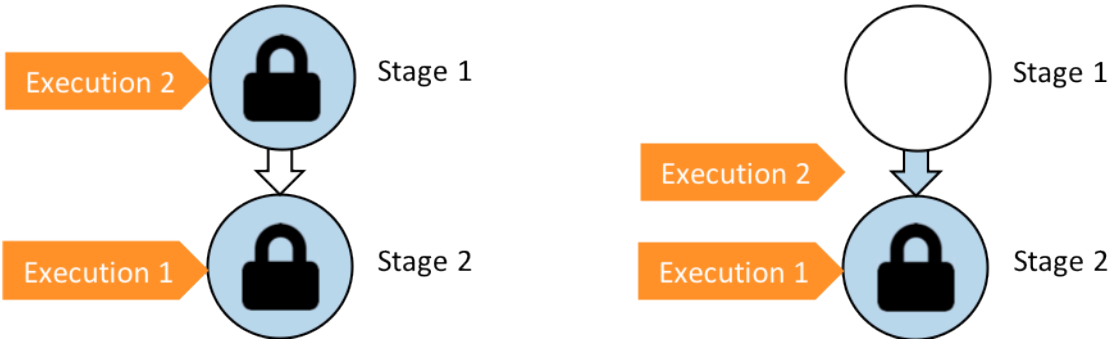


Sebelum: Stage 1 is locked as Execution 1 enters. Setelah: Stage 2 is locked as Execution 1 enters.

Aturan 2: Eksekusi selanjutnya menunggu panggung dibuka

Sementara panggung terkunci, eksekusi menunggu diadakan di depan panggung yang terkunci. Semua tindakan yang dikonfigurasi untuk suatu tahap harus diselesaikan dengan sukses sebelum tahap dianggap selesai. Kegagalan melepaskan kunci di atas panggung. Ketika eksekusi dihentikan, eksekusi tidak berlanjut dalam tahap dan tahap tidak terkunci.

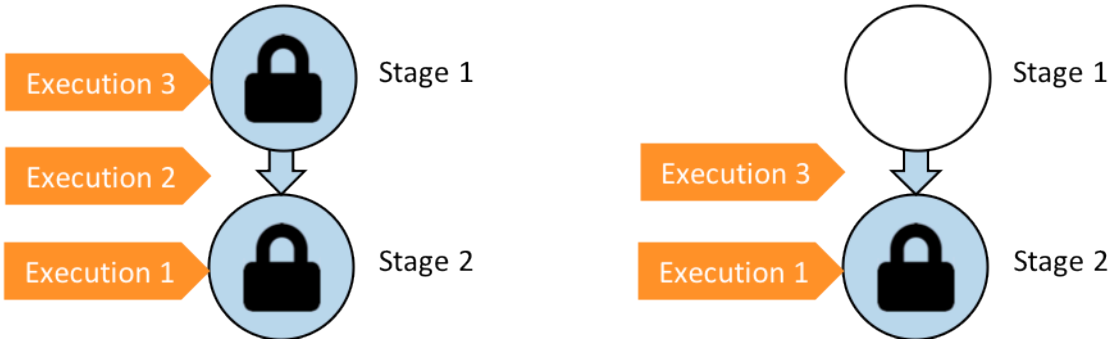
> **Note**
>
> Sebelum Anda menghentikan eksekusi, kami sarankan Anda menonaktifkan transisi di depan panggung. Dengan cara ini, ketika tahap dibuka karena eksekusi yang dihentikan, tahap tidak menerima eksekusi pipeline berikutnya.



Sebelum: Stage 2 is locked as Execution 1 enters. Setelah: Execution 2 exits Stage 1 and waits between stages.

Aturan 3: Eksekusi menunggu digantikan oleh eksekusi yang lebih baru

Eksekusi hanya digantikan di antara tahapan. Panggung terkunci memegang satu eksekusi di depan panggung menunggu panggung selesai. Eksekusi yang lebih baru menyusul eksekusi menunggu dan berlanjut ke tahap berikutnya segera setelah tahap dibuka. Eksekusi yang digantikan tidak berlanjut. Dalam contoh ini, Eksekusi 2 telah digantikan oleh Eksekusi 3 sambil menunggu tahap terkunci. Eksekusi 3 memasuki tahap berikutnya.



Sebelum: eksekusi 2 menunggu di antara tahapan sementara eksekusi 3 memasuki tahap 1. setelah: eksekusi 3 keluar tahap 1. eksekusi 2 digantikan oleh eksekusi 3.

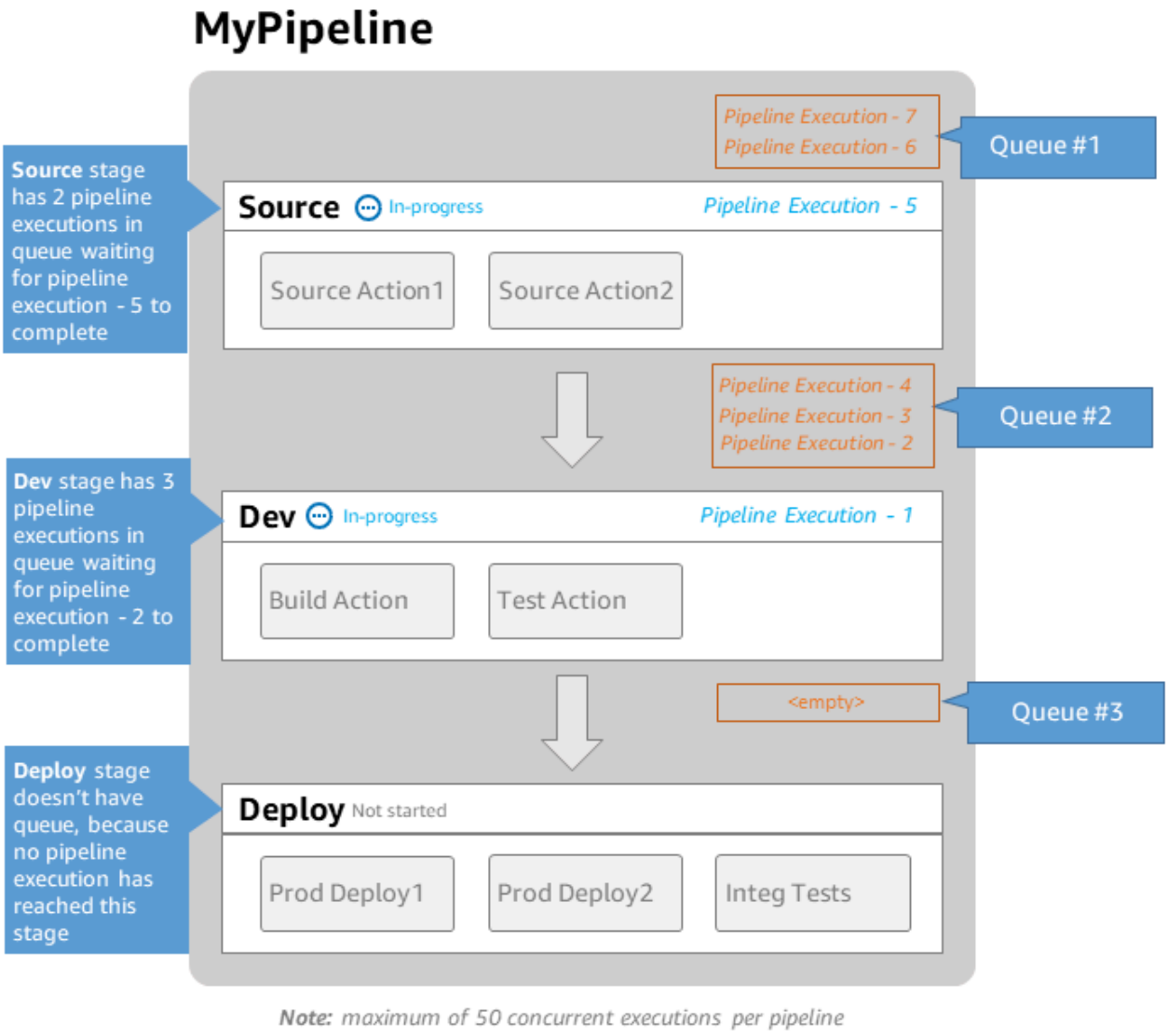
Untuk informasi selengkapnya tentang pertimbangan untuk melihat dan beralih di antara mode eksekusi, lihatMengatur atau mengubah mode eksekusi pipa. Untuk informasi selengkapnya tentang kuota dengan mode eksekusi, lihatKuota di AWS CodePipeline.

Bagaimana eksekusi diproses dalam mode ANTRIAN

Untuk pipeline dalam mode ANTRIAN, tahapan dikunci saat eksekusi sedang diproses; Namun, eksekusi menunggu tidak menyalip eksekusi yang sudah dimulai.

Eksekusi menunggu berkumpul di titik masuk ke tahapan terkunci dalam urutan mereka mencapai panggung, membentuk antrian eksekusi menunggu. Dengan mode ANTRIAN, Anda dapat memiliki beberapa antrian dalam pipeline yang sama. Ketika eksekusi antrian memasuki tahap, panggung terkunci dan tidak ada eksekusi lain yang bisa masuk. Perilaku ini tetap sama dengan mode SUPERSEDED. Ketika eksekusi selesai tahap, panggung menjadi tidak terkunci dan siap untuk eksekusi berikutnya.

Diagram berikut menunjukkan bagaimana tahapan dalam proses eksekusi proses pipa mode ANTRIAN. Misalnya, saat tahap Sumber memproses eksekusi 5, eksekusi untuk 6 dan 7 membentuk Antrian #1 dan menunggu di titik masuk tahap. Eksekusi berikutnya dalam antrian akan diproses setelah tahap terbuka.



Untuk informasi selengkapnya tentang pertimbangan untuk melihat dan beralih di antara mode eksekusi, lihatMengatur atau mengubah mode eksekusi pipa. Untuk informasi selengkapnya tentang kuota dengan mode eksekusi, lihatKuota di AWS CodePipeline.

Bagaimana eksekusi diproses dalam mode PARALLEL

Untuk pipeline dalam mode PARALLEL, eksekusi independen satu sama lain dan jangan menunggu eksekusi lain selesai sebelum memulai. Tidak ada antrian. Untuk melihat eksekusi paralel di pipeline, gunakan tampilan riwayat eksekusi.

Gunakan mode PARALLEL di lingkungan pengembangan di mana setiap fitur memiliki cabang fitur sendiri dan menyebarkan ke target yang tidak dibagikan oleh pengguna lain.

Untuk informasi selengkapnya tentang pertimbangan untuk melihat dan beralih di antara mode eksekusi, lihatMengatur atau mengubah mode eksekusi pipa. Untuk informasi selengkapnya tentang kuota dengan mode eksekusi, lihatKuota di AWS CodePipeline.

Mengelola Aliran Pipa

Aliran eksekusi pipa dapat dikontrol oleh:

- Transisi, yang mengontrol aliran eksekusi ke panggung. Transisi dapat diaktifkan atau dinonaktifkan. Ketika transisi dinonaktifkan, eksekusi pipa tidak dapat memasuki tahap. Eksekusi pipeline yang menunggu untuk memasuki tahap di mana transisi dinonaktifkan disebut eksekusi masuk. Setelah Anda mengaktifkan transisi, eksekusi masuk bergerak ke panggung dan menguncinya.

  Mirip dengan eksekusi yang menunggu tahap terkunci, ketika transisi dinonaktifkan, eksekusi yang menunggu untuk memasuki tahap masih dapat digantikan oleh eksekusi baru. Ketika transisi yang dinonaktifkan diaktifkan kembali, eksekusi terbaru, termasuk semua yang menggantikan eksekusi lama saat transisi dinonaktifkan, memasuki tahap.
- Tindakan persetujuan, yang mencegah pipeline dari transisi ke tindakan berikutnya sampai izin diberikan (misalnya, melalui persetujuan manual dari identitas resmi). Anda dapat menggunakan tindakan persetujuan ketika Anda ingin mengontrol waktu transisi pipeline ke tahap Produksi akhir, misalnya.

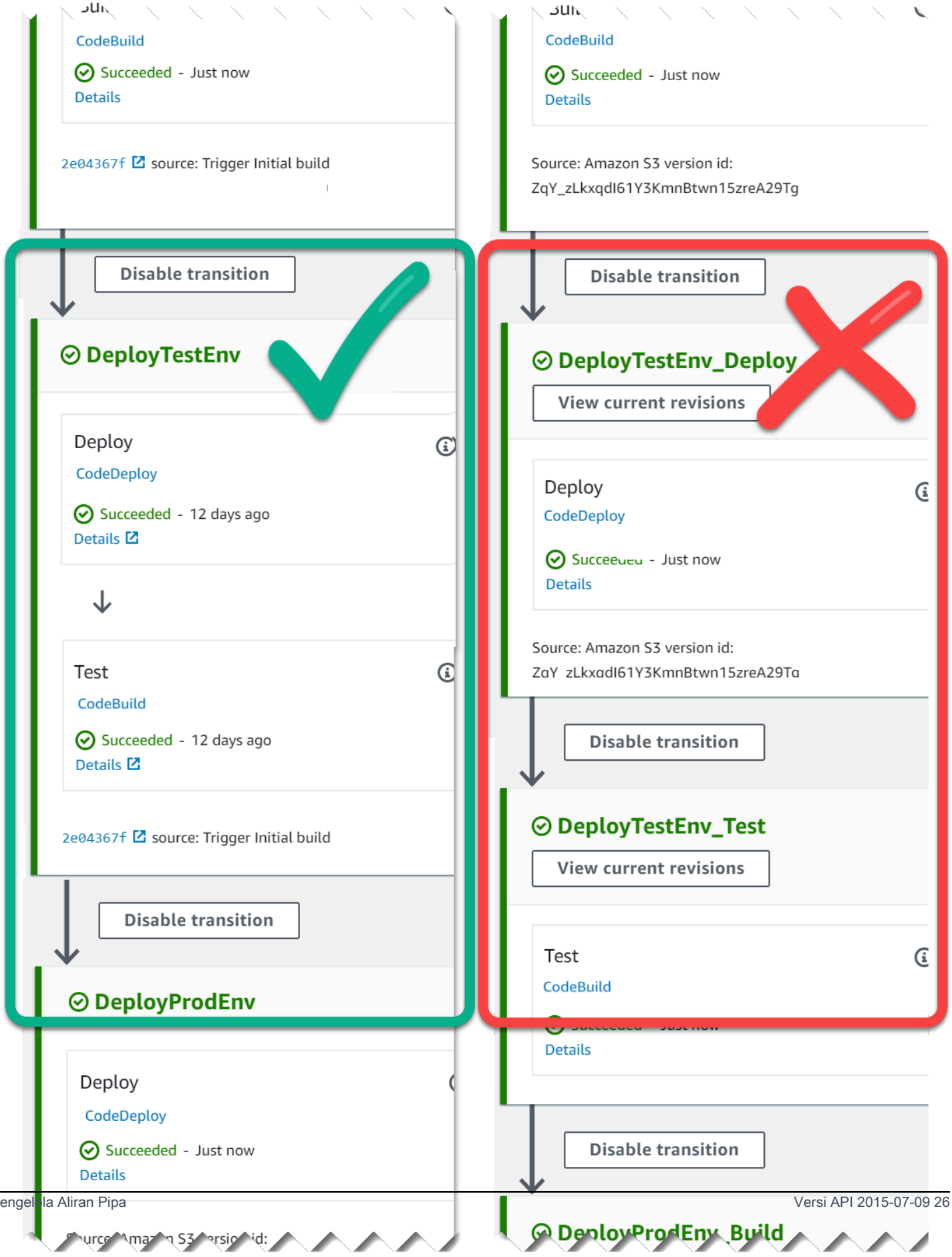
> **Note**
>
> Tahap dengan tindakan persetujuan dikunci sampai tindakan persetujuan disetujui atau ditolak atau telah habis waktu. Tindakan persetujuan waktu habis diproses dengan cara yang sama seperti tindakan yang gagal.

- Kegagalan, ketika suatu tindakan dalam suatu tahap tidak berhasil diselesaikan. Revisi tidak beralih ke tindakan berikutnya dalam tahap atau tahap selanjutnya dalam pipa. Berikut ini dapat terjadi:
  - Anda secara manual mencoba lagi tahap yang berisi tindakan gagal. Ini melanjutkan eksekusi (mencoba kembali tindakan yang gagal dan, jika berhasil, berlanjut di tahap/pipa).
  - Eksekusi lain memasuki tahap gagal dan menggantikan eksekusi yang gagal. Pada titik ini, eksekusi yang gagal tidak dapat dicoba lagi.

Struktur pipa yang direkomendasikan

Saat memutuskan bagaimana perubahan kode harus mengalir melalui pipeline Anda, yang terbaik adalah mengelompokkan tindakan terkait dalam satu tahap sehingga, ketika tahap terkunci, semua tindakan memproses eksekusi yang sama. Anda dapat membuat panggung untuk setiap lingkungan aplikasi Wilayah AWS, atau Availability Zone, dan sebagainya. Pipa dengan terlalu banyak tahapan (yaitu, terlalu granular) dapat memungkinkan terlalu banyak perubahan bersamaan, sementara pipa dengan banyak tindakan dalam tahap besar (terlalu kasar) dapat memakan waktu terlalu lama untuk melepaskan perubahan.

Sebagai contoh, tindakan pengujian setelah tindakan penerapan di tahap yang sama dijamin untuk menguji perubahan yang sama yang diterapkan. Dalam contoh ini, perubahan diterapkan ke lingkungan Pengujian dan kemudian diuji, dan kemudian perubahan terbaru dari lingkungan pengujian diterapkan ke lingkungan Produksi. Dalam contoh yang direkomendasikan, lingkungan Test dan lingkungan Prod adalah tahapan yang terpisah.


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

Kiri: pengujian terkait, penerapan, dan tindakan persetujuan dikelompokkan bersama (disarankan). Kanan: tindakan terkait dalam tahap terpisah (tidak disarankan).

Bagaimana Eksekusi Inbound Bekerja

Eksekusi masuk adalah eksekusi yang menunggu tahap, transisi, atau tindakan yang tidak tersedia tersedia sebelum bergerak maju. Tahap, transisi, atau tindakan berikutnya mungkin tidak tersedia karena:

- Eksekusi lain telah memasuki tahap berikutnya dan menguncinya.
- Transisi untuk memasuki tahap berikutnya dinonaktifkan.

Anda dapat menonaktifkan transisi untuk menahan eksekusi masuk jika Anda ingin mengontrol apakah eksekusi saat ini memiliki waktu untuk diselesaikan pada tahap berikutnya, atau jika Anda ingin menghentikan semua tindakan pada titik tertentu. Untuk menentukan apakah Anda memiliki eksekusi masuk, Anda dapat melihat pipeline di konsol atau melihat output dari get-pipeline-state perintah.

Eksekusi masuk beroperasi dengan pertimbangan berikut:

- Segera setelah aksi, transisi, atau tahap terkunci tersedia, eksekusi inbound yang sedang berlangsung memasuki tahap dan berlanjut melalui pipeline.
- Sementara eksekusi inbound menunggu, itu dapat dihentikan secara manual. Eksekusi inbound dapat memiliki`InProgress`,`Stopped`, atau `Failed` status.
- Ketika eksekusi masuk telah dihentikan atau gagal, itu tidak dapat dicoba lagi karena tidak ada tindakan yang gagal untuk dicoba lagi. Ketika eksekusi inbound telah dihentikan, dan transisi diaktifkan, eksekusi inbound yang dihentikan tidak berlanjut ke tahap.

Anda dapat melihat atau menghentikan eksekusi masuk.

Artefak input dan output

CodePipeline terintegrasi dengan alat pengembangan untuk memeriksa perubahan kode dan kemudian membangun dan menyebarkan melalui semua tahapan proses pengiriman berkelanjutan. Artefak adalah file yang dikerjakan oleh tindakan dalam pipeline, seperti file atau folder dengan kode aplikasi, file halaman indeks, skrip, dan sebagainya. Misalnya, artefak aksi sumber Amazon S3 adalah nama file (atau jalur file) tempat file kode sumber aplikasi disediakan

untuk tindakan sumber pipeline. File disediakan sebagai file ZIP, seperti contoh nama artefak berikut:`SampleApp_Windows.zip`. Artefak keluaran untuk aksi sumber, file kode sumber aplikasi, adalah artefak keluaran dari tindakan itu dan merupakan artefak input untuk tindakan selanjutnya, seperti tindakan build. Sebagai contoh lain, tindakan build mungkin menjalankan perintah build yang mengkompilasi kode sumber aplikasi untuk artefak input, yang merupakan file kode sumber aplikasi. Lihat halaman referensi konfigurasi tindakan untuk tindakan tertentu untuk detail tentang parameter artefak, seperti AWS CodeBuild membangun dan menguji referensi tindakan untuk CodeBuild tindakan.

Tindakan menggunakan artefak input dan output yang disimpan di bucket artefak Amazon S3 yang Anda pilih saat membuat pipeline. CodePipeline zip dan transfer file untuk artefak input atau output yang sesuai untuk jenis tindakan di panggung.

> **Note**
>
> Bucket artefak bukanlah bucket yang sama dengan bucket yang digunakan sebagai lokasi file sumber untuk pipeline di mana aksi sumber yang dipilih adalah S3.

Contoh:

1. CodePipeline memicu pipeline Anda untuk berjalan ketika ada komit ke repositori sumber, menyediakan artefak keluaran (file apa pun yang akan dibuat) dari tahap Sumber.
2. Artefak keluaran (file apa pun yang akan dibuat) dari langkah sebelumnya dicerna sebagai artefak input ke tahap Build. Artefak keluaran (aplikasi yang dibangun) dari tahap Build dapat berupa aplikasi yang diperbarui atau image Docker yang diperbarui yang dibuat ke wadah.
3. Artefak keluaran dari langkah sebelumnya (aplikasi yang dibangun) dicerna sebagai artefak input ke tahap Deploy, seperti pementasan atau lingkungan produksi di. AWS Cloud Anda dapat menyebarkan aplikasi ke armada penyebaran, atau Anda dapat menyebarkan aplikasi berbasis kontainer ke tugas yang berjalan di kluster ECS.

Saat membuat atau mengedit tindakan, Anda menetapkan artefak atau artefak input dan keluaran untuk tindakan tersebut. Misalnya, untuk pipeline dua tahap dengan tahap Sumber dan Penerapan, di Edit Tindakan, Anda memilih nama artefak aksi sumber untuk artefak input untuk tindakan penerapan.

- Saat Anda menggunakan konsol untuk membuat pipeline pertama, CodePipeline buat bucket Amazon S3 yang sama Akun AWS dan Wilayah AWS untuk menyimpan item untuk semua pipeline. Setiap kali Anda menggunakan konsol untuk membuat pipeline lain di Wilayah itu, CodePipeline buat folder untuk pipeline itu di bucket. Ini menggunakan folder itu untuk menyimpan artefak untuk pipeline Anda saat proses rilis otomatis berjalan. Bucket ini diberi nama codepipeline-*region*-*12345EXAMPLE*, di mana *region* adalah AWS Wilayah tempat Anda membuat pipeline, dan *12345EXAMPLE* merupakan nomor acak 12 digit yang memastikan nama bucket unik.

  > **Note**
  >
  > Jika Anda sudah memiliki bucket yang dimulai dengan codepipeline- region - di Wilayah tempat Anda membuat pipeline, CodePipeline gunakan itu sebagai bucket default. Ini juga mengikuti urutan leksikografis; misalnya, codepipeline- region-abcexample dipilih sebelum codepipeline- region-defexample.

CodePipeline memotong nama artefak, yang dapat menyebabkan beberapa nama bucket tampak serupa. Meskipun nama artefak tampaknya terpotong, CodePipeline peta ke ember artefak dengan cara yang tidak terpengaruh oleh artefak dengan nama terpotong. Pipa dapat berfungsi secara normal. Ini bukan masalah dengan folder atau artefak. Ada batas 100 karakter untuk nama pipeline. Meskipun nama folder artefak mungkin tampak dipersingkat, itu masih unik untuk pipeline Anda.

Saat membuat atau mengedit pipeline, Anda harus memiliki bucket artefak di pipeline Akun AWS dan Wilayah AWS, dan Anda harus memiliki satu bucket artefak per Wilayah tempat Anda berencana untuk menjalankan suatu tindakan. Jika Anda menggunakan konsol untuk membuat pipeline atau tindakan lintas wilayah, bucket artefak default dikonfigurasi oleh CodePipeline di Wilayah tempat Anda memiliki tindakan.

Jika Anda menggunakan AWS CLI untuk membuat pipeline, Anda dapat menyimpan artefak untuk pipeline itu di bucket Amazon S3 mana pun selama bucket itu Akun AWS sama Wilayah AWS dan sebagai pipeline. Anda dapat melakukan ini jika Anda khawatir tentang melampaui batas ember Amazon S3 yang diizinkan untuk akun Anda. Jika Anda menggunakan AWS CLI untuk membuat atau mengedit pipeline, dan menambahkan tindakan Lintas wilayah (tindakan dengan AWS penyedia di Wilayah yang berbeda dari pipeline Anda), Anda harus menyediakan bucket artefak untuk setiap Wilayah tambahan tempat Anda berencana untuk menjalankan tindakan.

- Setiap tindakan memiliki tipe. Bergantung pada jenisnya, tindakan mungkin memiliki satu atau kedua hal berikut:
  - Artefak input, yang merupakan artefak yang dikonsumsi atau dikerjakan selama aksi dijalankan.
  - Artefak keluaran, yang merupakan output dari tindakan.

  Setiap artefak keluaran dalam pipa harus memiliki nama yang unik. Setiap artefak masukan untuk suatu tindakan harus sesuai dengan artefak keluaran dari suatu tindakan sebelumnya di pipeline, apakah tindakan itu segera sebelum aksi dalam tahap atau berjalan dalam tahap beberapa tahap sebelumnya.

  Artefak dapat dikerjakan dengan lebih dari satu tindakan.

Bagaimana kondisi panggung bekerja?

Untuk setiap kondisi yang menentukan aturan, aturan dijalankan. Jika kondisinya gagal, hasilnya terlibat. Tahap melakukan hasil yang ditentukan hanya ketika kondisi gagal. Secara opsional, sebagai bagian dari aturan, Anda juga menentukan sumber daya mana yang CodePipeline harus digunakan untuk kasus-kasus tertentu. Misalnya, `CloudWatchAlarm` aturan akan menggunakan sumber daya CloudWatch alarm untuk menjalankan pemeriksaan untuk kondisi tersebut.

Suatu kondisi mungkin cocok dengan beberapa aturan, dan setiap aturan dapat menentukan salah satu dari tiga penyedia.

Aliran tingkat tinggi untuk menciptakan kondisi sebagai berikut.

1. Pilih jenis kondisi dari jenis kondisi yang tersedia di CodePipeline. Misalnya, gunakan tipe kondisi On Success untuk menyiapkan tahap sehingga setelah tahap berhasil, seperangkat aturan dapat digunakan untuk menjalankan pemeriksaan sebelum melanjutkan.
2. Pilih aturannya. Misalnya, `CloudWatchAlarm` aturan akan memeriksa alarm dan menggunakan EB untuk memeriksa ambang alarm yang telah dikonfigurasi sebelumnya. Jika cek berhasil, dan alarm di bawah ambang batas, tahap dapat dilanjutkan.
3. Konfigurasikan hasilnya, seperti rollback yang akan digunakan jika aturan gagal.

Kondisi digunakan untuk jenis ekspresi tertentu dan masing-masing memiliki opsi khusus untuk hasil yang tersedia sebagai berikut:

- Entri - Ketentuan untuk melakukan pemeriksaan yang, jika terpenuhi, memungkinkan masuk ke suatu panggung. Aturan terlibat dengan opsi hasil berikut: Gagal atau Lewati
- Pada Kegagalan - Kondisi untuk melakukan pemeriksaan untuk tahap ketika gagal. Aturan terlibat dengan opsi hasil berikut: Rollback
- Tentang Sukses - Kondisi untuk melakukan pemeriksaan untuk tahap ketika berhasil. Aturan terlibat dengan opsi hasil berikut: Rollback atau Gagal

Diagram berikut menunjukkan aliran contoh untuk tipe kondisi Entri di CodePipeline. Kondisi menjawab pertanyaan, Apa yang harus terjadi jika kondisi tidak terpenuhi, artinya aturan apa pun gagal? Dalam alur berikut, kondisi Entri dikonfigurasi dengan LambdaInvoke aturan dan `CloudWatchAlarm` aturan. Jika aturan gagal, maka hasil yang dikonfigurasi, seperti Fail, terlibat.

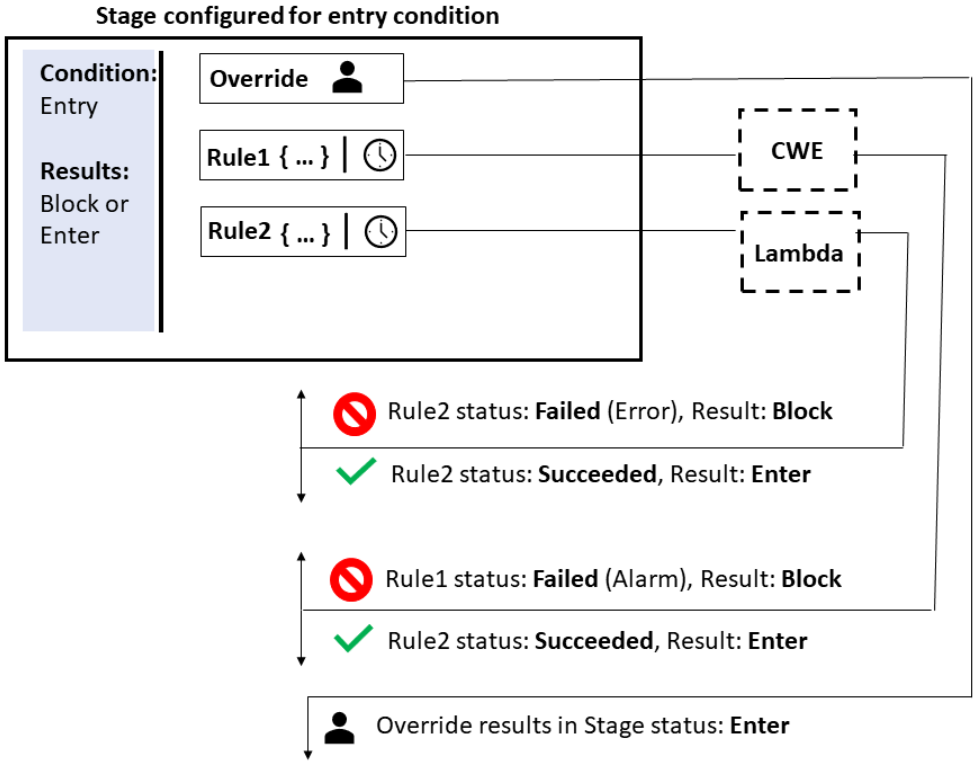


Diagram berikut menunjukkan aliran contoh untuk kondisi On Failure tipe in CodePipeline. Kondisi menjawab pertanyaan, Apa yang harus terjadi jika kondisi terpenuhi, artinya aturan semua berhasil cek mereka? Dalam alur berikut, kondisi On Failure dikonfigurasi dengan LambdaInvoke aturan dan `CloudWatchAlarm` aturan. Jika aturan berhasil, maka hasil yang dikonfigurasi, seperti Fail, terlibat.

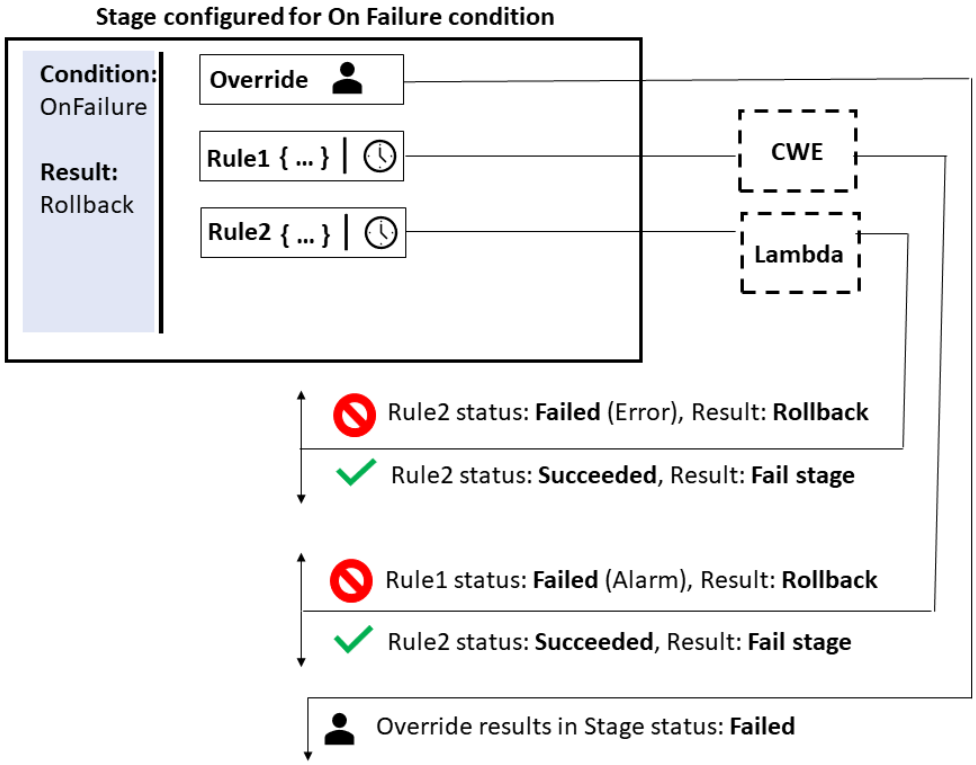
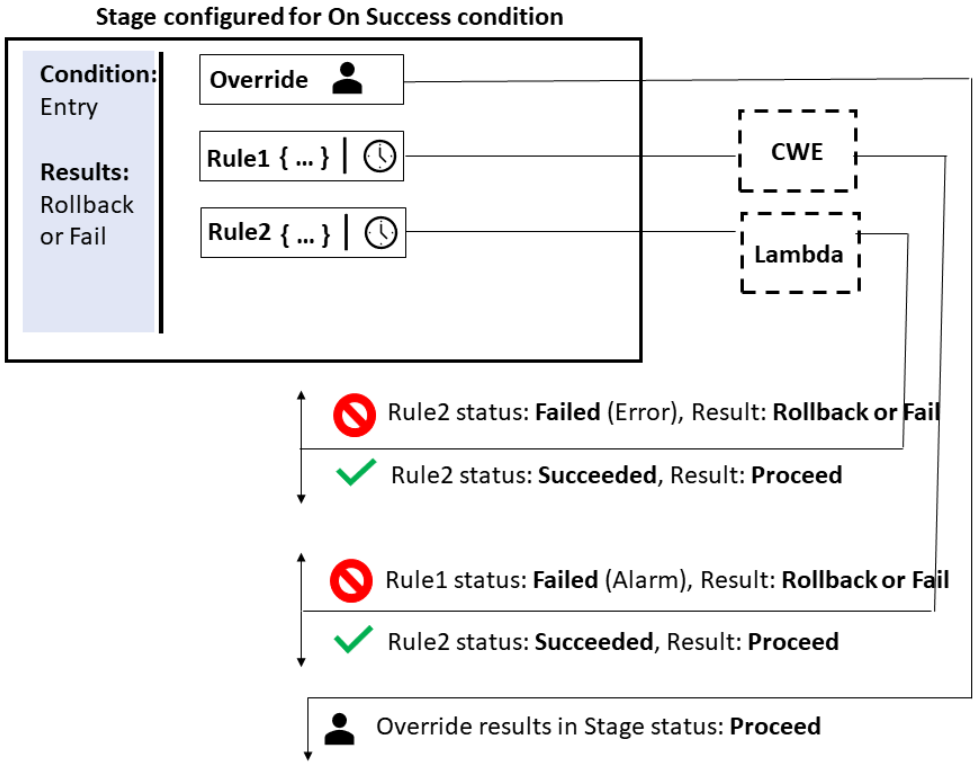


Diagram berikut menunjukkan aliran contoh untuk kondisi On Success tipe in CodePipeline. Kondisi menjawab pertanyaan, Apa yang harus terjadi jika kondisi terpenuhi, artinya aturan semua berhasil cek mereka? Dalam alur berikut, kondisi On Success dikonfigurasi dengan `LambdaInvoke` aturan dan `CloudWatchAlarm` aturan. Jika aturan berhasil, maka hasil yang dikonfigurasi, seperti Fail, terlibat.



Aturan untuk kondisi panggung

Saat Anda mengonfigurasi kondisi tahap, Anda memilih dari aturan yang telah ditentukan sebelumnya dan menentukan hasil untuk aturan tersebut. Status kondisi akan Gagal jika salah satu aturan dalam kondisi gagal dan Berhasil jika semua aturan berhasil. Bagaimana kriteria dipenuhi untuk kondisi On Failure dan On Success tergantung pada jenis aturan.

Berikut ini adalah aturan terkelola yang dapat Anda tambahkan ke kondisi panggung.

- Kondisi dapat menggunakan aturan Command untuk menentukan perintah untuk memenuhi kriteria aturan untuk kondisi. Untuk informasi lebih lanjut tentang aturan ini, lihatCommands.
- Kondisi dapat menggunakan AWS DeploymentWindowaturan untuk menentukan waktu penerapan yang disetujui untuk mengizinkan penerapan. Kriteria aturan akan diukur dengan ekspresi cron yang disediakan untuk jendela penerapan. Aturan berhasil ketika tanggal dan waktu di jendela penerapan memenuhi kriteria dalam ekspresi cron untuk aturan tersebut. Untuk informasi lebih lanjut tentang aturan ini, lihatDeploymentWindow.
- Kondisi dapat menggunakan aturan AWS Lambda untuk memeriksa status kesalahan yang dikembalikan dari fungsi Lambda yang dikonfigurasi. Aturan terpenuhi ketika cek menerima

hasil fungsi Lambda. Kesalahan dari fungsi Lambda memenuhi kriteria kondisi On Failure. Untuk informasi lebih lanjut tentang aturan ini, lihatLambdaInvoke.

- Kondisi dapat menggunakan AWS CloudWatchAlarmaturan untuk memeriksa alarm yang dikonfigurasi dari CloudWatch peristiwa. Aturan terpenuhi ketika cek mengembalikan status alarm OK, ALARM, atau INSUFF_DATA. Untuk kondisi On Success, OK dan INSUFFICIENT_DATA memenuhi kriteria. ALARM memenuhi kriteria kondisi On Failure. Untuk informasi lebih lanjut tentang aturan ini, lihatCloudWatchAlarm.
- Kondisi dapat menggunakan VariableCheckaturan untuk membuat kondisi di mana variabel output diperiksa terhadap ekspresi yang disediakan. Aturan melewati pemeriksaan ketika nilai variabel memenuhi kriteria aturan, seperti nilai yang sama atau lebih besar dari variabel output tertentu. Untuk informasi lebih lanjut tentang aturan ini, lihatVariableCheck.

Jenis pipa

CodePipeline menyediakan jenis pipa berikut, yang berbeda dalam karakteristik dan harga, sehingga Anda dapat menyesuaikan fitur pipa dan biaya dengan kebutuhan aplikasi Anda.

- Pipa tipe V1 memiliki struktur JSON yang berisi pipa standar, tahap, dan parameter tingkat aksi.
- Pipa tipe V2 memiliki struktur yang sama dengan tipe V1, bersama dengan parameter tambahan untuk keamanan rilis dan konfigurasi pemicu.

Untuk informasi tentang harga CodePipeline, lihat Harga.

Lihat CodePipeline referensi struktur pipa halaman untuk detail tentang parameter di setiap jenis pipeline. Untuk informasi tentang jenis pipa yang harus dipilih, lihatJenis pipa apa yang tepat untuk saya?.

Jenis pipa apa yang tepat untuk saya?

Jenis pipa ditentukan oleh serangkaian karakteristik dan fitur yang didukung oleh setiap versi pipa.

Berikut ini adalah ringkasan kasus penggunaan dan karakteristik yang tersedia untuk setiap jenis pipa.

	Tipe V1	Jenis V2
Karakteristik		
Kasus penggunaan	• Penerapan standar	• Penerapan dengan konfigurasi dari melewati variabel tingkat pipeline saat runtime • Penerapan di mana pipeline dikonfigurasi untuk memulai pada tag Git
Variabel tingkat tindakan	Didukung	Didukung
Mode eksekusi PARALEL	Tidak didukung	Didukung
Variabel tingkat pipa	Tidak didukung	Didukung
Mode eksekusi ANTRIAN	Tidak didukung	Didukung
Rollback untuk tahapan pipa	Tidak didukung	Didukung
Pengesampingan revisi sumber	Tidak didukung	Didukung
Kondisi panggung	Tidak didukung	Didukung
Memicu dan memfilter tag Git, permintaan tarik, cabang, atau jalur file	Tidak didukung	Didukung
Tindakan Perintah	Tidak didukung	Didukung
Membuat kondisi Entri dengan hasil Lewati	Tidak didukung	Didukung
Konfigurasikan tahapan untuk coba lagi otomatis pada kegagalan	Tidak didukung	Didukung

Untuk informasi tentang harga CodePipeline, lihat Harga.

Anda dapat membuat dan menjalankan skrip Python untuk membantu Anda menganalisis potensi biaya pemindahan pipeline tipe V1 ke pipeline tipe V2.

> **Note**
>
> Contoh skrip di bawah ini dimaksudkan untuk tujuan demonstrasi dan evaluasi saja. Ini bukan alat penawaran dan tidak menjamin biaya untuk penggunaan sebenarnya dari pipa tipe V2, dan itu tidak termasuk pajak apa pun yang mungkin berlaku. Untuk informasi tentang harga CodePipeline, lihat Harga.

Untuk membuat dan menjalankan skrip untuk membantu Anda mengevaluasi biaya pemindahan pipeline tipe V1 ke pipeline tipe V2

1. Unduh dan instal python.
2. Buka jendela terminal. Jalankan perintah berikut untuk membuat skrip python baru bernama PipelineCostAnalyzer.py.

   ```
   vi PipelineCostAnalyzer.py
   ```

3. Salin dan tempel kode berikut ke dalam PipelineCostAnalyzerskrip.py.

   ```
   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. Dari terminal atau command prompt, ubah direktori ke tempat Anda membuat skrip penganalisis.

   Dari direktori itu, jalankan perintah berikut, di mana wilayah adalah Wilayah AWS tempat Anda membuat pipeline V1 yang ingin Anda analisis. Anda juga dapat mengevaluasi pipeline tertentu secara opsional dengan memberikan namanya:

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

   Misalnya, jalankan perintah berikut untuk menjalankan skrip python bernama PipelineCostAnalyzer.py. Dalam contoh ini, Wilayah adalah`us-west-2`.

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

   > **Note**
   >
   > Skrip ini akan menganalisis semua pipeline V1 dalam yang ditentukan Wilayah AWS kecuali Anda menentukan nama pipeline tertentu.

5. Dalam contoh keluaran berikut dari skrip, kita dapat melihat daftar eksekusi tindakan, daftar eksekusi tindakan yang memenuhi syarat untuk penagihan, total runtime eksekusi tindakan ini, dan perkiraan biaya tindakan ini seperti yang dilakukan dalam 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
   ```

Dalam contoh ini, nilai negatif di baris terakhir mewakili perkiraan jumlah yang mungkin disimpan dengan pindah ke pipeline tipe V2.

> **Note**
>
> Output skrip dan contoh terkait yang menunjukkan biaya dan informasi lainnya hanya perkiraan. Mereka dimaksudkan untuk tujuan demonstrasi dan evaluasi saja dan tidak menjamin penghematan aktual. Untuk informasi tentang harga CodePipeline, lihat Harga.

Memulai dengan CodePipeline

Jika Anda baru mengenal CodePipeline, Anda dapat mengikuti tutorial dalam panduan ini setelah mengikuti langkah-langkah di bagian ini untuk mengatur.

CodePipeline Konsol menyertakan informasi bermanfaat dalam panel yang dapat dilipat yang dapat Anda buka dari ikon informasi atau tautan Info apa pun di halaman. ( ). Anda dapat menutup panel ini kapan saja.

CodePipeline Konsol juga menyediakan cara untuk mencari sumber daya Anda dengan cepat, seperti repositori, membangun proyek, aplikasi penerapan, dan saluran pipa. Pilih Pergi ke sumber daya atau tekan tombol /, dan kemudian ketik nama sumber daya. Setiap kecocokan akan muncul di daftar. Pencarian peka huruf besar/kecil. Anda hanya melihat sumber daya yang izin untuk menampilkannya Anda memiliki. Untuk informasi selengkapnya, lihat Menampilkan sumber daya di konsol.

Sebelum Anda dapat menggunakan AWS CodePipeline untuk pertama kalinya, Anda harus membuat Akun AWS dan membuat pengguna administratif pertama Anda.

Topik

- Langkah 1: Buat pengguna Akun AWS dan administratif
- Langkah 2: Menerapkan kebijakan terkelola untuk akses administratif CodePipeline
- Langkah 3: Instal AWS CLI
- Langkah 4: Buka konsol untuk CodePipeline
- Langkah selanjutnya

Langkah 1: Buat pengguna Akun AWS dan administratif

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/pendaftaran.

2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon atau pesan teks dan memasukkan kode verifikasi pada keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Kapan saja, Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan masuk https://aws.amazon.comke/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke Konsol Manajemen AWSsebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .
2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.
2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Langkah 2: Menerapkan kebijakan terkelola untuk akses administratif CodePipeline

Anda harus memberikan izin untuk berinteraksi CodePipeline. Cara tercepat untuk melakukan ini adalah dengan menerapkan kebijakan `AWSCodePipeline_FullAccess` terkelola kepada pengguna administratif.

> **Note**
>
> `AWSCodePipeline_FullAccess`Kebijakan ini mencakup izin yang memungkinkan pengguna konsol meneruskan peran IAM ke CodePipeline atau lainnya. Layanan AWS Ini memungkinkan layanan untuk mengambil peran dan melakukan tindakan atas nama Anda. Saat Anda melampirkan kebijakan ke pengguna, peran, atau grup, `iam:PassRole` izin akan diterapkan. Pastikan kebijakan tersebut hanya diterapkan pada pengguna tepercaya. Saat

pengguna dengan izin ini menggunakan konsol untuk membuat atau mengedit pipeline, pilihan berikut tersedia:

- Buat peran CodePipeline layanan atau pilih yang sudah ada dan berikan peran ke CodePipeline
- Mungkin memilih untuk membuat aturan CloudWatch Peristiwa untuk deteksi perubahan dan meneruskan peran layanan CloudWatch Acara ke CloudWatch Acara

Untuk informasi selengkapnya, lihat Memberikan izin pengguna untuk meneruskan peran ke peran. Layanan AWS

> **Note**
>
> `AWSCodePipeline_FullAccess`Kebijakan ini menyediakan akses ke semua CodePipeline tindakan dan sumber daya yang dapat diakses oleh pengguna IAM, serta semua tindakan yang mungkin saat membuat tahapan dalam pipeline, seperti membuat tahapan yang mencakup, Elastic Beanstalk CodeDeploy, atau Amazon S3. Sebagai praktik terbaik, Anda harus memberi izin kepada individu saja yang mereka butuhkan untuk menjalankan tugasnya. Untuk informasi selengkapnya tentang cara membatasi pengguna IAM pada serangkaian CodePipeline tindakan dan sumber daya terbatas, lihat. Hapus izin dari peran CodePipeline layanan

Untuk memberikan akses dan menambahkan izin bagi pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti instruksi di Buat rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center .
- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti instruksi dalam Buat peran untuk penyedia identitas pihak ketiga (federasi) dalam Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Buat peran untuk pengguna IAM dalam Panduan Pengguna IAM.

- (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

Langkah 3: Instal AWS CLI

Untuk memanggil CodePipeline perintah dari AWS CLI pada mesin pengembangan lokal, Anda harus menginstal CLI AWS . Langkah ini opsional jika Anda berniat untuk mulai hanya menggunakan langkah-langkah dalam panduan ini untuk CodePipeline konsol.

Untuk menginstal dan mengkonfigurasi AWS CLI

1. Di mesin lokal Anda, unduh dan instal file AWS CLI. Ini akan memungkinkan Anda untuk berinteraksi dengan CodePipeline dari baris perintah. Untuk informasi selengkapnya, lihat Menyiapkan dengan Antarmuka Baris AWS Perintah.

   > **Note**
   >
   > CodePipeline hanya berfungsi dengan AWS CLI versi 1.7.38 dan yang lebih baru. Untuk menentukan versi mana AWS CLI yang mungkin telah Anda instal, jalankan perintahaws --version. Untuk meng-upgrade versi yang lebih lama AWS CLI ke versi terbaru, ikuti petunjuk di Menghapus Instalasi AWS CLI, dan kemudian ikuti petunjuk di Instalasi AWS Command Line Interface.

2. Konfigurasikan AWS CLI dengan configure perintah, sebagai berikut:

   ```
   aws configure
   ```

   Saat diminta, tentukan kunci AWS akses dan kunci akses AWS rahasia pengguna IAM yang akan Anda gunakan. CodePipeline Ketika diminta untuk nama wilayah default, tentukan wilayah tempat Anda akan membuat alur, seperti `us-east-2`. Saat diminta untuk format output default, tentukan `json`. Contoh:

   ```
   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**
>
> Untuk informasi selengkapnya tentang IAM, kunci akses, dan kunci rahasia, lihat Mengelola Kunci Akses untuk Pengguna IAM dan Bagaimana Cara Mendapatkan Kredensi? .
> Untuk informasi selengkapnya tentang Wilayah dan titik akhir yang tersedia CodePipeline, lihat AWS CodePipeline titik akhir dan kuota.

Langkah 4: Buka konsol untuk CodePipeline

- Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

Langkah selanjutnya

Anda telah menyelesaikan prasyaratnya. Anda dapat mulai menggunakan CodePipeline. Untuk mulai bekerja dengan CodePipeline, lihatCodePipeline tutorial.

Integrasi produk dan layanan dengan CodePipeline

Secara default, AWS CodePipeline terintegrasi dengan sejumlah produk Layanan AWS dan layanan mitra. Gunakan informasi di bagian berikut untuk membantu Anda mengonfigurasi CodePipeline agar terintegrasi dengan produk dan layanan yang Anda gunakan.

Sumber daya terkait berikut dapat membantu Anda ketika bekerja dengan layanan ini.

Topik

- Integrasi dengan tipe CodePipeline tindakan
- Integrasi umum dengan CodePipeline
- Contoh dari komunitas

Integrasi dengan tipe CodePipeline tindakan

Informasi integrasi dalam topik ini diatur berdasarkan jenis CodePipeline tindakan.

Topik

- Integrasi tindakan sumber
- Bangun integrasi tindakan
- Integrasi tindakan uji
- Menyebarkan integrasi tindakan
- Integrasi tindakan persetujuan dengan Amazon Simple Notification Service
- Memanggil integrasi tindakan

Integrasi tindakan sumber

Informasi berikut diatur berdasarkan jenis CodePipeline tindakan dan dapat membantu Anda mengonfigurasi CodePipeline untuk diintegrasikan dengan penyedia tindakan sumber berikut.

Topik

- Tindakan sumber Amazon ECR
- Tindakan sumber Amazon S3

- Koneksi ke Bitbucket Cloud, GitHub (melalui GitHub Aplikasi), Server GitHub Perusahaan, GitLab .com, dan dikelola sendiri GitLab
- CodeCommit tindakan sumber
- GitHub (melalui OAuth aplikasi) tindakan sumber

Tindakan sumber Amazon ECR

Amazon ECR adalah layanan repositori gambar AWS Docker. Anda menggunakan perintah push and pull Docker untuk mengunggah gambar Docker ke repositori Anda. URI dan gambar repositori Amazon ECR digunakan dalam definisi tugas Amazon ECS untuk referensi informasi gambar sumber.

Pelajari lebih lanjut:

- Untuk melihat parameter konfigurasi dan JSON/YAML cuplikan contoh, lihat Referensi tindakan sumber Amazon ECR
- Buat pipeline, tahapan, dan tindakan
- Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR

Tindakan sumber Amazon S3

Amazon S3 adalah penyimpanan untuk internet. Anda dapat menggunakan Amazon S3 untuk menyimpan dan mengambil data sebanyak apa pun kapan pun, dari mana pun di web. Anda dapat mengonfigurasi CodePipeline untuk menggunakan bucket Amazon S3 berversi sebagai tindakan sumber untuk kode Anda.

> **Note**
>
> Amazon S3 juga dapat disertakan dalam pipeline sebagai tindakan penerapan.

Pelajari lebih lanjut:

- Untuk melihat parameter konfigurasi dan JSON/YAML cuplikan contoh, lihat Referensi tindakan sumber Amazon S3
- Langkah 1: Buat bucket sumber S3 untuk aplikasi Anda

- Buat pipeline (CLI)
- CodePipeline menggunakan Amazon EventBridge (sebelumnya Amazon CloudWatch Events) untuk mendeteksi perubahan di bucket sumber Amazon S3 Anda. Lihat Integrasi umum dengan CodePipeline.

Koneksi ke Bitbucket Cloud, GitHub (melalui GitHub Aplikasi), Server GitHub Perusahaan, GitLab .com, dan dikelola sendiri GitLab

Koneksi (`CodeStarSourceConnection`tindakan) digunakan untuk mengakses Bitbucket Cloud pihak ketiga, GitHub Enterprise Server GitHub, GitLab .com, atau repositori yang GitLab dikelola sendiri.

> **Note**
>
> Fitur ini tidak tersedia di Asia Pasifik (Hong Kong), Asia Pasifik (Hyderabad), Asia Pasifik (Jakarta), Asia Pasifik (Melbourne), Asia Pasifik (Osaka), Afrika (Cape Town), Timur Tengah (Bahrain), Timur Tengah (UEA), Eropa (Spanyol), Eropa (Zurich), Israel (Tel Aviv), atau (AS-Barat) Wilayah. AWS GovCloud Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline. Untuk pertimbangan dengan tindakan ini di Wilayah Eropa (Milan), lihat catatan diCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Awan Bitbucket	Anda dapat mengonfigurasi CodePipeline untuk menggunakan repositor i Bitbucket Cloud sebagai sumber kode Anda. Anda sebelumnya harus membuat akun Bitbucket dan setidaknya satu repositori Bitbucket Cloud. Anda dapat menambahkan tindakan sumber untuk repositori Bitbucket Cloud Anda dengan membuat pipeline atau mengedit yang sudah ada. **Note** Anda dapat membuat koneksi ke repositori Bitbucket Cloud. Jenis penyedia Bitbucket yang diinstal, seperti Bitbucket Server, tidak didukung.

	Anda dapat mengatur sumber daya yang disebut koneksi untuk memungkin kan saluran pipa Anda mengakses repositori kode pihak ketiga. Ketika Anda membuat koneksi, Anda menginstal aplikasi konektor dengan repositori kode pihak ketiga Anda, dan kemudian mengaitkannya dengan koneksi Anda. Untuk Bitbucket Cloud, gunakan opsi Bitbucket di konsol atau `CodestarS ourceConnection` tindakan di CLI. Lihat Koneksi Bitbucket Cloud. Anda dapat menggunakan opsi klon Penuh untuk tindakan ini untuk mereferen sikan metadata Git repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir. Pelajari lebih lanjut: • Untuk melihat parameter konfigurasi dan JSON/YAML cuplikan contoh, lihat. CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri • Untuk melihat tutorial Memulai yang membuat pipeline dengan sumber Bitbucket Cloud, lihat Memulai koneksi.
GitHub atau GitHub Enterpris e Cloud	Anda dapat mengonfigurasi CodePipeline untuk menggunakan GitHub repositori sebagai sumber kode Anda. Anda sebelumnya harus membuat GitHub akun dan setidaknya satu GitHub repositori. Anda dapat menambahk an tindakan sumber untuk GitHub repositori Anda dengan membuat pipeline atau mengedit yang sudah ada. Anda dapat mengatur sumber daya yang disebut koneksi untuk memungkin kan saluran pipa Anda mengakses repositori kode pihak ketiga. Ketika Anda membuat koneksi, Anda menginstal aplikasi konektor dengan repositori kode pihak ketiga Anda, dan kemudian mengaitkannya dengan koneksi Anda. Gunakan opsi penyedia GitHub (melalui GitHub Aplikasi) di konsol atau `CodestarSourceConnection` tindakan di CLI. Lihat GitHub koneksi. Anda dapat menggunakan opsi klon Penuh untuk tindakan ini untuk mereferen sikan metadata Git repositori sehingga tindakan hilir dapat melakukan perintah

	Git secara langsung. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir. Pelajari lebih lanjut: • Untuk melihat parameter konfigurasi dan JSON/YAML cuplikan contoh, lihat CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri • Untuk tutorial yang menunjukkan cara menghubungkan ke GitHub repositori dan menggunakan opsi klon Penuh, lihat. Tutorial: Gunakan klon lengkap dengan sumber GitHub pipa • Tindakan saat ini GitHub (melalui GitHub Aplikasi) adalah tindakan sumber versi 2 untuk GitHub. Tindakan GitHub (melalui OAuth aplikasi) adalah tindakan versi 1 GitHub yang dikelola dengan otentikasi OAuth token. Meskipun kami tidak merekomendasikan penggunaan tindakan GitHub (melalui OAuth aplikasi), saluran pipa yang ada dengan tindakan GitHub (melalui OAuth aplikasi) akan terus berfungsi tanpa dampak apa pun. Sekarang Anda dapat menggunakan tindakan CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri sumber di pipeline yang mengelola aksi GitHub sumber Anda dengan GitHub aplikasi. Jika Anda memiliki pipeline yang menggunakan tindakan GitHub (melalui OAuth aplikasi), lihat langkah-langkah untuk memperbaruinya untuk menggunakan tindakan GitHub (melalui GitHub Aplikasi) diPerbarui tindakan sumber GitHub (melalui OAuth aplikasi) ke tindakan sumber GitHub (melalui GitHub Aplikasi).
GitHub Server Perusahaan	Anda dapat mengonfigurasi CodePipeline untuk menggunakan repositori GitHub Enterprise Server sebagai sumber kode Anda. Anda sebelumnya harus membuat GitHub akun dan setidaknya satu GitHub repositori. Anda dapat menambahkan tindakan sumber untuk repositori GitHub Enterprise Server Anda dengan membuat pipeline atau mengedit yang sudah ada. Anda dapat mengatur sumber daya yang disebut koneksi untuk memungkinkan saluran pipa Anda mengakses repositori kode pihak ketiga. Ketika Anda

	membuat koneksi, Anda menginstal aplikasi konektor dengan repositori kode pihak ketiga Anda, dan kemudian mengaitkannya dengan koneksi Anda. Gunakan opsi penyedia GitHub Enterprise Server di konsol atau `CodestarSourceConnection` tindakan di CLI. Lihat GitHub Koneksi Enterprise Server. Anda dapat menggunakan opsi klon Penuh untuk tindakan ini untuk mereferen sikan metadata Git repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir. Pelajari lebih lanjut: • Untuk melihat parameter konfigurasi dan JSON/YAML cuplikan contoh, lihat CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri • Untuk tutorial yang menunjukkan cara menghubungkan ke GitHub repositor i dan menggunakan opsi klon Penuh, lihat. Tutorial: Gunakan klon lengkap dengan sumber GitHub pipa
GitLab.com	Anda dapat mengonfigurasi CodePipeline untuk menggunakan GitLab repositori.com sebagai sumber kode Anda. Anda harus sebelumnya telah membuat akun GitLab .com dan setidaknya satu GitLab repositori.com. Anda dapat menambahkan tindakan sumber untuk GitLab repositori.com Anda dengan membuat pipeline atau mengedit yang sudah ada. Gunakan opsi GitLabpenyedia di konsol atau `CodestarSourceConnection` tindakan dengan `GitLab` penyedia di CLI. Lihat GitLabkon eksi .com. Pelajari lebih lanjut: • Untuk melihat parameter konfigurasi dan JSON/YAML cuplikan contoh, lihat CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri

GitLab dikelola sendiri	Anda dapat mengonfigurasi CodePipeline untuk menggunakan instalasi yang GitLab dikelola sendiri sebagai sumber kode Anda. Anda harus sebelumnya membuat GitLab akun dan memiliki langganan untuk dikelola sendiri GitLab (Edisi Perusahaan atau Edisi Komunitas). Anda dapat menambahkan tindakan sumber untuk repositori yang GitLab dikelola sendiri dengan membuat pipeline atau mengedit yang sudah ada. Anda dapat mengatur sumber daya yang disebut koneksi untuk memungkinkan saluran pipa Anda mengakses repositori kode pihak ketiga. Ketika Anda membuat koneksi, Anda menginstal aplikasi konektor dengan repositori kode pihak ketiga Anda, dan kemudian mengaitkannya dengan koneksi Anda. Gunakan opsi penyedia yang GitLab dikelola sendiri di konsol atau `CodestarSourceConnection` tindakan di CLI. Lihat Koneksi untuk dikelola GitLab sendiri. Anda dapat menggunakan opsi klon Penuh untuk tindakan ini untuk mereferensikan metadata Git repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir. Pelajari lebih lanjut: • Untuk melihat parameter konfigurasi dan JSON/YAML cuplikan contoh, lihat CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri • Untuk langkah-langkah membuat koneksi dengan jenis penyedia ini, lihatKoneksi untuk dikelola GitLab sendiri.

CodeCommit tindakan sumber

CodeCommitadalah layanan kontrol versi yang dapat Anda gunakan untuk menyimpan dan mengelola aset secara pribadi (seperti dokumen, kode sumber, dan file biner) di cloud. Anda dapat mengonfigurasi CodePipeline untuk menggunakan cabang di CodeCommit repositori sebagai sumber kode Anda. Buat repositori dan kaitkan dengan direktori kerja di mesin lokal Anda. Kemudian Anda dapat membuat pipeline yang menggunakan cabang sebagai bagian dari aksi sumber dalam satu

tahap. Anda dapat terhubung ke CodeCommit repositori dengan membuat pipeline atau mengedit yang sudah ada.

Anda dapat menggunakan opsi klon Penuh untuk tindakan ini untuk mereferensikan metadata Git repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.

Pelajari lebih lanjut:

- Untuk melihat parameter konfigurasi dan JSON/YAML cuplikan contoh, lihat. CodeCommit referensi tindakan sumber
- Tutorial: Buat pipeline sederhana (CodeCommit repositori)
- CodePipeline menggunakan Amazon CloudWatch Events untuk mendeteksi perubahan dalam CodeCommit repositori yang digunakan sebagai sumber untuk pipeline. Setiap tindakan sumber memiliki aturan acara yang sesuai. Aturan acara ini memulai pipeline Anda saat terjadi perubahan di repositori. Lihat Integrasi umum dengan CodePipeline.

GitHub (melalui OAuth aplikasi) tindakan sumber

Tindakan GitHub (melalui OAuth aplikasi) adalah tindakan versi 1 GitHub yang dikelola dengan OAuth Aplikasi. Di Wilayah yang tersedia, Anda juga dapat menggunakan tindakan CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri sumber di pipeline yang mengelola tindakan GitHub sumber Anda dengan GitHub Aplikasi. Jika Anda memiliki pipeline yang menggunakan tindakan GitHub (melalui OAuth aplikasi), lihat langkah-langkah untuk memperbaruinya untuk menggunakan tindakan GitHub (melalui GitHub aplikasi) diPerbarui tindakan sumber GitHub (melalui OAuth aplikasi) ke tindakan sumber GitHub (melalui GitHub Aplikasi).

> **Note**
>
> Meskipun kami tidak merekomendasikan penggunaan tindakan GitHub (melalui OAuth aplikasi), saluran pipa yang ada dengan tindakan GitHub (melalui OAuth aplikasi) akan terus berfungsi tanpa dampak apa pun.

Pelajari lebih lanjut:

- Untuk informasi selengkapnya tentang akses OAuth berbasis GitHub (melalui OAuth aplikasi) yang berbeda dengan GitHub akses berbasis aplikasi, lihat. https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps
- Untuk melihat lampiran yang berisi detail tindakan GitHub (melalui OAuth aplikasi), lihat. Lampiran A: GitHub (melalui OAuth aplikasi) tindakan sumber

Bangun integrasi tindakan

Informasi berikut diatur berdasarkan jenis CodePipeline tindakan dan dapat membantu Anda mengonfigurasi CodePipeline untuk diintegrasikan dengan penyedia tindakan build berikut.

Topik

- CodeBuild membangun tindakan
- CloudBees membangun tindakan
- Amazon ECR membangun dan mempublikasikan tindakan
- Jenkins membangun tindakan
- TeamCity membangun tindakan

CodeBuild membangun tindakan

CodeBuild adalah layanan build yang terkelola penuh yang mengompilasi kode sumber Anda, menjalankan tes unit, dan menghasilkan artefak yang siap dideploy.

Anda dapat menambahkan CodeBuild sebagai tindakan build ke tahap pembuatan pipeline. Untuk informasi selengkapnya, lihat Referensi Konfigurasi CodePipeline Tindakan untukAWS CodeBuild membangun dan menguji referensi tindakan.

> **Note**
>
> CodeBuild juga dapat dimasukkan dalam pipeline sebagai tindakan pengujian, dengan atau tanpa keluaran build.

Pelajari lebih lanjut:

- Untuk melihat parameter konfigurasi dan JSON/YAML cuplikan contoh, lihat. AWS CodeBuild membangun dan menguji referensi tindakan

- Apa itu CodeBuild?
- CodeBuild— Layanan Build yang Dikelola Sepenuhnya

CloudBees membangun tindakan

Anda dapat mengonfigurasi CodePipeline CloudBeesuntuk digunakan untuk membangun atau menguji kode Anda dalam satu atau beberapa tindakan dalam pipeline.

Pelajari lebih lanjut:

- Re: invent 2017: Cloud Pertama dengan AWS

Amazon ECR membangun dan mempublikasikan tindakan

Amazon ECR adalah layanan repositori gambar AWS Docker. Anda menggunakan perintah push and pull Docker untuk mengunggah gambar Docker ke repositori Anda.

Anda dapat menambahkan `ECRBuildAndPublish` tindakan ke pipeline Anda untuk mengotomatiskan pembuatan dan mendorong gambar. Untuk informasi selengkapnya, lihat referensi konfigurasi CodePipeline Tindakan untuk`ECRBuildAndPublish`membangun referensi tindakan.

Jenkins membangun tindakan

Anda dapat mengonfigurasi CodePipeline untuk menggunakan Jenkins CI untuk membangun atau menguji kode Anda dalam satu atau beberapa tindakan dalam pipeline. Anda harus sebelumnya membuat proyek Jenkins dan menginstal dan mengkonfigurasi CodePipeline Plugin untuk Jenkins untuk proyek itu. Anda dapat terhubung ke proyek Jenkins dengan membuat pipeline baru atau mengedit yang sudah ada.

Akses untuk Jenkins dikonfigurasi berdasarkan per proyek. Anda harus menginstal CodePipeline Plugin untuk Jenkins pada setiap instance Jenkins yang ingin Anda gunakan. CodePipeline Anda juga harus mengonfigurasi CodePipeline akses ke proyek Jenkins. Amankan proyek Jenkins Anda dengan mengonfigurasinya untuk menerima koneksi HTTPS/SSL saja. Jika project Jenkins Anda diinstal pada instans Amazon EC2, pertimbangkan untuk memberikan AWS kredensialnya dengan menginstal AWS CLI pada setiap instans. Kemudian konfigurasikan AWS profil pada instance tersebut dengan kredensyal yang ingin Anda gunakan untuk koneksi. Ini adalah alternatif untuk menambahkan dan menyimpannya melalui antarmuka web Jenkins.

Pelajari lebih lanjut:

- Mengakses Jenkins
- Tutorial: Buat pipeline empat tahap

TeamCity membangun tindakan

Anda dapat mengonfigurasi CodePipeline TeamCityuntuk digunakan untuk membangun dan menguji kode Anda dalam satu atau beberapa tindakan dalam pipeline.

Pelajari lebih lanjut:

- TeamCity Plugin untuk CodePipeline

Integrasi tindakan uji

Informasi berikut diatur berdasarkan jenis CodePipeline tindakan dan dapat membantu Anda mengonfigurasi CodePipeline untuk diintegrasikan dengan penyedia tindakan pengujian berikut.

Topik

- CodeBuild tindakan uji
- AWS Device Farm tindakan uji
- Tindakan uji Ghost Inspector
- OpenText LoadRunner Tindakan uji cloud
- Mencerminkan otomatisasi uji

CodeBuild tindakan uji

CodeBuildadalah layanan build yang dikelola sepenuhnya di cloud. CodeBuild mengkompilasi kode sumber Anda, menjalankan pengujian unit, dan menghasilkan artefak yang siap digunakan.

Anda dapat menambahkan CodeBuild ke pipeline sebagai tindakan pengujian. Untuk informasi selengkapnya, lihat Referensi Konfigurasi CodePipeline Tindakan untukAWS CodeBuild membangun dan menguji referensi tindakan.

> **Note**
>
> CodeBuild juga dapat dimasukkan dalam pipeline sebagai aksi build, dengan artefak keluaran build wajib.

Pelajari lebih lanjut:

- Untuk melihat parameter konfigurasi dan JSON/YAML cuplikan contoh, lihat. AWS CodeBuild membangun dan menguji referensi tindakan
- Apa itu CodeBuild?

AWS Device Farm tindakan uji

AWS Device Farmadalah layanan pengujian aplikasi yang dapat Anda gunakan untuk menguji dan berinteraksi dengan aplikasi Android, iOS, dan web Anda di ponsel dan tablet fisik nyata. Anda dapat mengonfigurasi CodePipeline AWS Device Farm untuk digunakan untuk menguji kode Anda dalam satu atau beberapa tindakan dalam pipeline. AWS Device Farm memungkinkan Anda untuk mengunggah tes Anda sendiri atau menggunakan uji kompatibilitas bawaan dan bebas skrip. Karena pengujian dilakukan secara paralel, pengujian pada beberapa perangkat dimulai dalam hitungan menit. Laporan pengujian yang berisi hasil tingkat tinggi, log tingkat rendah, pixel-to-pixel tangkapan layar, dan data kinerja diperbarui saat pengujian selesai. AWS Device Farm mendukung pengujian aplikasi Android, iOS, dan Fire OS asli dan hibrida, termasuk yang dibuat dengan, Titanium PhoneGap, Xamarin, Unity, dan kerangka kerja lainnya. Ini mendukung akses jarak jauh aplikasi Android, yang memungkinkan Anda berinteraksi langsung dengan perangkat uji.

Pelajari lebih lanjut:

- Untuk melihat parameter konfigurasi dan JSON/YAML cuplikan contoh, lihat. AWS Device Farm referensi tindakan uji
- Apa itu AWS Device Farm?
- Menggunakan AWS Device Farm dalam Tahap CodePipeline Uji

Tindakan uji Ghost Inspector

Anda dapat mengonfigurasi CodePipeline untuk menggunakan Ghost Inspector untuk menguji kode Anda dalam satu atau beberapa tindakan dalam pipeline.

Pelajari lebih lanjut:

- Dokumentasi Ghost Inspector untuk integrasi layanan dengan CodePipeline

OpenText LoadRunner Tindakan uji cloud

Anda dapat mengonfigurasi CodePipeline untuk menggunakan OpenText LoadRunner Cloud dalam satu atau beberapa tindakan dalam pipeline.

Pelajari lebih lanjut:

- LoadRunner Dokumentasi cloud untuk diintegrasikan dengan CodePipeline

Mencerminkan otomatisasi uji

Reflect adalah solusi otomatisasi pengujian bertenaga AI yang memberdayakan Anda untuk menyederhanakan pengujian dan mengatasi tantangan proses manual. Dengan otomatisasi pengujian tanpa kode, Reflect menyederhanakan pembuatan, eksekusi, dan pemeliharaan pengujian, memungkinkan Anda membuat pengujian yang kuat dan berulang tanpa memerlukan pengetahuan teknis apa pun. Dengan menghilangkan kompleksitas dan memastikan gangguan minimal pada alur kerja Anda, ini membantu Anda mempercepat pengujian dan dengan percaya diri memberikan aplikasi berkualitas tinggi setiap saat.

Pelajari lebih lanjut:

- AWS CodePipeline uji integrasi dengan Reflect

Menyebarkan integrasi tindakan

Informasi berikut diatur berdasarkan jenis CodePipeline tindakan dan dapat membantu Anda mengonfigurasi CodePipeline untuk diintegrasikan dengan penyedia tindakan penerapan berikut.

Topik

- Tindakan penerapan Amazon EC2
- Tindakan penerapan Amazon Elastic Kubernetes Service EKS
- Tindakan penerapan Amazon S3
- AWS AppConfig menyebarkan tindakan

- CloudFormation menyebarkan tindakan
- CloudFormation StackSets menyebarkan tindakan
- Tindakan penerapan Amazon ECS
- Tindakan penyebaran Elastic Beanstalk
- OpsWorks menyebarkan tindakan
- Tindakan penyebaran Service Catalog
- Amazon Alexa menerapkan tindakan
- CodeDeploy menyebarkan tindakan
- XebiaLabs menyebarkan tindakan

Tindakan penerapan Amazon EC2

Amazon EC2 memungkinkan Anda membuat dan mengelola komputasi di cloud. Anda dapat menambahkan tindakan ke pipeline yang menggunakan Amazon EC2 sebagai penyedia penerapan yang menerapkan aplikasi ke instans Anda.

Pelajari lebih lanjut:

- Lihat halaman referensi tindakan diReferensi tindakan Amazon EC2.
- Untuk tutorial, lihat Tutorial: Terapkan ke instans Amazon EC2 dengan CodePipeline.

Tindakan penerapan Amazon Elastic Kubernetes Service **EKS**

Amazon EKS memungkinkan Anda membuat dan mengelola cluster kubernetes. Anda dapat menambahkan tindakan ke pipeline yang menggunakan Amazon EKS sebagai penyedia penerapan yang menyebarkan gambar ke klaster. Anda dapat menggunakan template helm atau file manifes kubernetes.

Pelajari lebih lanjut:

- Lihat halaman referensi tindakan diReferensi tindakan penerapan Amazon Elastic Kubernetes Service EKS.
- Untuk tutorial, lihat Tutorial: Terapkan ke Amazon EKS dengan CodePipeline.

Tindakan penerapan Amazon S3

Amazon S3 adalah penyimpanan untuk internet. Anda dapat menggunakan Amazon S3 untuk menyimpan dan mengambil data sebanyak apa pun kapan pun, dari mana pun di web. Anda dapat menambahkan tindakan ke pipeline yang menggunakan Amazon S3 sebagai penyedia penerapan.

> **Note**
>
> Amazon S3 juga dapat dimasukkan dalam pipeline sebagai aksi sumber.

Pelajari lebih lanjut:

- Buat pipeline, tahapan, dan tindakan
- Tutorial: Membuat pipeline yang menggunakan Amazon S3 sebagai penyedia penerapan

AWS AppConfig menyebarkan tindakan

AWS AppConfig adalah kemampuan AWS Systems Manager untuk membuat, mengelola, dan dengan cepat menyebarkan konfigurasi aplikasi. Anda dapat menggunakan AppConfig dengan aplikasi yang dihosting di instans EC2, wadah AWS Lambda, aplikasi seluler, atau perangkat IoT.

Pelajari lebih lanjut:

- CodePipeline Referensi Konfigurasi Tindakan untuk AWS AppConfig menyebarkan referensi tindakan
- Tutorial: Buat pipeline yang digunakan AWS AppConfig sebagai penyedia penyebaran

CloudFormation menyebarkan tindakan

CloudFormationmemberi pengembang dan administrator sistem cara mudah untuk membuat dan mengelola kumpulan AWS sumber daya terkait, menggunakan templat untuk menyediakan dan memperbarui sumber daya tersebut. Anda dapat menggunakan templat sampel layanan atau membuatnya sendiri. Template menjelaskan AWS sumber daya dan dependensi atau parameter runtime apa pun yang diperlukan untuk menjalankan aplikasi Anda.

Model Aplikasi AWS Tanpa Server (AWS SAM) diperluas CloudFormation untuk menyediakan cara yang disederhanakan untuk mendefinisikan dan menyebarkan aplikasi tanpa server. AWS SAM

mendukung Amazon API Gateway APIs, fungsi AWS Lambda, dan tabel Amazon DynamoDB. Anda dapat menggunakan CodePipeline dengan CloudFormation dan AWS SAM untuk terus mengirimkan aplikasi tanpa server Anda.

Anda dapat menambahkan tindakan ke pipeline yang digunakan AWS CloudFormation sebagai penyedia penerapan. Saat digunakan AWS CloudFormation sebagai penyedia penerapan, Anda dapat mengambil tindakan pada CloudFormation tumpukan dan mengubah set sebagai bagian dari eksekusi pipeline. CloudFormation dapat membuat, memperbarui, mengganti, dan menghapus tumpukan dan mengubah set saat pipeline berjalan. Akibatnya, AWS dan sumber daya kustom dapat dibuat, disediakan, diperbarui, atau dihentikan selama eksekusi pipeline sesuai dengan spesifikasi yang Anda berikan dalam CloudFormation template dan definisi parameter.

Pelajari lebih lanjut:

- CodePipeline Referensi Konfigurasi Tindakan untuk CloudFormation menyebarkan referensi tindakan
- Pengiriman Berkelanjutan dengan CodePipeline - Pelajari cara menggunakan CodePipeline untuk membangun alur kerja pengiriman berkelanjutan untuk CloudFormation.
- Mengotomatisasi Penerapan Aplikasi Berbasis Lambda — Pelajari cara menggunakan Model Aplikasi AWS Tanpa Server dan untuk membangun alur kerja pengiriman berkelanjutan CloudFormation untuk aplikasi berbasis Lambda Anda.

CloudFormation StackSets menyebarkan tindakan

CloudFormationmemberi Anda cara untuk menyebarkan sumber daya di beberapa akun dan AWS Wilayah.

Anda dapat menggunakannya CodePipeline CloudFormation untuk memperbarui definisi kumpulan tumpukan dan menerapkan pembaruan ke instance Anda.

Anda dapat menambahkan tindakan berikut ke pipeline untuk digunakan AWS CloudFormation StackSets sebagai penyedia penerapan.

- CloudFormationStackSet
- CloudFormationStackInstances

Pelajari lebih lanjut:

- CodePipeline Referensi Konfigurasi Tindakan untuk CloudFormation StackSets menyebarkan referensi tindakan
- Tutorial: Buat pipeline dengan AWS CloudFormation StackSets tindakan penerapan

Tindakan penerapan Amazon ECS

Amazon ECS adalah layanan manajemen kontainer yang sangat skalabel dan berkinerja tinggi yang memungkinkan Anda menjalankan aplikasi berbasis kontainer di. AWS Cloud Saat membuat pipeline, Anda dapat memilih Amazon ECS sebagai penyedia penerapan. Perubahan kode di repositori kontrol sumber memicu pipeline Anda untuk membuat image Docker baru, mendorongnya ke registri penampung, lalu menerapkan gambar yang diperbarui ke Amazon ECS. Anda juga dapat menggunakan tindakan penyedia ECS (Biru/Hijau) CodePipeline untuk merutekan dan menyebarkan lalu lintas ke Amazon ECS. CodeDeploy

Pelajari lebih lanjut:

- Apa itu Amazon ECS?
- Tutorial: Penerapan Berkelanjutan dengan CodePipeline
- Buat pipeline, tahapan, dan tindakan
- Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR

Tindakan penyebaran Elastic Beanstalk

Elastic Beanstalk adalah layanan untuk menyebarkan dan menskalakan aplikasi dan layanan web yang dikembangkan dengan Java, .NET, PHP, Node.js, Python, Ruby, Go, dan Docker pada server yang sudah dikenal seperti Apache, Nginx, Passenger, dan IIS. Anda dapat mengonfigurasi CodePipeline untuk menggunakan Elastic Beanstalk untuk menyebarkan kode Anda. Anda dapat membuat aplikasi dan lingkungan Elastic Beanstalk untuk digunakan dalam tindakan penerapan dalam tahap baik sebelum membuat pipeline atau saat Anda menggunakan wizard Create Pipeline.

> **Note**
>
> Fitur ini tidak tersedia di Asia Pasifik (Hyderabad), Asia Pasifik (Melbourne), Timur Tengah (UEA), Eropa (Spanyol), atau Eropa (Zurich). Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline.

Pelajari lebih lanjut:

- Memulai menggunakan Elastic Beanstalk
- Buat pipeline, tahapan, dan tindakan

OpsWorks menyebarkan tindakan

OpsWorks adalah layanan manajemen konfigurasi yang membantu Anda mengonfigurasi dan mengoperasikan aplikasi dari segala bentuk dan ukuran menggunakan Chef. Dengan menggunakan AWS OpsWorks Stacks, Anda dapat menentukan arsitektur aplikasi dan spesifikasi setiap komponen termasuk instalasi paket, konfigurasi perangkat lunak dan sumber daya seperti penyimpanan. Anda dapat mengonfigurasi CodePipeline untuk digunakan AWS OpsWorks Stacks untuk menyebarkan kode Anda bersama dengan buku masak dan aplikasi Chef kustom di. OpsWorks

- Custom Chef Cookbooks — OpsWorks menggunakan Chef Cookbooks untuk menangani tugas-tugas seperti menginstal dan mengkonfigurasi paket dan menyebarkan aplikasi.
- Aplikasi — OpsWorks Aplikasi terdiri dari kode yang ingin Anda jalankan di server aplikasi. Kode aplikasi disimpan dalam repositori, seperti bucket Amazon S3.

Sebelum Anda membuat pipeline, Anda membuat OpsWorks stack dan layer. Anda dapat membuat OpsWorks aplikasi untuk digunakan dalam tindakan penerapan dalam tahap baik sebelum Anda membuat pipeline atau ketika Anda menggunakan wizard Create Pipeline.

CodePipeline dukungan untuk saat OpsWorks ini tersedia di Wilayah AS Timur (Virginia N.) (us-east-1) saja.

Pelajari lebih lanjut:

- Menggunakan CodePipeline dengan AWS OpsWorks Stacks
- Buku Masak dan Resep
- AWS OpsWorks Aplikasi

Tindakan penyebaran Service Catalog

Service Catalog memungkinkan organisasi untuk membuat dan mengelola katalog produk yang disetujui untuk digunakan. AWS

Anda dapat mengonfigurasi CodePipeline untuk menerapkan pembaruan dan versi templat produk Anda ke Service Catalog. Anda dapat membuat produk Service Catalog untuk digunakan dalam tindakan penerapan dan kemudian menggunakan wizard Create Pipeline untuk membuat pipeline.

Pelajari lebih lanjut:

- Tutorial: Buat pipeline yang di-deploy ke Service Catalog
- Buat pipeline, tahapan, dan tindakan

Amazon Alexa menerapkan tindakan

Amazon Alexa Skills Kit memungkinkan Anda membangun dan mendistribusikan keterampilan berbasis cloud kepada pengguna perangkat berkemampuan Alexa.

> **Note**
>
> Fitur ini tidak tersedia di Wilayah Asia Pasifik (Hong Kong) atau Eropa (Milan). Untuk menggunakan tindakan penerapan lain yang tersedia di Wilayah tersebut, lihatMenyebarkan integrasi tindakan.

Anda dapat menambahkan tindakan ke pipeline yang menggunakan Alexa Skills Kit sebagai penyedia penerapan. Perubahan sumber terdeteksi oleh pipeline Anda, dan kemudian pipeline Anda menyebarkan pembaruan ke keterampilan Alexa Anda di layanan Alexa.

Pelajari lebih lanjut:

- Tutorial: Buat pipeline yang menerapkan keterampilan Amazon Alexa

CodeDeploy menyebarkan tindakan

CodeDeploymengoordinasikan penerapan aplikasi ke Amazon EC2/Instans lokal, platform komputasi Amazon Elastic Container Service, dan platform komputasi tanpa server. AWS Lambda Anda dapat mengonfigurasi CodePipeline untuk digunakan CodeDeploy untuk menyebarkan kode Anda. Anda dapat membuat grup CodeDeploy aplikasi, deployment, dan deployment untuk digunakan dalam tindakan deploy dalam satu tahap baik sebelum membuat pipeline atau saat Anda menggunakan wizard Create Pipeline.

Pelajari lebih lanjut:

- Langkah 3: Buat aplikasi di CodeDeploy
- Tutorial: Buat pipeline sederhana (CodeCommit repositori)

XebiaLabs menyebarkan tindakan

Anda dapat mengonfigurasi CodePipeline XebiaLabsuntuk digunakan untuk menyebarkan kode Anda dalam satu atau beberapa tindakan dalam pipeline.

Pelajari lebih lanjut:

- Menggunakan XL Deploy dengan CodePipeline

Integrasi tindakan persetujuan dengan Amazon Simple Notification Service

Amazon SNS adalah layanan pemberitahuan push yang cepat, fleksibel, dan dikelola sepenuhnya yang memungkinkan Anda mengirim pesan individual atau menyebarkan pesan ke sejumlah besar penerima. Amazon SNS membuatnya sederhana dan hemat biaya untuk mengirim pemberitahuan push ke pengguna perangkat seluler, penerima email, atau bahkan mengirim pesan ke layanan terdistribusi lainnya.

Saat Anda membuat permintaan persetujuan manual di CodePipeline, Anda dapat secara opsional mempublikasikan ke topik di Amazon SNS sehingga semua pengguna IAM yang berlangganan itu diberi tahu bahwa tindakan persetujuan siap ditinjau.

Pelajari lebih lanjut:

- Apa itu Amazon SNS?
- Berikan izin Amazon SNS ke peran layanan CodePipeline

Memanggil integrasi tindakan

Informasi berikut diatur berdasarkan jenis CodePipeline tindakan dan dapat membantu Anda mengonfigurasi CodePipeline untuk diintegrasikan dengan penyedia tindakan pemanggilan berikut.

Topik

- Amazon Inspector memanggil tindakan
- Lambda memanggil tindakan
- Step Functions memanggil tindakan

Amazon Inspector memanggil tindakan

Amazon Inspector adalah layanan manajemen kerentanan yang secara otomatis menemukan beban kerja dan terus memindai mereka untuk kerentanan perangkat lunak dan paparan jaringan yang tidak diinginkan. Amazon Inspector mendukung beberapa format arsip termasuk tar dan perang, dan Amazon Inspector mendukung binari termasuk binari Rust dan Go.

Anda dapat mengonfigurasi CodePipeline `InspectorScan` tindakan untuk mengotomatiskan pemindaian kode sumber atau repositori gambar Amazon ECR untuk kerentanan.

Pelajari lebih lanjut:

- CodePipeline referensi konfigurasi tindakan untuk Amazon Inspector `InspectorScan` memanggil referensi tindakan

Lambda memanggil tindakan

Lambda – Memungkinkan Anda menjalankan kode tanpa penyediaan atau pengelolaan server. Anda dapat mengonfigurasi CodePipeline untuk menggunakan fungsi Lambda untuk menambahkan fleksibilitas dan fungsionalitas ke saluran pipa Anda. Anda dapat membuat fungsi Lambda untuk ditambahkan sebagai tindakan dalam tahap baik sebelum membuat pipeline atau saat Anda menggunakan wizard Buat Pipeline.

Pelajari lebih lanjut:

- CodePipeline Referensi konfigurasi tindakan untuk AWS Lambda memanggil referensi tindakan
- Memanggil AWS Lambda fungsi dalam pipeline di CodePipeline

Step Functions memanggil tindakan

Step Functions memungkinkan Anda membuat dan mengkonfigurasi mesin status. Anda dapat mengonfigurasi CodePipeline untuk menggunakan tindakan panggilan Step Functions untuk memicu eksekusi mesin status.

Pelajari lebih lanjut:

- CodePipeline Referensi Konfigurasi Tindakan untuk AWS Step Functions memanggil referensi tindakan
- Tutorial: Gunakan AWS Step Functions tindakan pemanggilan dalam pipeline

Integrasi umum dengan CodePipeline

Layanan AWS Integrasi berikut tidak didasarkan pada jenis CodePipeline tindakan.

Amazon CloudWatch	Amazon CloudWatch memantau AWS sumber daya Anda. Pelajari lebih lanjut: • Apa itu Amazon CloudWatch?
Amazon EventBridge	Amazon EventBridge adalah layanan web yang mendeteksi perubahan Layanan AWS berdasarkan aturan yang Anda tetapkan dan memanggil tindakan dalam satu atau lebih yang ditentukan Layanan AWS saat perubahan terjadi. • Mulai eksekusi pipeline secara otomatis ketika ada perubahan — Anda dapat mengonfigurasi CodePipeline sebagai target dalam aturan yang disiapkan di Amazon EventBridge. Ini mengatur saluran pipa untuk memulai secara otomatis ketika layanan lain berubah. Pelajari lebih lanjut: • Apa itu Amazon EventBridge? • Mulai pipa di CodePipeline. • CodeCommit tindakan sumber dan EventBridge • Menerima pemberitahuan saat status pipeline berubah — Anda dapat mengatur EventBridge aturan untuk mendeteksi dan bereaksi terhadap perubahan status eksekusi untuk pipeline, tahapan, atau tindakan. Pelajari lebih lanjut: • Memantau CodePipeline peristiwa

	• Tutorial: Menyiapkan aturan CloudWatch Acara untuk menerima pemberitahuan email untuk perubahan status pipeline
AWS Cloud9	AWS Cloud9 adalah IDE online, yang Anda akses melalui browser web Anda. IDE menawarkan pengalaman pengeditan kode yang kaya dengan dukungan untuk beberapa bahasa pemrograman dan debugger runtime, serta terminal bawaan. Di latar belakang, instans Amazon EC2 menampung lingkungan AWS Cloud9 pengembangan. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS Cloud9. Pelajari lebih lanjut: • Menyiapkan AWS Cloud9
AWS CloudTrail	CloudTrailmenangkap panggilan AWS API dan peristiwa terkait yang dibuat oleh atau atas nama AWS akun dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengonfigurasi CloudTrail untuk menangkap panggilan API dari CodePipeline konsol, CodePipeline perintah dari AWS CLI, dan dari CodePipeline API. Pelajari lebih lanjut: • Pencatatan panggilan CodePipeline API dengan AWS CloudTrail
AWS CodeStar Pemberitahuan	Anda dapat mengatur notifikasi agar pengguna mengetahui perubahan penting, seperti saat pipeline mulai dieksekusi. Untuk informasi selengkapnya, lihat Membuat aturan notifikasi.

AWS Key Management Service	AWS KMS adalah layanan terkelola yang memungkinkan Anda membuat dan mengendalikan kunci enkripsi yang digunakan untuk mengenkripsi data Anda. Secara default, CodePipeline penggunaan AWS KMS untuk mengenkripsi artefak untuk saluran pipa yang disimpan di bucket Amazon S3. Pelajari lebih lanjut: • Untuk membuat pipeline yang menggunakan bucket sumber, bucket artefak, dan peran layanan dari satu AWS akun dan CodeDeploy sumber daya dari AWS akun lain, Anda harus membuat kunci KMS yang dikelola pelanggan, menambahkan kunci ke pipeline, dan menyiapkan kebijakan dan peran akun untuk mengaktifkan akses lintas akun. Untuk informasi selengkapnya, lihat Buat pipeline CodePipeline yang menggunakan sumber daya dari AWS akun lain. • Untuk membuat pipeline dari satu AWS akun yang menyebarkan CloudFormation tumpukan ke AWS akun lain, Anda harus membuat kunci KMS yang dikelola pelanggan, menambahkan kunci ke pipeline, dan menyiapkan kebijakan dan peran akun untuk menyebarkan tumpukan ke akun lain. AWS Untuk informasi selengkapnya, lihat Bagaimana CodePipeline cara menggunakan AWS CloudFormation tumpukan di akun lain? • Untuk mengonfigurasi enkripsi sisi server untuk bucket artefak S3 pipeline, Anda dapat menggunakan kunci KMS AWS terkelola default atau membuat kunci KMS yang dikelola pelanggan dan menyiapkan kebijakan bucket untuk menggunakan kunci enkripsi. Untuk informasi selengkapnya, lihat Konfigurasikan enkripsi sisi server untuk artefak yang disimpan di Amazon S3 CodePipeline. Untuk AWS KMS key, Anda dapat menggunakan ID kunci, kunci ARN, atau alias ARN. **Note** Alias hanya dikenali di akun yang membuat kunci KMS. Untuk tindakan lintas akun, Anda hanya dapat menggunakan ID kunci

atau ARN kunci untuk mengidentifikasi kunci. Tindakan lintas akun melibatkan penggunaan peran dari akun lain (accountB), sehingga menentukan ID kunci akan menggunakan kunci dari akun lain (accounTB).

Contoh dari komunitas

Bagian berikut menyediakan tautan ke unggahan blog, artikel, dan contoh yang disediakan komunitas.

> **Note**
>
> Tautan ini disediakan hanya untuk tujuan informasi, dan tidak boleh dianggap sebagai daftar lengkap atau dukungan atas isi contoh. AWS tidak bertanggung jawab atas konten atau keakuratan konten eksternal.

Topik

- Contoh integrasi: Posting blog

Contoh integrasi: Posting blog

- Melacak status AWS CodePipeline build dari repositori Git pihak ketiga

  Pelajari cara menyiapkan sumber daya yang akan menampilkan pipeline Anda dan membangun status tindakan di repositori pihak ketiga Anda, sehingga memudahkan pengembang untuk melacak status tanpa beralih konteks.

  Dipublikasikan Maret 2021

- Lengkap CI/CD dengan AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, dan AWS CodePipeline

  Pelajari cara menyiapkan pipeline yang menggunakan CodeCommit,, dan CodeDeploy layanan untuk mengompilasi CodePipeline CodeBuild, membangun, dan menginstal aplikasi Java yang dikendalikan versi ke satu set instans Amazon EC2 Linux.

Telah Terbit September 2020

- Menguji dan membuat CI/CD pipeline untuk AWS Step Functions

  Pelajari cara menyiapkan sumber daya yang akan mengoordinasikan mesin status Step Functions dan pipeline Anda.

Dipublikasikan Maret 2020

- Menerapkan DevSecOps Menggunakan CodePipeline

  Pelajari cara menggunakan CI/CD pipeline CodePipeline untuk mengotomatiskan kontrol keamanan preventif dan detektif. Posting ini mencakup cara menggunakan pipeline untuk membuat grup keamanan sederhana dan melakukan pemeriksaan keamanan selama tahap sumber, pengujian, dan produksi untuk meningkatkan postur keamanan AWS akun Anda.

Dipublikasikan Maret 2017

- Penerapan Berkelanjutan ke Amazon ECS Menggunakan CodePipeline,, CodeBuild Amazon ECR, dan CloudFormation

  Pelajari cara membuat pipeline penerapan berkelanjutan ke Amazon Elastic Container Service (Amazon ECS). Aplikasi dikirimkan sebagai wadah Docker menggunakan CodePipeline, CodeBuild, Amazon ECR, dan. CloudFormation

  - Unduh contoh CloudFormation template dan instruksi untuk menggunakannya untuk membuat pipeline penerapan berkelanjutan Anda sendiri dari ECS Reference Architecture: Continuous Deployment repo on. GitHub

Telah Terbit Januari 2017

- Penerapan Berkelanjutan untuk Aplikasi Tanpa Server

  Pelajari cara menggunakan koleksi Layanan AWS untuk membuat pipeline penerapan berkelanjutan untuk aplikasi tanpa server Anda. Anda akan menggunakan Model Aplikasi Tanpa Server (SAM) untuk menentukan aplikasi dan sumber dayanya dan CodePipeline untuk mengatur penerapan aplikasi Anda.

  - Lihat contoh aplikasi yang ditulis dalam Go dengan framework Gin dan shim proxy API Gateway.

Telah Terbit Desember 2016

- Menskalakan DevOps Deployment dengan dan Dynatrace CodePipeline

Pelajari cara menggunakan solusi pemantauan Dynatrace untuk menskalakan saluran pipa CodePipeline, menganalisis eksekusi pengujian secara otomatis sebelum kode dilakukan, dan pertahankan waktu tunggu yang optimal.

Telah Terbit November 2016

- Buat Pipeline untuk AWS Elastic Beanstalk dalam CodePipeline Menggunakan AWS CloudFormation dan CodeCommit

  Pelajari cara menerapkan pengiriman berkelanjutan dalam CodePipeline pipeline untuk aplikasi di AWS Elastic Beanstalk. Semua AWS sumber daya disediakan secara otomatis melalui penggunaan template. CloudFormation Panduan ini juga menggabungkan CodeCommit dan AWS Identity and Access Management (IAM).

  Diterbitkan Mei 2016

- Otomatiskan CodeCommit dan masuk CodePipeline CloudFormation

  Gunakan CloudFormation untuk mengotomatiskan penyediaan AWS sumber daya untuk pipeline pengiriman berkelanjutan yang menggunakan CodeCommit,, CodePipeline, CodeDeploy dan. AWS Identity and Access Management

  Telah Terbit April 2016

- Buat Pipeline Lintas Akun di AWS CodePipeline

  Pelajari cara mengotomatiskan penyediaan akses lintas akun ke pipeline dengan menggunakan. AWS CodePipeline AWS Identity and Access Management Termasuk contoh dalam CloudFormation template.

  Dipublikasikan Maret 2016

- Menjelajahi ASP.NET Core Bagian 2: Pengiriman Berkelanjutan

  Pelajari cara membuat sistem pengiriman berkelanjutan penuh untuk aplikasi ASP.NET Core menggunakan CodeDeploy dan AWS CodePipeline.

  Dipublikasikan Maret 2016

- Membuat Pipeline Menggunakan AWS CodePipeline Konsol

  Pelajari cara menggunakan AWS CodePipeline konsol untuk membuat pipeline dua tahap dalam panduan berdasarkan. AWS CodePipeline Tutorial: Buat pipeline empat tahap

Dipublikasikan Maret 2016

- Mengejek AWS CodePipeline Pipa dengan AWS Lambda

  Pelajari cara menjalankan fungsi Lambda yang memungkinkan Anda memvisualisasikan tindakan dan tahapan dalam CodePipeline proses pengiriman perangkat lunak saat Anda mendesainnya, sebelum pipeline beroperasi. Saat Anda mendesain struktur pipa, Anda dapat menggunakan fungsi Lambda untuk menguji apakah pipeline Anda akan berhasil diselesaikan.

  Telah Terbit Februari 2016

- Menjalankan AWS Lambda Fungsi dalam CodePipeline Menggunakan AWS CloudFormation

  Pelajari cara membuat AWS CloudFormation tumpukan yang menyediakan semua AWS sumber daya yang digunakan dalam tugas panduan penggunaMemanggil AWS Lambda fungsi dalam pipeline di CodePipeline.

  Telah Terbit Februari 2016

- Penyediaan Tindakan Kustom CodePipeline di AWS CloudFormation

  Pelajari cara menggunakan AWS CloudFormation untuk menyediakan tindakan kustom di CodePipeline.

  Telah Terbit Januari 2016

- Penyediaan CodePipeline dengan AWS CloudFormation

  Pelajari cara menyediakan saluran pengiriman kontinu dasar dalam CodePipeline menggunakan AWS CloudFormation.

  Telah Terbit Desember 2015

- Menerapkan dari CodePipeline ke OpsWorks Menggunakan Tindakan Kustom dan AWS Lambda

  Pelajari cara mengonfigurasi pipeline Anda dan AWS Lambda fungsi yang akan digunakan untuk AWS OpsWorks digunakan CodePipeline.

  Telah Terbit Juli 2015

CodePipeline tutorial

Setelah Anda menyelesaikan langkah-langkahnyaMemulai dengan CodePipeline, Anda dapat mencoba salah satu AWS CodePipeline tutorial di panduan pengguna ini.

Topik

- Tutorial: Terapkan ke instans Amazon EC2 dengan CodePipeline
- Tutorial: Buat dan dorong gambar Docker ke Amazon ECR dengan CodePipeline (tipe V2)
- Tutorial: Terapkan ke Amazon EKS dengan CodePipeline
- Tutorial: Buat pipeline yang menjalankan perintah dengan komputasi (tipe V2)
- Tutorial: Gunakan tag Git untuk memulai pipeline Anda
- Tutorial: Filter nama cabang untuk permintaan tarik untuk memulai pipeline Anda (tipe V2)
- Tutorial: Gunakan variabel tingkat pipa
- Tutorial: Buat pipeline sederhana (ember S3)
- Tutorial: Buat pipeline sederhana (CodeCommit repositori)
- Tutorial: Buat pipeline empat tahap
- Tutorial: Menyiapkan aturan CloudWatch Acara untuk menerima pemberitahuan email untuk perubahan status pipeline
- Tutorial: Membuat pipeline yang membangun dan menguji aplikasi Android Anda AWS Device Farm
- Tutorial: Buat pipeline yang menguji aplikasi iOS Anda AWS Device Farm
- Tutorial: Buat pipeline yang di-deploy ke Service Catalog
- Tutorial: Buat pipeline dengan AWS CloudFormation
- Tutorial: Buat pipeline yang menggunakan variabel dari AWS CloudFormation tindakan penerapan
- Tutorial: Penerapan Standar Amazon ECS dengan CodePipeline
- Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR
- Tutorial: Buat pipeline yang menerapkan keterampilan Amazon Alexa
- Tutorial: Membuat pipeline yang menggunakan Amazon S3 sebagai penyedia penerapan
- Tutorial: Buat pipeline yang menerbitkan aplikasi tanpa server Anda ke AWS Serverless Application Repository

- Tutorial: Penerapan fungsi Lambda dengan CodePipeline
- Tutorial: Menggunakan variabel dengan tindakan panggilan Lambda
- Tutorial: Gunakan AWS Step Functions tindakan pemanggilan dalam pipeline
- Tutorial: Buat pipeline yang digunakan AWS AppConfig sebagai penyedia penyebaran
- Tutorial: Gunakan klon lengkap dengan sumber GitHub pipa
- Tutorial: Gunakan klon lengkap dengan sumber CodeCommit pipa
- Tutorial: Buat pipeline dengan AWS CloudFormation StackSets tindakan penerapan
- Tutorial: Buat aturan pemeriksaan variabel untuk pipeline sebagai kondisi entri

Tutorial: Terapkan ke instans Amazon EC2 dengan CodePipeline

Tutorial ini membantu Anda membuat tindakan penerapan CodePipeline yang menyebarkan kode Anda ke instance yang telah Anda konfigurasi di Amazon EC2.

> **Note**
>
> Sebagai bagian dari pembuatan pipa di konsol, ember artefak S3 akan digunakan oleh CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

> **Note**
>
> Tindakan EC2 penerapan hanya tersedia untuk saluran pipa tipe V2.

Prasyarat

Ada beberapa sumber daya yang harus Anda miliki sebelum Anda dapat menggunakan tutorial ini untuk membuat pipeline CD Anda. Berikut adalah hal-hal yang Anda butuhkan untuk memulai:

> **Note**
>
> Semua sumber daya ini harus dibuat dalam AWS Wilayah yang sama.

- Sebuah repositori kontrol sumber (tutorial ini menggunakan GitHub) di mana Anda akan menambahkan file sampel`script.sh`.
- Anda harus menggunakan peran CodePipeline layanan yang ada yang telah diperbarui dengan izin untuk tindakan ini. Untuk memperbarui peran layanan Anda, lihatIzin kebijakan peran layanan untuk tindakan penerapan EC2.

Setelah Anda memenuhi prasyarat ini, Anda dapat melanjutkan dengan tutorial dan membuat pipeline CD Anda.

Langkah 1: Buat instans Amazon EC2 Linux

Pada langkah ini, Anda membuat instans Amazon EC2 di mana Anda akan menerapkan aplikasi sampel. Sebagai bagian dari proses ini, buat peran instance di IAM, jika Anda belum membuat peran instance di Wilayah tempat Anda ingin membuat sumber daya.

Untuk membuat peran instance

1. Buka konsol IAM di https://console.aws.amazon.com/iam/).
2. Dari dasbor konsol, pilih Peran.
3. Pilih Buat peran.
4. Di bawah Pilih jenis entitas tepercaya, pilih Layanan AWS. Di bawah Pilih kasus penggunaan, pilih EC2. Di bawah Pilih kasus penggunaan Anda, pilih EC2. Pilih Berikutnya.
5. Cari dan pilih kebijakan bernama **`AmazonSSMManagedEC2InstanceDefaultPolicy`**.
6. Cari dan pilih kebijakan bernama **`AmazonSSMManagedInstanceCore`**. Pilih Berikutnya: Tanda.
7. Pilih Berikutnya: Tinjau. Masukkan nama untuk peran (misalnya,**`EC2InstanceRole`**).

> **Note**
>
> Catat nama peran Anda untuk langkah selanjutnya. Anda memilih peran ini saat membuat instance Anda.

> **Note**
>
> Anda akan menambahkan izin ke peran ini untuk memungkinkan akses ke bucket artefak S3 untuk pipeline Anda setelah pembuatan pipeline.

Pilih Buat peran.

Untuk meluncurkan instance

1. Buka konsol Amazon EC2 di. https://console.aws.amazon.com/ec2/
2. Dari navigasi samping, pilih Instans, dan pilih Luncurkan instance dari bagian atas halaman.
3. Di Nama, masukkan **`MyInstances`**. Ini memberikan contoh tag Key dari **`Name`** dan tag Nilai dari. **`MyInstances`**
4. Di bawah Application and OS Images (Amazon Machine Image), cari opsi Amazon Linux AMI dengan AWS logo, dan pastikan itu dipilih. (AMI ini digambarkan sebagai Amazon Linux 2 AMI (HVM) dan diberi label “Memenuhi syarat tingkat gratis”.)
5. Di bawah Jenis instans, pilih `t2.micro` tipe yang memenuhi syarat tingkat gratis sebagai konfigurasi perangkat keras untuk instans Anda.
6. Di bawah Key pair (login), pilih key pair atau buat satu.
7. Di bawah Pengaturan jaringan, pastikan statusnya Aktifkan.
8. Perluas Detail lanjutan. Di profil instans IAM, pilih peran IAM yang Anda buat di prosedur sebelumnya (misalnya,**`EC2InstanceRole`**).

   > **Note**
   >
   > Jangan biarkan peran instance kosong karena ini menciptakan peran default dan tidak memilih peran yang Anda buat.

9. Di bawah Ringkasan, di bawah Jumlah instance, masukkan2.
10. Pilih Luncurkan instans.
11. Anda dapat melihat status peluncuran di halaman Instans. Saat Anda meluncurkan instans, status awalnya adalah `pending`. Setelah instans dimulai, statusnya berubah menjadi `running`, dan ia menerima nama DNS publik. (Jika kolom DNS Publik tidak ditampilkan, pilih ikon Tampilkan/Sembunyikan, lalu pilih DNS Publik.)

Langkah 2: Tambahkan izin bucket artefak ke peran instans EC2

Anda harus memperbarui peran instans EC2 yang Anda buat untuk instans Anda agar dapat mengaksesnya ke bucket artefak pipeline Anda.

> **Note**
>
> Saat membuat instance, Anda membuat atau menggunakan peran instans EC2 yang ada. Untuk menghindari `Access Denied` kesalahan, Anda harus menambahkan izin bucket S3 ke peran instance untuk memberikan izin instance ke bucket artefak CodePipeline . Buat peran default atau perbarui peran yang ada dengan `s3:GetObject` izin yang tercakup ke bucket artefak untuk Wilayah pipeline Anda.

1. Arahkan ke pipeline Anda di CodePipeline konsol. Pilih Pengaturan. Lihat nama dan lokasi penyimpanan artefak untuk pipa yang ada. Catat ember artefak Amazon Resource Name (ARN) dan salin.
2. Arahkan ke konsol IAM dan pilih Peran. Pilih peran instance yang Anda buat di Langkah 1 dari tutorial ini.
3. Pada tab Izin, pilih Tambahkan kebijakan inline.
4. Tambahkan JSON berikut ke dokumen kebijakan, ganti nilai di `Resource` bidang dengan bucket ARN.

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

5. Pilih Perbarui.

Langkah 3: Tambahkan file skrip ke repositori Anda

Rekatkan teks contoh ini untuk membuat `script.sh` file Anda untuk langkah pasca-skrip dalam penerapan.

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

Untuk menambahkan **`script.sh`** file ke repositori sumber Anda

1. Buka editor teks lalu salin dan tempel file di atas ke file baru.

2. Komit dan dorong `script.sh` file Anda ke repositori sumber Anda.

   a. Tambahkan file.

   ```
   git add .
   ```

   b. Lakukan perubahan.

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

   c. Dorong komit.

   ```
   git push
   ```

   Catat jalur di repositori Anda.

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

Langkah 4: Membuat pipeline Anda

Gunakan CodePipeline wizard untuk membuat tahapan pipeline dan menghubungkan repositori sumber Anda.

Untuk membuat alur Anda

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyPipeline`**.
5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
6. Di peran Layanan, pilih Gunakan peran layanan yang ada, lalu pilih peran CodePipeline layanan yang telah diperbarui dengan izin yang diperlukan untuk tindakan ini. Untuk mengonfigurasi

peran CodePipeline layanan Anda untuk tindakan ini, lihatIzin kebijakan peran layanan untuk tindakan penerapan EC2.

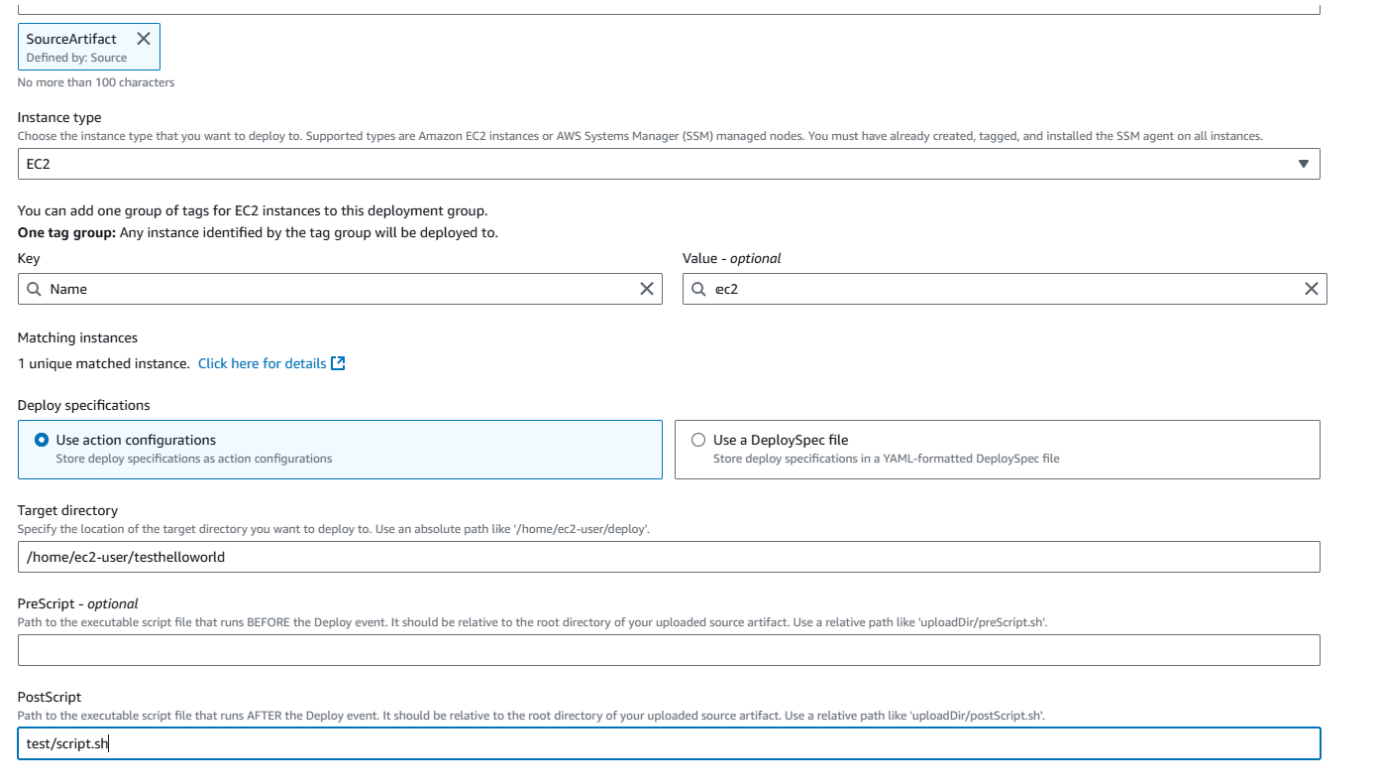
7. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.
8. Pada Langkah 3: Tambahkan halaman tahap sumber, tambahkan tahap sumber:

   a. Di penyedia Sumber, pilih GitHub (melalui GitHub Aplikasi).

   b. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.

   c. Dalam nama Repositori, pilih nama repositori Anda GitHub .

   Pilih Berikutnya.
9. Pada Langkah 4: Tambahkan halaman tahap build, pilih Lewati.
10. Pada Langkah 5: Tambahkan halaman tahap penerapan, pilih EC2.



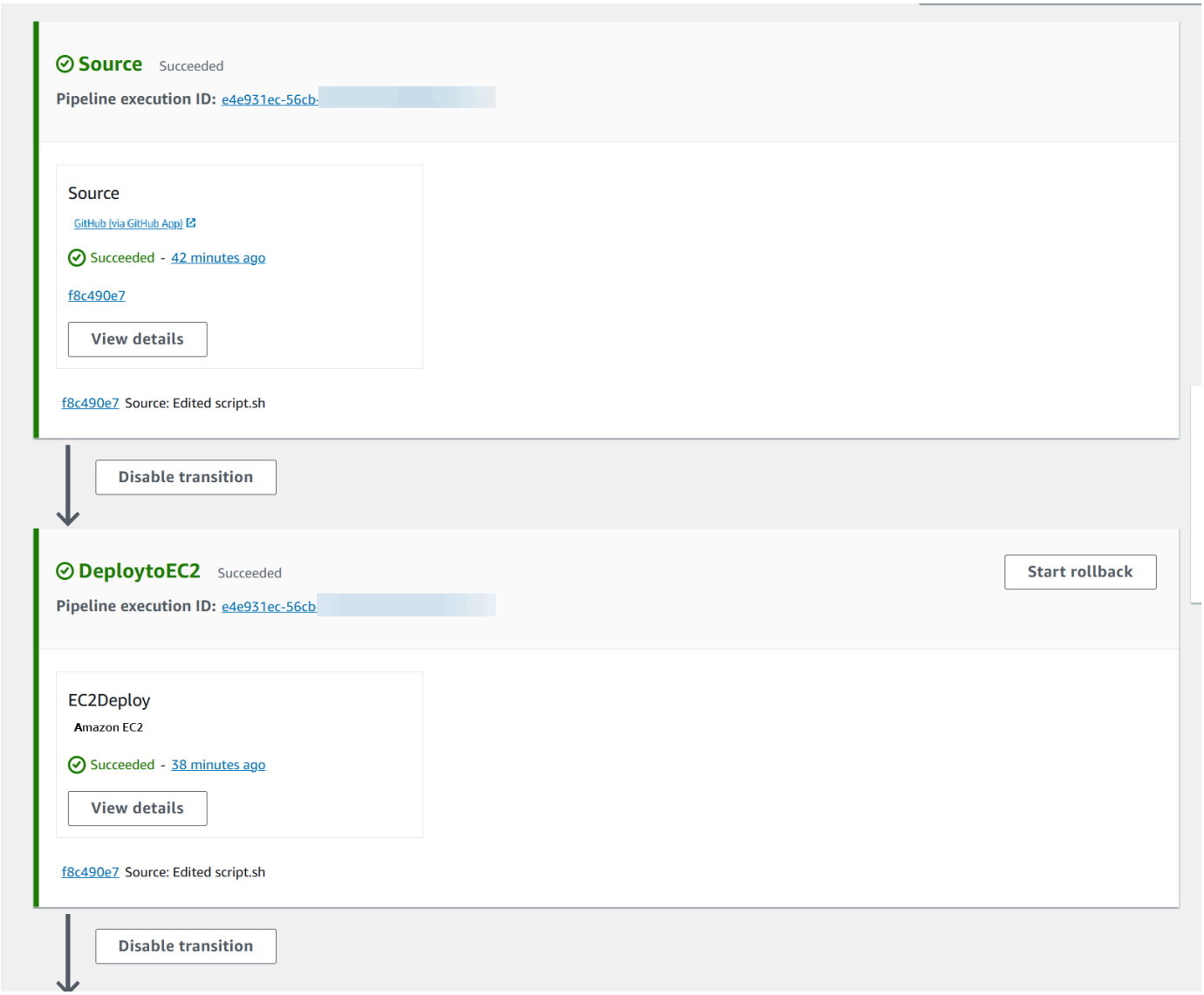
   a. Untuk direktori Target, masukkan direktori pada instance yang ingin Anda gunakan, seperti/home/ec2-user/testhelloworld.

> **Note**
>
> Tentukan direktori penyebaran yang Anda inginkan untuk digunakan tindakan pada instance. Tindakan akan mengotomatiskan pembuatan direktori yang ditentukan pada instance sebagai bagian dari penerapan.

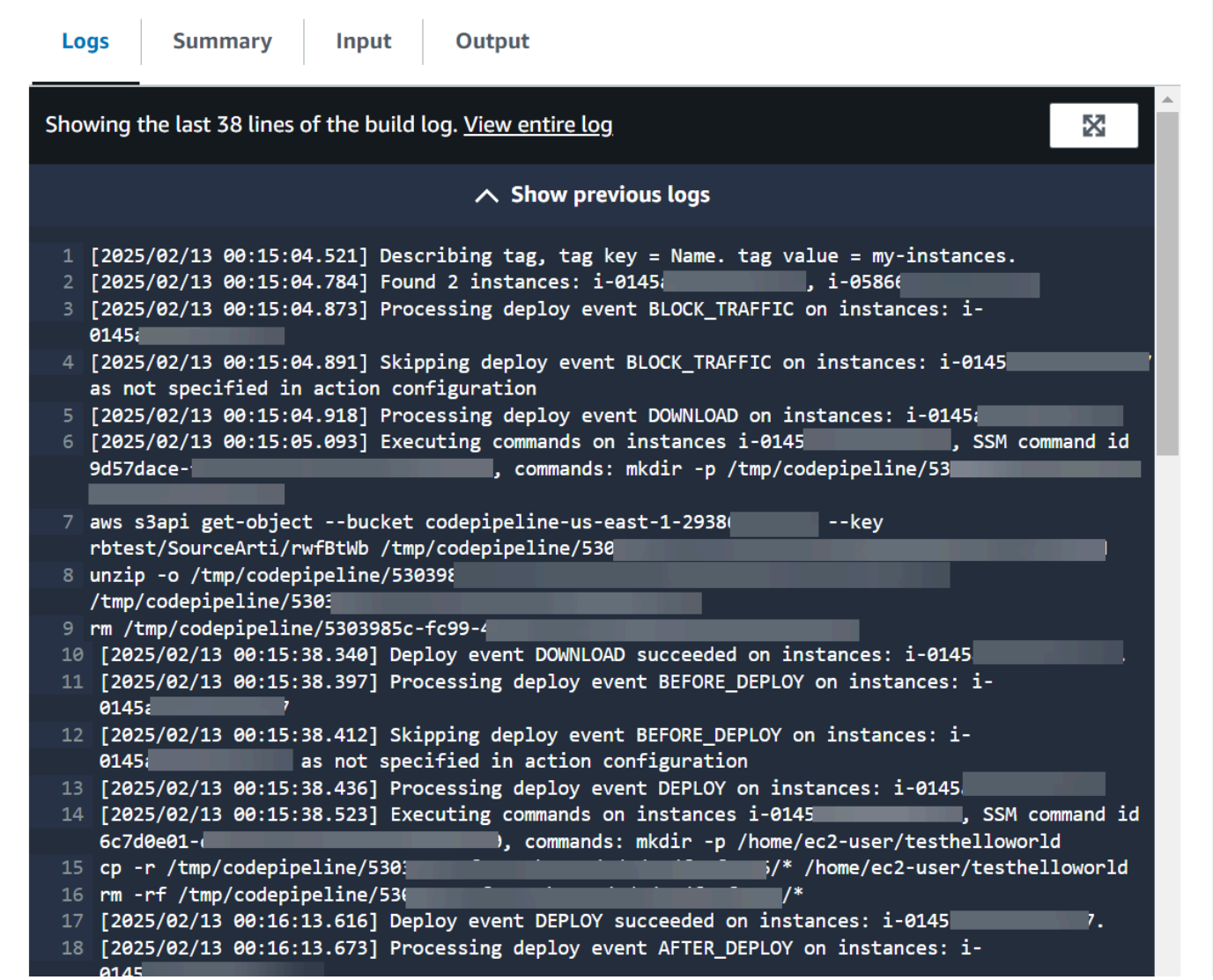
b. Untuk PostScript, masukkan jalur dan nama file untuk skrip Anda, seperti`test/script.sh`.

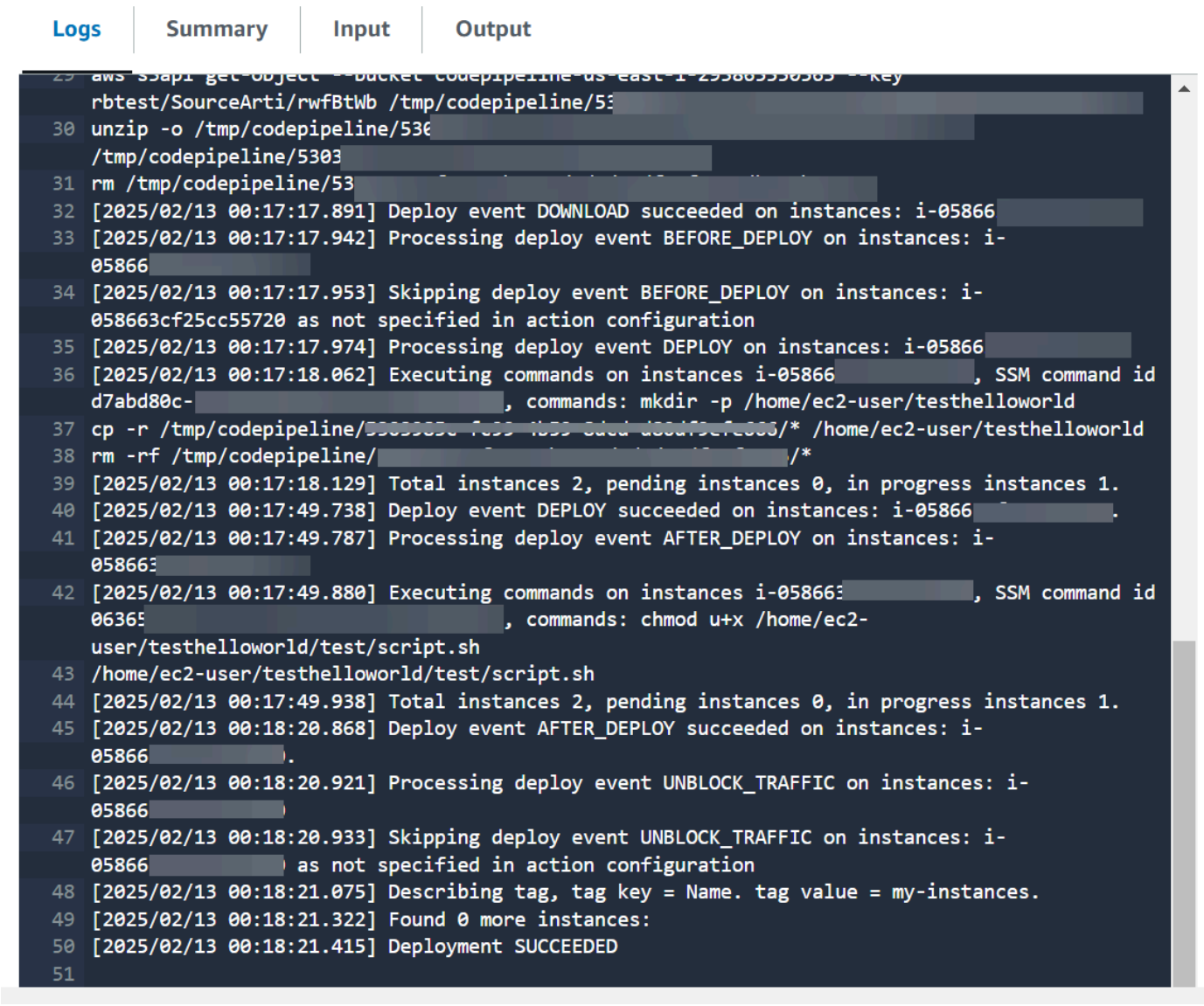
c. Pilih Berikutnya.

11. Pada halaman Langkah 6: Tinjau, tinjau konfigurasi pipeline Anda dan pilih Buat pipeline untuk membuat pipeline.



12. Setelah pipeline berhasil berjalan, pilih Lihat detail untuk melihat log pada tindakan untuk melihat keluaran tindakan komputasi terkelola.





Langkah 5: Uji Pipa Anda

Pipeline Anda harus memiliki segalanya untuk menjalankan penerapan AWS berkelanjutan end-to-end asli. Sekarang, uji fungsinya dengan mendorong perubahan kode ke repositori sumber Anda.

Untuk menguji pipa Anda

1. Buat perubahan kode ke repositori sumber yang dikonfigurasi, komit, dan dorong perubahan.
2. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
3. Pilih pipeline Anda dari daftar.

4. Perhatikan kemajuan pipa melalui tahapannya. Pipeline Anda harus selesai dan tindakan Anda menerapkan skrip pada instance Anda.
5. Untuk informasi pemecahan masalah selengkapnya, lihat Tindakan Penerapan EC2 gagal dengan pesan kesalahan `No such file`.

Tutorial: Buat dan dorong gambar Docker ke Amazon ECR dengan CodePipeline (tipe V2)

Tutorial ini membantu Anda membuat tindakan build CodePipeline yang berjalan dan mendorong image Docker Anda ke Amazon ECR setelah perubahan pada kode sumber Anda. Tutorial ini juga menunjukkan kepada Anda cara menambahkan tindakan penerapan Amazon ECS yang menyebarkan gambar yang Anda dorong.

Important

Sebagai bagian dari pembuatan pipa di konsol, ember artefak S3 akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Note

Tutorial ini adalah untuk tindakan ECRBuild AndPublish build untuk CodePipeline pipeline dengan repositori GitHub sumber dan tindakan standar Amazon ECS untuk menerapkan ke cluster Amazon ECS. Untuk tutorial yang menggunakan pipeline dengan repositori gambar ECR sebagai sumber untuk tindakan penerapan Amazon ECS ke CodeDeploy biru/hijau, lihat. CodePipeline Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR

> **Important**
>
> Tindakan ini menggunakan CodeBuild komputasi CodePipeline terkelola untuk menjalankan perintah di lingkungan build. Menjalankan tindakan perintah akan dikenakan biaya terpisah di AWS CodeBuild.

Prasyarat

Ada beberapa sumber daya yang harus Anda miliki sebelum Anda dapat menggunakan tutorial ini untuk membuat pipeline CD Anda. Berikut adalah hal-hal yang Anda butuhkan untuk memulai:

> **Note**
>
> Semua sumber daya ini harus dibuat dalam AWS Wilayah yang sama.

- Sebuah repositori kontrol sumber (tutorial ini menggunakan GitHub) di mana Anda akan menambahkan berikut untuk tutorial ini:
  - Pada Langkah 1, Anda akan menambahkan contoh Dockerfile ke repositori sumber Anda sebagai artefak input untuk tindakan build di. ECRBuild AndPublish CodePipeline
  - Pada Langkah 2, Anda akan menambahkan contoh file imagedefinitions.json ke repositori sumber Anda sebagai persyaratan untuk tindakan penerapan standar Amazon ECS di. CodePipeline
- Repositori gambar Amazon ECR yang berisi gambar yang telah Anda buat dari Dockerfile Anda. Untuk informasi selengkapnya, lihat Membuat Repositori dan Mendorong Gambar di Panduan Pengguna Amazon Elastic Container Registry.
- Cluster dan layanan Amazon ECS yang dibuat di Wilayah yang sama dengan repositori gambar. Untuk informasi selengkapnya, lihat Membuat Cluster dan Membuat Layanan di Panduan Pengembang Layanan Amazon Elastic Container.

Setelah Anda memenuhi prasyarat ini, Anda dapat melanjutkan dengan tutorial dan membuat pipeline CD Anda.

Langkah 1: Tambahkan Dockerfile ke repositori sumber Anda

Tutorial ini menggunakan ECRBuild AndPublish tindakan untuk membangun gambar Docker Anda dan mendorong gambar ke Amazon ECR. Tindakan komputasi terkelola CodePipeline digunakan CodeBuild untuk menjalankan perintah untuk login ECR dan push gambar. Anda tidak perlu menambahkan `buildspec.yml` file ke repositori kode sumber Anda untuk memberi tahu CodeBuild cara melakukannya. Anda hanya menyediakan Dockerfile di repositori Anda sebagai berikut untuk contoh ini.

Tempel teks contoh ini untuk membuat `Dockerfile` file Anda. Sampel Dockerfile ini sama dengan sampel yang digunakan dalam instruksi gambar ECR dalam prasyarat.

```
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
```

Untuk menambahkan **`Dockerfile`** file ke repositori sumber Anda

1. Buka editor teks dan kemudian salin dan tempel Dockerfile di atas ke file baru.
2. Komit dan dorong `Dockerfile` file Anda ke repositori sumber Anda.

   a. Tambahkan file.

   ```
   git add .
   ```

   b. Lakukan perubahan.

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

c. Dorong komit.

```
git push
```

Pastikan untuk menempatkan file di tingkat root repositori Anda.

```
/ Dockerfile
```

Langkah 2: Tambahkan file imagedefinitions.json ke repositori sumber Anda

Tutorial ini menggunakan tindakan penerapan standar Amazon ECS untuk menyebarkan container Anda CodePipeline ke cluster Amazon ECS Anda. Tindakan penerapan standar Amazon ECS memerlukan file imagedefinitions.json yang berisi nama gambar dan URI Anda. Untuk informasi selengkapnya tentang file imagedefinitions.json, lihat. file imagedefinitions.json untuk tindakan penerapan standar Amazon ECS

Tempel teks contoh ini untuk membuat `imagedefinitions.json` file Anda. Gunakan nama di Dockerfile Anda, sepertihello-world, dan gunakan URI dari repositori Amazon ECR tempat gambar disimpan.

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

Untuk menambahkan **`imagedefinitions.json`** file ke repositori sumber Anda

1. Buka editor teks lalu salin dan tempel contoh di atas ke file baru.
2. Komit dan dorong `imagedefinitions.json` file Anda ke repositori sumber Anda.

   a. Tambahkan file.

```
git add .
```

b. Lakukan perubahan.

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

c. Dorong komit.

```
git push
```

Pastikan untuk menempatkan file di tingkat root repositori Anda.

```
/ imagedefinitions.json
```

Langkah 3: Membuat pipeline Anda

Gunakan CodePipeline wizard untuk membuat tahapan pipeline dan menghubungkan repositori sumber Anda.

Untuk membuat alur Anda

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**MyPipeline**.
5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
6. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.
7. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.
8. Pada Langkah 3: Tambahkan halaman tahap sumber, tambahkan tahap sumber:

   a. Di penyedia Sumber, pilih GitHub (melalui GitHub Aplikasi).

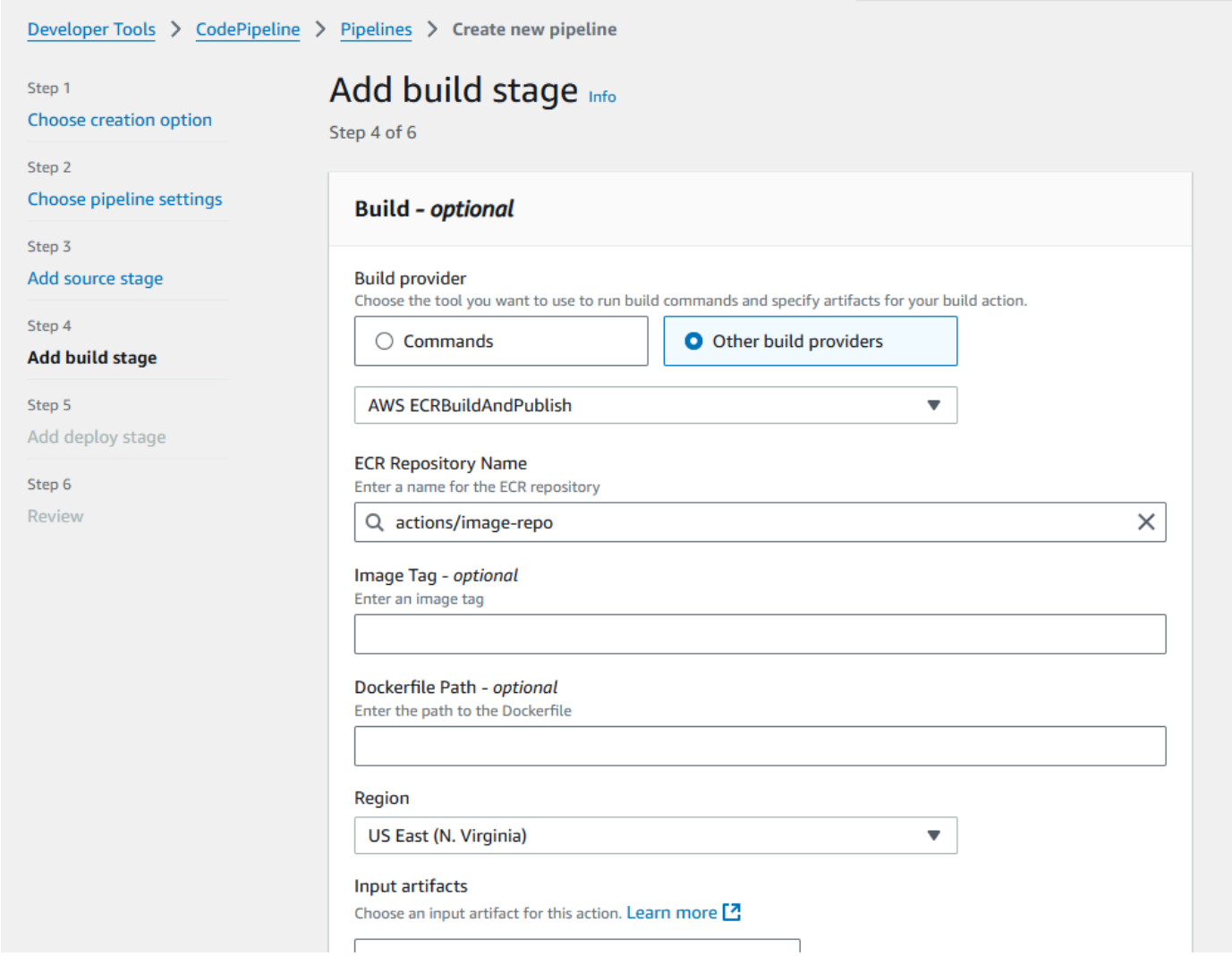
b. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.

c. Dalam nama Repositori, pilih nama repositori Anda GitHub .

d. Di cabang Default, pilih cabang yang ingin Anda tentukan saat pipeline dimulai secara manual atau dengan peristiwa sumber yang bukan tag Git. Jika sumber perubahan bukan pemicu atau jika eksekusi pipeline dimulai secara manual, maka perubahan yang digunakan akan menjadi komit HEAD dari cabang default.

Pilih Berikutnya.

9. Pada Langkah 4: Tambahkan halaman tahap build, pilih Penyedia build lain pilih ECRBuildAndPublish.



a. Untuk nama repositori ECR, pilih repositori gambar Anda.

b. Pilih Berikutnya.

10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

11. Pada Langkah 6: Tambahkan halaman tahap penerapan, pilih Lewati tahap penerapan. Anda akan menambahkan tindakan ECS pada langkah berikut.
12. Pada halaman Langkah 7: Tinjau, tinjau konfigurasi pipeline Anda dan pilih Buat pipeline untuk membuat pipeline.
13. Edit pipeline Anda untuk menambahkan tindakan penerapan Amazon ECS ke pipeline Anda:

    a. Di kanan atas, pilih Edit.

    b. Di bagian bawah diagram, pilih + Tambahkan tahap. Dalam nama Panggung, masukkan nama, seperti**`Deploy`**.

    c. Pilih + Tambahkan grup tindakan.

    d. Di Nama tindakan, masukkan nama.

    e. Di penyedia Action, pilih Amazon ECS. Izinkan Wilayah ke default ke Wilayah alur.

    f. Dalam artefak Input, pilih artefak input dari tahap sumber, seperti. `SourceArtifact`

    g. Untuk nama Cluster, pilih klaster Amazon ECS tempat layanan Anda berjalan.

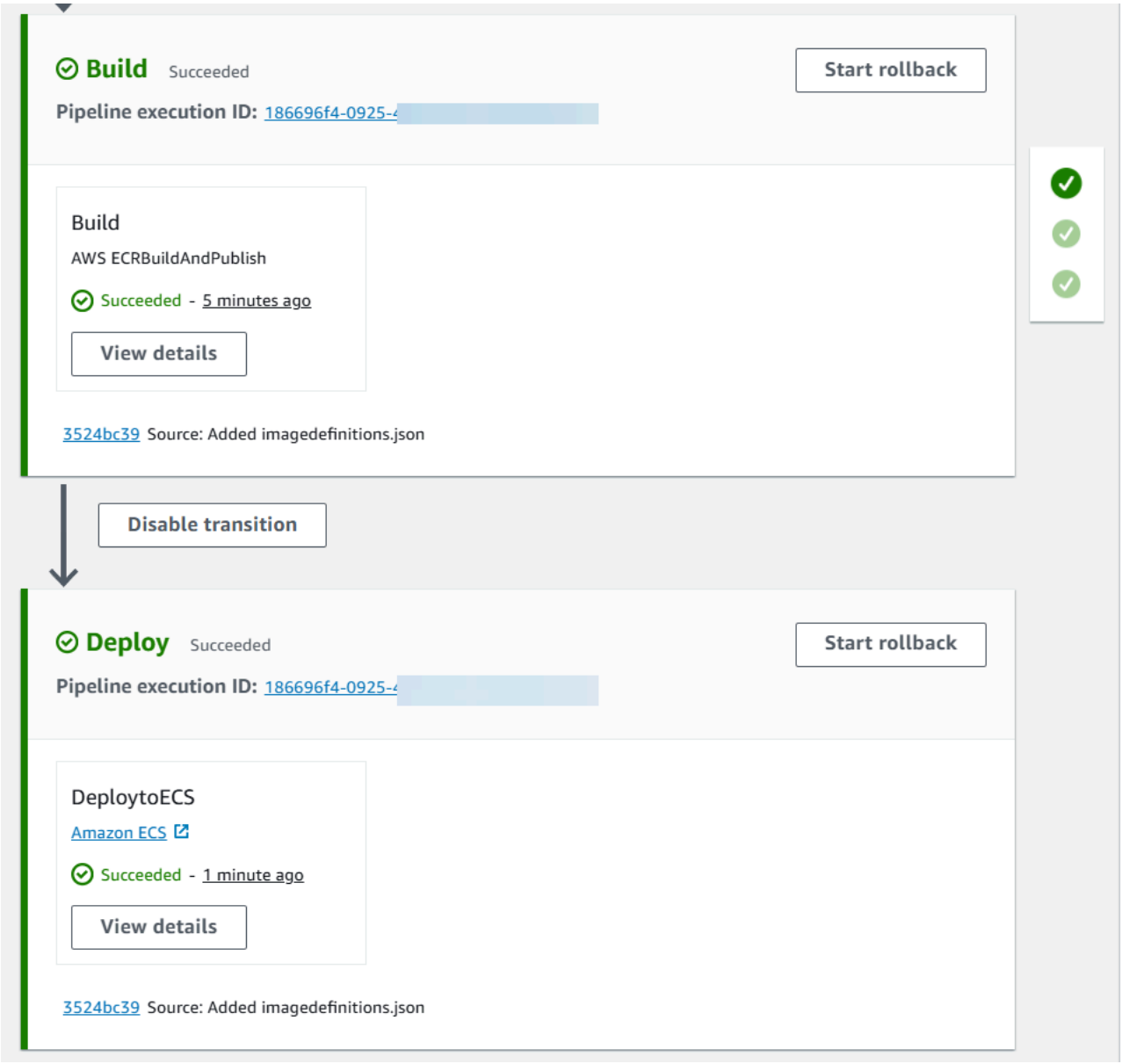
    h. Untuk nama Layanan, pilih layanan yang akan diperbarui.

    i. Pilih Simpan.

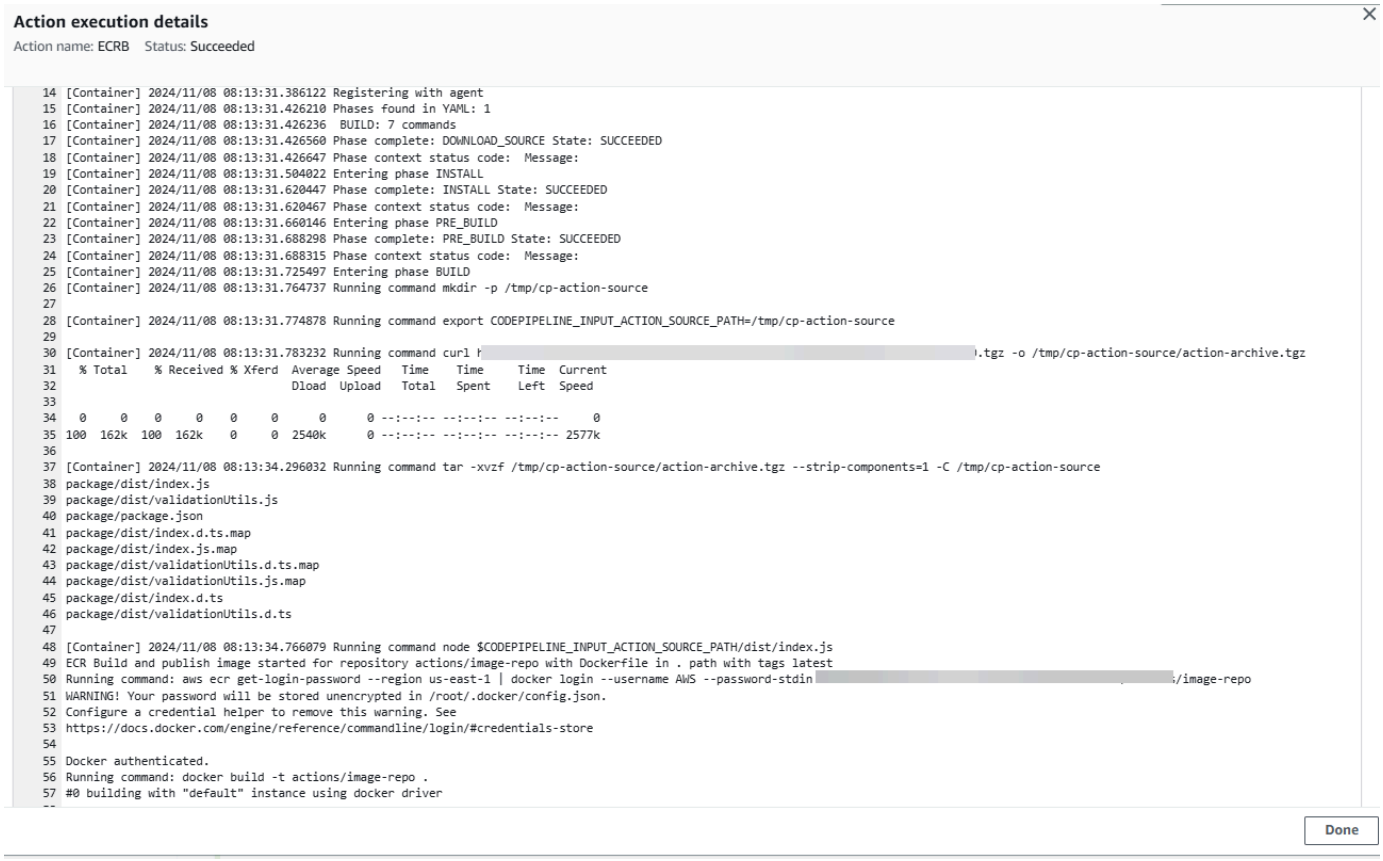
    j. Di panggung yang Anda edit, pilih Selesai. Di AWS CodePipeline panel, pilih Simpan, lalu pilih Simpan pada pesan peringatan.

    k. Untuk mengirimkan perubahan dan memulai pembuatan pipeline, pilih Rilis perubahan, lalu pilih Rilis.

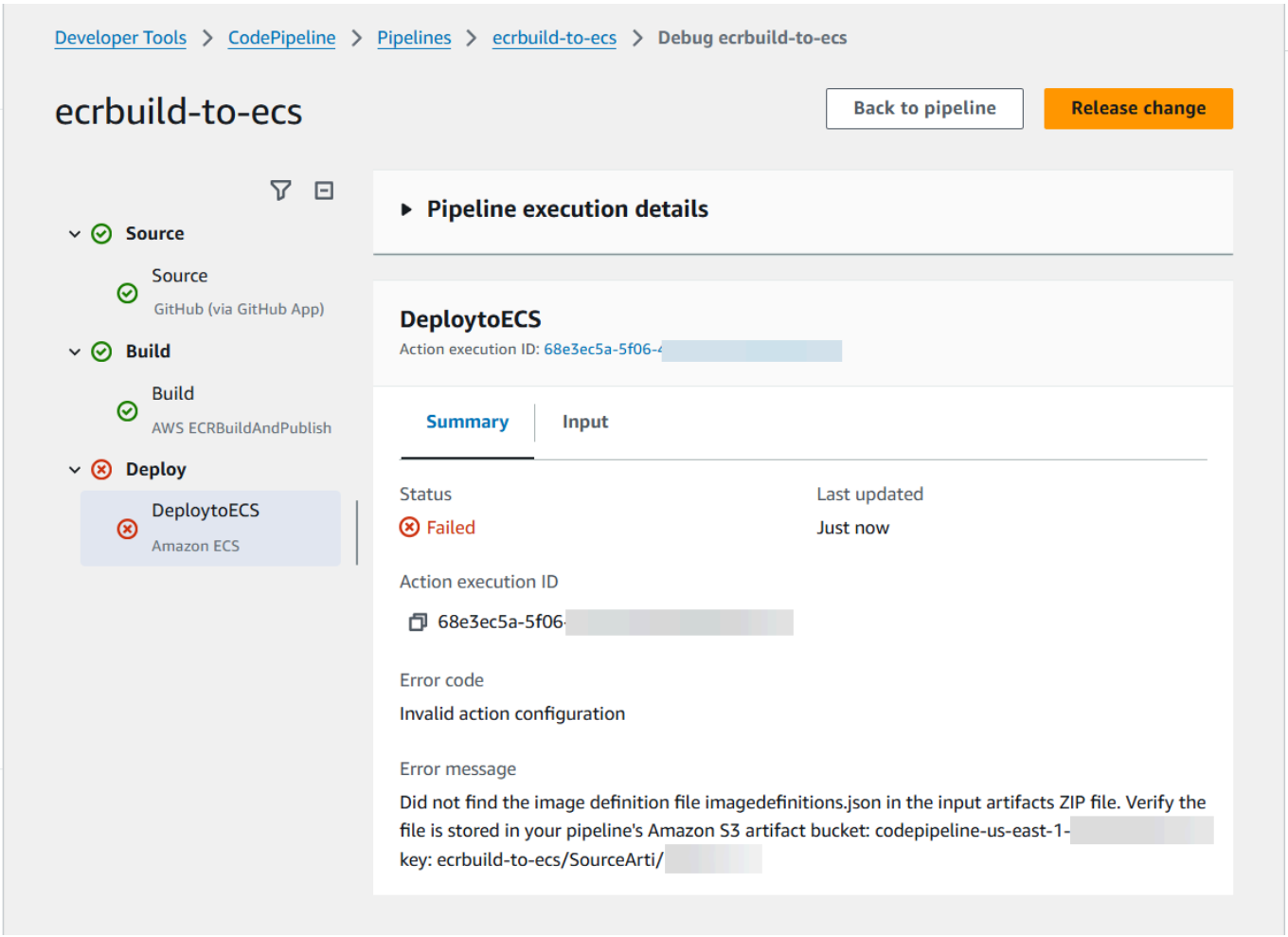
14. Setelah pipa berjalan, lihat struktur dan status pipa.



15. Setelah pipeline berhasil berjalan, pilih Lihat detail untuk melihat log pada tindakan untuk melihat keluaran tindakan komputasi terkelola.



16. Memecahkan masalah tindakan yang gagal. Misalnya, tindakan penerapan ECS dapat gagal jika file imagedefinitions.json tidak ada di repositori sumber. Berikut ini adalah contoh pesan kesalahan yang ditampilkan ketika file imagedefinitions.json hilang.



Langkah 4: Uji Pipa Anda

Pipeline Anda harus memiliki segalanya untuk menjalankan penerapan AWS berkelanjutan end-to-end asli. Sekarang, uji fungsinya dengan mendorong perubahan kode ke repositori sumber Anda.

Untuk menguji pipa Anda

1. Buat perubahan kode ke repositori sumber yang dikonfigurasi, komit, dan dorong perubahan.
2. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
3. Pilih pipeline Anda dari daftar.
4. Perhatikan kemajuan pipa melalui tahapannya. Pipeline Anda harus selesai dan tindakan Anda mendorong image Docker ke ECR yang dibuat dari perubahan kode Anda.

Tutorial: Terapkan ke Amazon EKS dengan CodePipeline

Tutorial ini membantu Anda membuat tindakan penerapan CodePipeline yang menyebarkan kode Anda ke cluster yang telah Anda konfigurasi di Amazon EKS.

Tindakan EKS mendukung kluster EKS publik dan swasta. Cluster pribadi adalah tipe yang direkomendasikan oleh EKS; Namun, kedua jenis didukung.

> **Note**
>
> Sebagai bagian dari pembuatan pipa di konsol, ember artefak S3 akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

> **Note**
>
> Tindakan ini menggunakan CodeBuild komputasi CodePipeline terkelola untuk menjalankan perintah di lingkungan build. Menjalankan tindakan perintah akan dikenakan biaya terpisah di AWS CodeBuild.

> **Note**
>
> Tindakan EKS penerapan hanya tersedia untuk saluran pipa tipe V2.

Prasyarat

Ada beberapa sumber daya yang harus Anda miliki sebelum Anda dapat menggunakan tutorial ini untuk membuat pipeline CD Anda. Berikut adalah hal-hal yang Anda butuhkan untuk memulai:

> **Note**
>
> Semua sumber daya ini harus dibuat dalam AWS Wilayah yang sama.

- Sebuah repositori kontrol sumber (tutorial ini menggunakan GitHub) di mana Anda akan menambahkan file sampel`deployment.yaml`.
- Anda harus menggunakan peran CodePipeline layanan yang ada yang akan Anda perbarui dengan izin untuk tindakan ini menggunakan Langkah 3: Perbarui kebijakan peran CodePipeline layanan di IAM di bawah ini. Izin yang diperlukan didasarkan pada jenis cluster yang Anda buat. Untuk informasi selengkapnya, lihat Izin kebijakan peran layanan.
- Gambar kerja dan tag repositori yang telah Anda dorong ke ECR atau repositori gambar Anda.

Setelah Anda memenuhi prasyarat ini, Anda dapat melanjutkan dengan tutorial dan membuat pipeline CD Anda.

Langkah 1: (Opsional) Buat cluster di Amazon EKS

Anda dapat memilih untuk membuat cluster EKS dengan endpoint publik atau pribadi.

Pada langkah-langkah berikut, Anda membuat klaster publik atau pribadi di EKS. Langkah ini opsional jika Anda telah membuat cluster Anda.

Buat klaster publik di Amazon EKS

Pada langkah ini, Anda membuat cluster di EKS.

Buat klaster publik

1. Buka konsol EKS, lalu pilih Create cluster.
2. Dalam Nama, beri nama cluster Anda. Pilih Berikutnya.
3. Pilih Buat.

Buat klaster pribadi di Amazon EKS

Jika Anda memilih untuk membuat cluster dengan titik akhir pribadi, pastikan untuk melampirkan subnet pribadi saja, dan pastikan mereka memiliki koneksi internet.

Ikuti lima sub-langkah berikutnya untuk membuat cluster dengan endpoint pribadi.

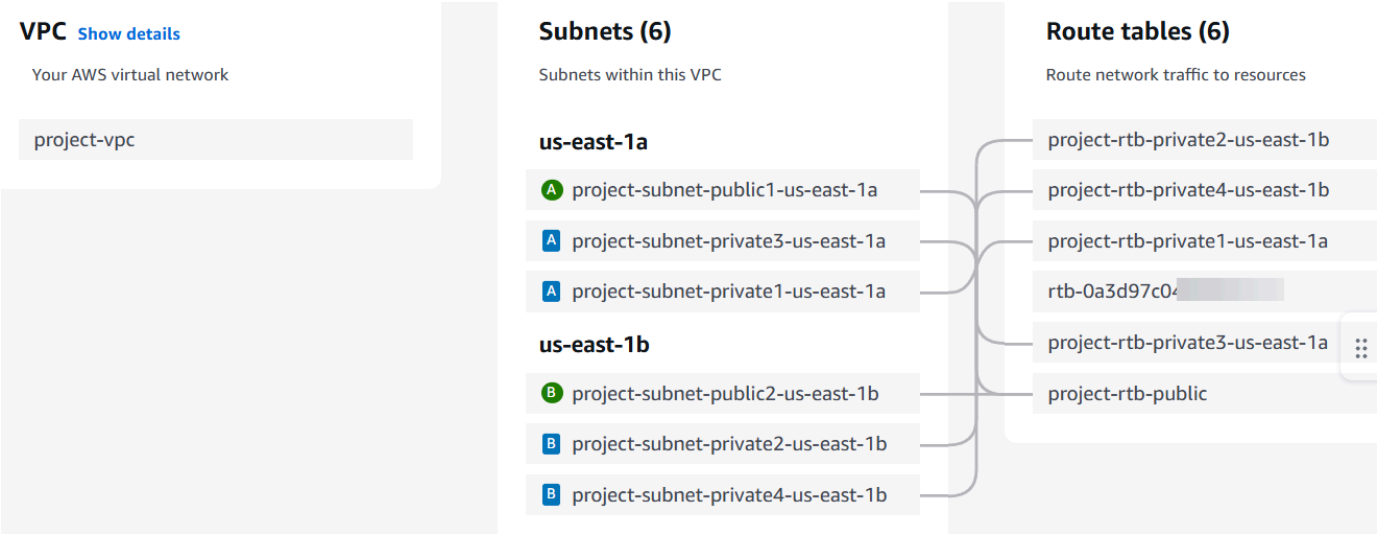
Buat VPC di konsol

1. Buka konsol VPC, lalu pilih Buat VPC.
2. Di bawah pengaturan VPC, pilih VPC dan lainnya.

3. Pilih untuk membuat satu subnet publik dan 4 pribadi. Pilih Buat VPC.
4. Pada halaman subnet, pilih Private.

Tentukan subnet pribadi di VPC Anda

1. Arahkan ke VPC Anda dan pilih ID VPC untuk membuka halaman detail VPC.
2. Pada halaman detail VPC, pilih tab Peta sumber daya.
3. Lihat diagram dan catat subnet pribadi Anda. Subnet ditampilkan dengan label untuk menunjukkan status publik atau pribadi, dan setiap subnet dipetakan ke tabel rute.



   Perhatikan bahwa klaster pribadi akan memiliki semua subnet pribadi.
4. Buat subnet publik untuk meng-host gateway NAT. Anda dapat melampirkan hanya satu gateway internet untuk satu VPC pada suatu waktu.

Buat gateway NAT di subnet publik

1. Di subnet publik, buat gateway NAT. Arahkan ke konsol VPC, lalu pilih gateway Internet. Pilih Buat gateway internet.
2. Di Nama, masukkan nama untuk gateway internet Anda. Pilih Buat gateway internet.

Perbarui tabel rute untuk subnet pribadi untuk mengarahkan lalu lintas ke gateway NAT.

Tambahkan gateway NAT ke tabel rute Anda untuk subnet pribadi Anda

1. Arahkan ke konsol VPC, lalu pilih Subnet.

2. Untuk setiap subnet pribadi, pilih dan kemudian pilih tabel rute untuk subnet tersebut di halaman detail, Pilih Edit tabel rute.
3. Perbarui tabel rute untuk subnet pribadi untuk mengarahkan lalu lintas internet ke gateway NAT. Pilih Tambahkan rute. Pilih gateway NAT dari opsi yang akan ditambahkan. Pilih gateway internet yang Anda buat.
4. Untuk subnet publik, buat tabel rute kustom. Verifikasi bahwa daftar kontrol akses jaringan (ACL) untuk subnet publik Anda memungkinkan lalu lintas masuk dari subnet pribadi.
5. Pilih Simpan perubahan.

Pada langkah ini, Anda membuat cluster di EKS.

Buat klaster pribadi

1. Buka konsol EKS, lalu pilih Create cluster.
2. Dalam Nama, beri nama cluster Anda. Pilih Berikutnya.
3. Tentukan VPC Anda dan informasi konfigurasi lainnya. Pilih Buat.

Kluster EKS Anda dapat berupa klaster publik atau pribadi. Langkah ini untuk cluster yang HANYA memiliki titik akhir pribadi. Pastikan bahwa jika cluster Anda bersifat pribadi.

Langkah 2: Konfigurasikan klaster pribadi Anda di Amazon EKS

Langkah ini hanya berlaku jika Anda telah membuat cluster pribadi. Langkah ini untuk cluster yang HANYA memiliki titik akhir pribadi.

Konfigurasikan klaster Anda

1. Lampirkan subnet pribadi hanya di cluster EKS di bawah tab Networking. Pasang subnet pribadi yang ditangkap di Tentukan subnet pribadi di bagian VPC Anda di bawah. Langkah 1: (Opsional) Buat cluster di Amazon EKS
2. Pastikan subnet pribadi memiliki akses ke internet karena CodePipeline menyimpan dan mengambil artefak dari bucket artefak S3 untuk pipeline Anda.

Langkah 3: Perbarui kebijakan peran CodePipeline layanan di IAM

Pada langkah ini, Anda akan memperbarui peran CodePipeline layanan yang ada, seperti**cp-service-role**, dengan izin yang diperlukan oleh CodePipeline untuk terhubung dengan cluster Anda. Jika Anda tidak memiliki peran yang ada, buat yang baru.

Perbarui peran CodePipeline layanan Anda dengan langkah-langkah berikut.

Untuk memperbarui kebijakan peran CodePipeline layanan

1. Buka konsol IAM di https://console.aws.amazon.com/iam/).
2. Dari dasbor konsol, pilih Peran.
3. Cari peran CodePipeline layanan Anda, seperti**cp-service-role**.
4. Tambahkan kebijakan inline baru.
5. Di editor Kebijakan, masukkan yang berikut ini.

   - Untuk klaster publik, tambahkan izin berikut.

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

- Untuk klaster pribadi, tambahkan izin berikut. Cluster pribadi akan memerlukan izin tambahan untuk VPC Anda, jika berlaku.

  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. Pilih Perbarui kebijakan.

Langkah 4: Buat entri akses untuk peran CodePipeline layanan

Pada langkah ini, Anda membuat entri akses di klaster yang akan menambahkan peran CodePipeline layanan yang diperbarui di Langkah 3, bersama dengan kebijakan akses terkelola.

1. Buka konsol EKS dan arahkan ke cluster Anda.
2. Pilih tab Access.
3. Di bawah entri akses IAM, pilih Buat entri akses.
4. Di ARN utama IAM, masukkan peran yang baru saja Anda perbarui untuk tindakan tersebut, seperti. **`cp-service-role`** Pilih Berikutnya.
5. Pada Langkah 2: Tambahkan halaman kebijakan akses, di Nama kebijakan, pilih kebijakan terkelola untuk akses, misalnya`AmazonEKSClusterAdminPolicy`. Pilih Tambahkan kebijakan. Pilih Berikutnya.

> **Note**
>
> Ini adalah kebijakan yang digunakan CodePipeline tindakan untuk berbicara dengan Kubernetes. Sebagai praktik terbaik, untuk mengurangi izin dalam kebijakan Anda dengan hak istimewa paling sedikit daripada kebijakan administratif, lampirkan kebijakan khusus sebagai gantinya.

6. Pada halaman ulasan, pilih Buat.

Langkah 5: Buat repositori sumber dan tambahkan file konfigurasi **`helm chart`**

Pada langkah ini, Anda membuat file konfigurasi yang sesuai untuk tindakan Anda (file manifes Kubernetes atau bagan Helm) dan menyimpan file konfigurasi di repositori sumber Anda. Gunakan file yang sesuai untuk konfigurasi Anda. Untuk informasi lebih lanjut, lihat https://kubernetes. io/docs/reference/kubectl/quick-referensi/ atau https://helm. sh/docs/topics/charts/.

- Untuk Kubernetes, gunakan file manifes.
- Untuk Helm, gunakan bagan Helm.

1. Buat atau gunakan GitHub repositori yang ada.
2. Buat struktur baru di repositori Anda untuk file bagan helm Anda seperti yang ditunjukkan pada contoh di bawah ini.

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

3. Tambahkan file ke tingkat root repositori Anda.

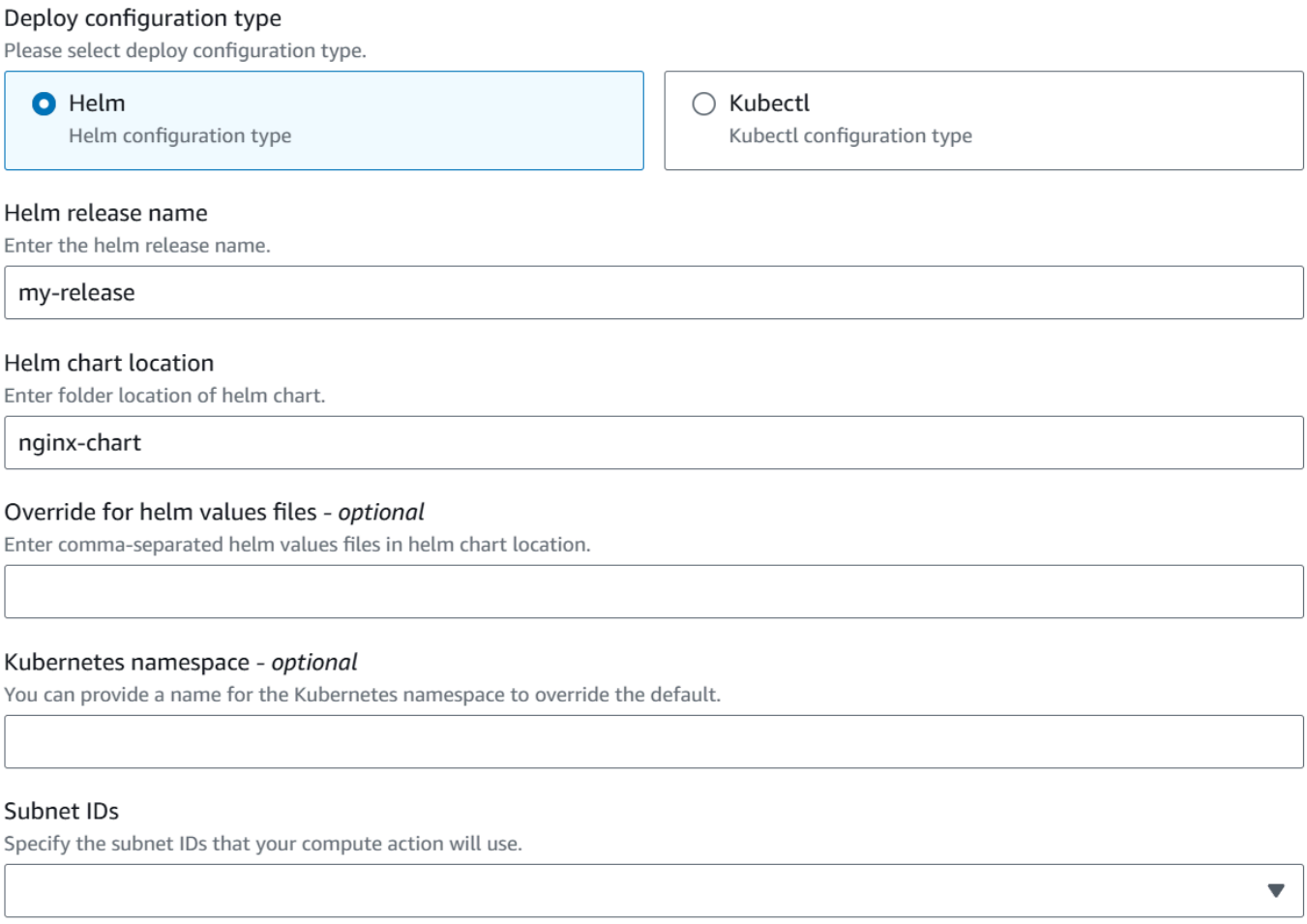
Langkah 6: Membuat pipeline Anda

Gunakan CodePipeline wizard untuk membuat tahapan pipeline dan menghubungkan repositori sumber Anda.

Untuk membuat alur Anda

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.

3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyEKSPipeline`**.
5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
6. Di peran Layanan, pilih peran layanan yang Anda perbarui di Langkah 3.
7. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.
8. Pada Langkah 3: Tambahkan halaman tahap sumber, untuk penyedia Sumber, pilih untuk membuat koneksi ke GitHub repositori Anda.
9. Pada Langkah 4: Tambahkan halaman tahap build, pilih Lewati.
10. Pada Langkah 5: Tambahkan halaman tahap penerapan, pilih Amazon EKS.

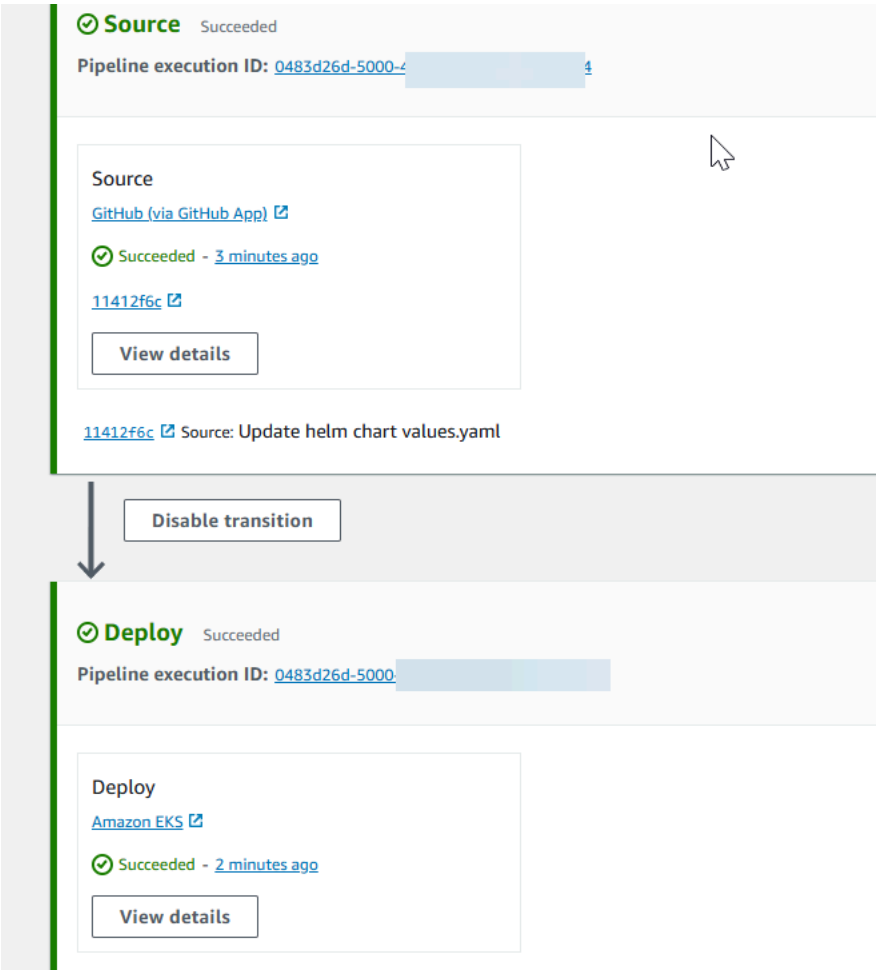


    a. Di bawah Menyebarkan jenis konfigurasi, pilih Helm.

b. Di lokasi bagan Helm, masukkan nama rilis, sepertimy-`release`. Untuk lokasi bagan Helm, masukkan jalur untuk file bagan helm Anda, seperti. `mychart`

c. Pilih Berikutnya.

11. Pada halaman Langkah 6: Tinjau, tinjau konfigurasi pipeline Anda dan pilih Buat pipeline untuk membuat pipeline.



12. Setelah pipeline berhasil berjalan, pilih Lihat detail untuk melihat log pada tindakan untuk melihat output tindakan.

Tutorial: Buat pipeline yang menjalankan perintah dengan komputasi (tipe V2)

Dalam tutorial ini, Anda mengonfigurasi pipeline yang terus menjalankan perintah build yang disediakan menggunakan tindakan Command dalam tahap build. Untuk informasi selengkapnya tentang tindakan Perintah, lihatReferensi tindakan perintah.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Prasyarat

Anda harus sudah memiliki yang berikut:

- Sebuah GitHub repositori. Anda dapat menggunakan GitHub repositori yang Anda buat. Tutorial: Gunakan klon lengkap dengan sumber GitHub pipa

Langkah 1: Buat file sumber dan dorong ke GitHub repositori Anda

Di bagian ini, Anda membuat dan mendorong file sumber contoh Anda ke repositori yang digunakan pipeline untuk tahap sumber Anda. Untuk contoh ini, Anda menghasilkan dan mendorong yang berikut:

- File `README.txt`.

Untuk membuat file sumber

1. Buat file dengan teks berikut:

   ```
   Sample readme file
   ```

2. Simpan file sebagai `README.txt`.

Untuk mendorong file ke GitHub repositori Anda

1. Dorong atau unggah file ke repositori Anda. File-file ini adalah artefak sumber yang dibuat oleh wizard Create Pipeline untuk tindakan penerapan Anda. AWS CodePipeline File Anda akan terlihat seperti ini di direktori lokal Anda:

```
README.txt
```

2. Untuk menggunakan baris perintah Git dari repositori kloning di komputer lokal Anda:

   a. Jalankan perintah berikut untuk mementaskan semua file Anda sekaligus:

   ```
   git add -A
   ```

   b. Jalankan perintah berikut untuk mengkomit file dengan pesan komit.

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

   c. Jalankan perintah berikut untuk mendorong file dari repo lokal Anda ke repositori Anda:

   ```
   git push
   ```

Langkah 2: Buat alur Anda

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan tindakan GitHub (via GitHub App) untuk repositori tempat file sumber disimpan.
- Sebuah tahap build dengan tindakan Commands.

Untuk membuat alur dengan wizard

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyCommandsPipeline`**.
5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.

6. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.

   > **Note**
   >
   > Jika Anda menggunakan peran layanan yang ada, untuk menggunakan tindakan Perintah, Anda perlu menambahkan izin berikut untuk peran layanan. Cakupkan izin ke tingkat sumber daya pipeline dengan menggunakan izin berbasis sumber daya dalam pernyataan kebijakan peran layanan. Untuk informasi selengkapnya, lihat contoh kebijakan diIzin kebijakan peran layanan.
   >
   > - log: CreateLogGroup
   > - log: CreateLogStream
   > - log: PutLogEvents

7. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.
8. Pada Langkah 3: Tambahkan halaman tahap sumber, tambahkan tahap sumber:

   a. Di penyedia Sumber, pilih GitHub (melalui GitHub Aplikasi).

   b. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.

   c. Dalam nama Repositori, pilih nama GitHub repositori.com Anda.

   d. Di cabang Default, pilih cabang yang ingin Anda tentukan saat pipeline dimulai secara manual atau dengan peristiwa sumber yang bukan tag Git. Jika sumber perubahan bukan pemicu atau jika eksekusi pipeline dimulai secara manual, maka perubahan yang digunakan akan menjadi komit HEAD dari cabang default. Secara opsional, Anda juga dapat menentukan webhook dengan pemfilteran (pemicu). Untuk informasi selengkapnya, lihat Otomatiskan saluran pipa awal menggunakan pemicu dan penyaringan.

   Pilih Berikutnya.

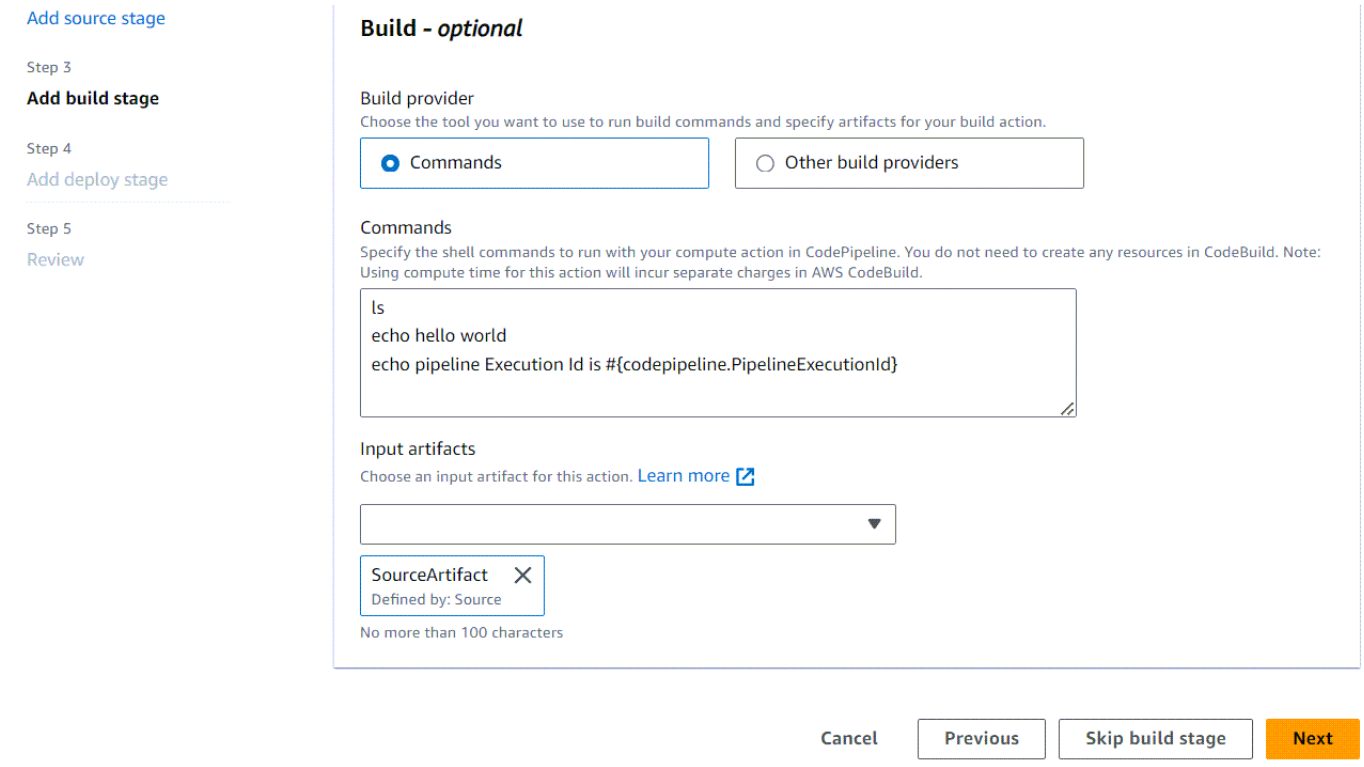
9. Pada Langkah 4: Tambahkan tahap build, pilih Command.

   > **Note**
   >
   > Menjalankan tindakan Commands akan dikenakan biaya terpisah di AWS CodeBuild.

Masukkan perintah-perintah berikut.

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

Pilih Berikutnya.



10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

11. Pada Langkah 6: Tambahkan tahap penerapan, pilih Lewati tahap penerapan, lalu terima pesan peringatan dengan memilih Lewati lagi.

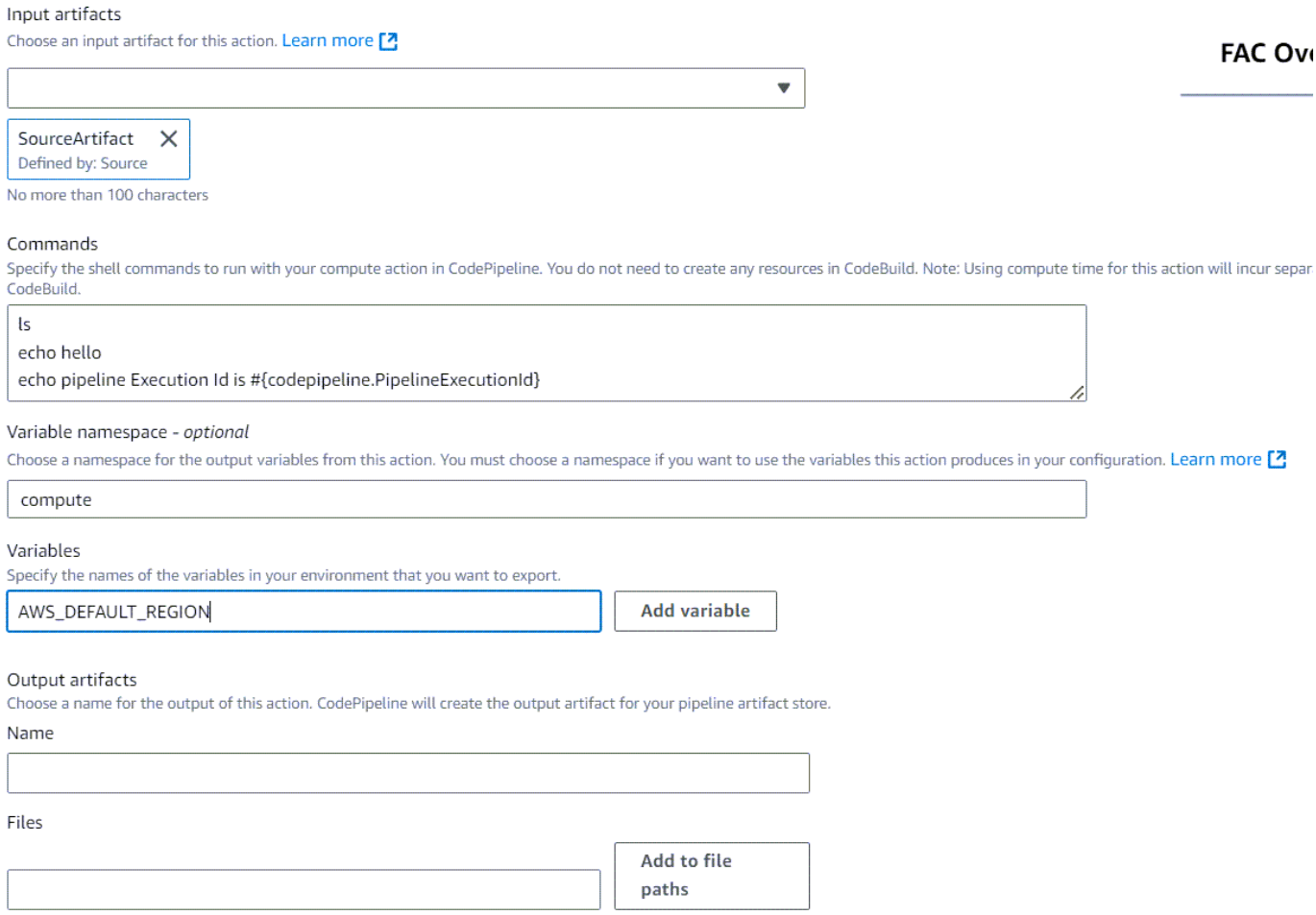
    Pilih Berikutnya.

12. Pada Langkah 7: Tinjau, tinjau informasinya, lalu pilih Buat pipeline.

13. Sebagai langkah terakhir untuk membuat tindakan Anda, tambahkan variabel lingkungan ke tindakan yang akan menghasilkan variabel keluaran untuk tindakan tersebut. Pada tindakan

Perintah, pilih Edit. Di layar Edit, tentukan namespace variabel untuk tindakan Anda dengan memasukkan `compute` bidang Namespace variabel.

Tambahkan variabel CodeBuild output`AWS_Default_Region`, lalu pilih Tambahkan variabel.



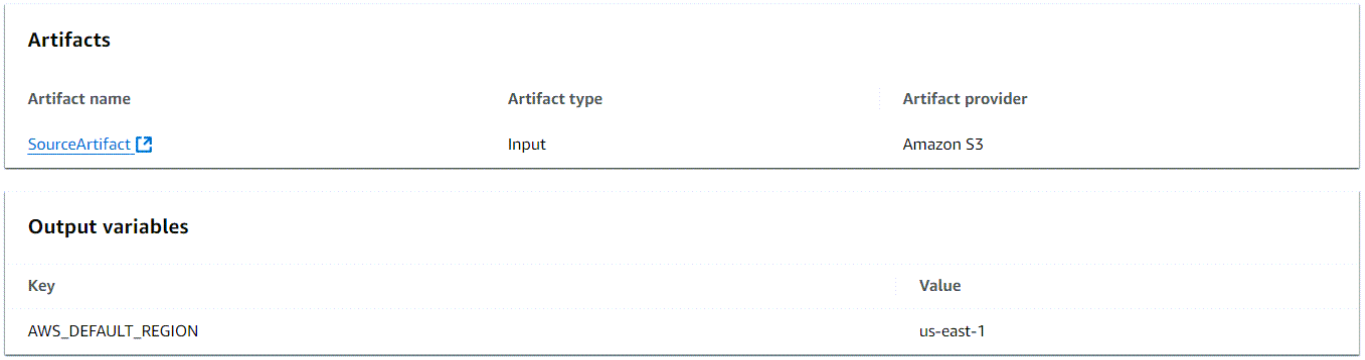
Langkah 3: Jalankan pipeline Anda dan verifikasi perintah build

Lepaskan perubahan untuk menjalankan pipeline Anda. Verifikasi bahwa perintah build dijalankan dengan melihat riwayat eksekusi, log build, dan variabel keluaran.

Untuk melihat log tindakan dan variabel keluaran

1. Setelah pipeline berjalan dengan sukses, Anda dapat melihat log dan output untuk tindakan tersebut.
2. Untuk melihat variabel output untuk tindakan, pilih Riwayat, lalu pilih Timeline.

Lihat variabel output yang ditambahkan ke tindakan. Output untuk tindakan Commands menunjukkan variabel output diselesaikan ke Wilayah tindakan.



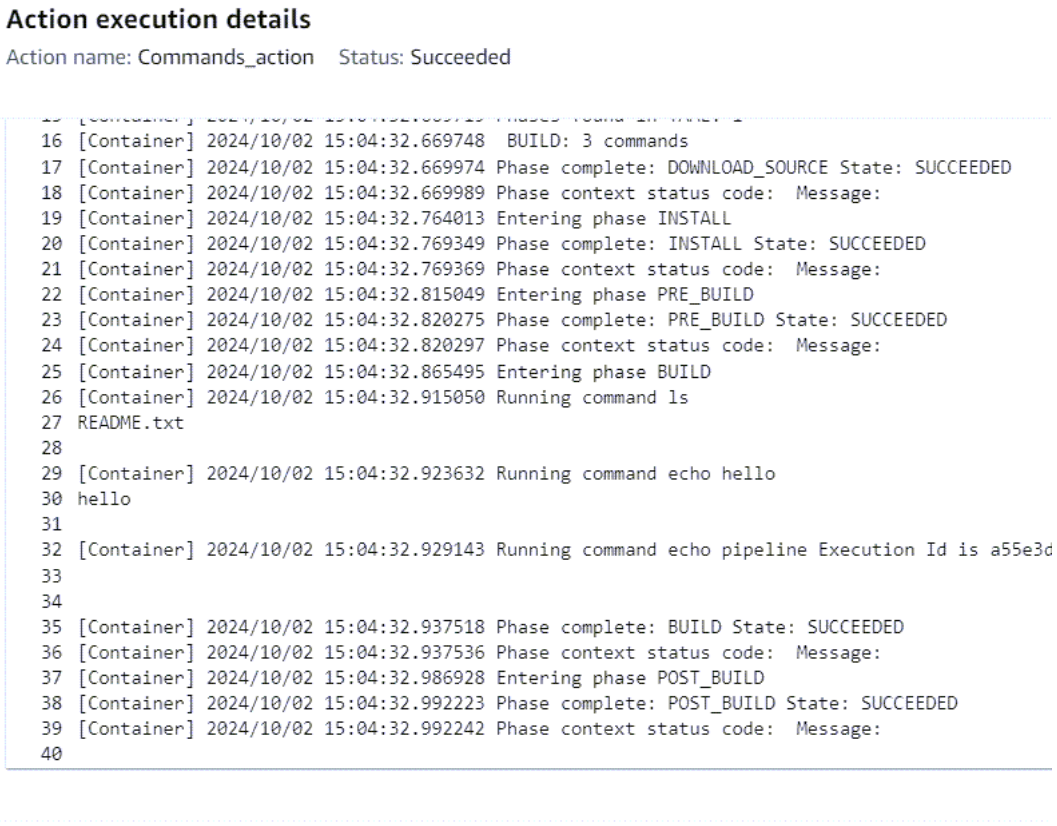
Artifacts

Artifact name	Artifact type	Artifact provider
SourceArtifact	Input	Amazon S3

Output variables

Key	Value
AWS_DEFAULT_REGION	us-east-1

3. Untuk melihat log tindakan, pilih Lihat detail pada tindakan Perintah yang berhasil. Lihat log untuk tindakan Perintah.



Action execution details

Action name: Commands_action Status: Succeeded

```
16 [Container] 2024/10/02 15:04:32.669748  BUILD: 3 commands
17 [Container] 2024/10/02 15:04:32.669974 Phase complete: DOWNLOAD_SOURCE State: SUCCEEDED
18 [Container] 2024/10/02 15:04:32.669989 Phase context status code:  Message:
19 [Container] 2024/10/02 15:04:32.764013 Entering phase INSTALL
20 [Container] 2024/10/02 15:04:32.769349 Phase complete: INSTALL State: SUCCEEDED
21 [Container] 2024/10/02 15:04:32.769369 Phase context status code:  Message:
22 [Container] 2024/10/02 15:04:32.815049 Entering phase PRE_BUILD
23 [Container] 2024/10/02 15:04:32.820275 Phase complete: PRE_BUILD State: SUCCEEDED
24 [Container] 2024/10/02 15:04:32.820297 Phase context status code:  Message:
25 [Container] 2024/10/02 15:04:32.865495 Entering phase BUILD
26 [Container] 2024/10/02 15:04:32.915050 Running command ls
27 README.txt
28
29 [Container] 2024/10/02 15:04:32.923632 Running command echo hello
30 hello
31
32 [Container] 2024/10/02 15:04:32.929143 Running command echo pipeline Execution Id is a55e3d
33
34
35 [Container] 2024/10/02 15:04:32.937518 Phase complete: BUILD State: SUCCEEDED
36 [Container] 2024/10/02 15:04:32.937536 Phase context status code:  Message:
37 [Container] 2024/10/02 15:04:32.986928 Entering phase POST_BUILD
38 [Container] 2024/10/02 15:04:32.992223 Phase complete: POST_BUILD State: SUCCEEDED
39 [Container] 2024/10/02 15:04:32.992242 Phase context status code:  Message:
40
```

Tutorial: Gunakan tag Git untuk memulai pipeline Anda

Dalam tutorial ini, Anda akan membuat pipeline yang terhubung ke GitHub repositori Anda di mana tindakan sumber dikonfigurasi untuk jenis pemicu tag Git. Ketika tag Git dibuat pada komit, pipeline Anda dimulai. Contoh ini menunjukkan cara membuat pipeline yang memungkinkan pemfilteran tag berdasarkan sintaks nama tag. Untuk informasi selengkapnya tentang pemfilteran dengan pola glob, lihat. Bekerja dengan pola glob dalam sintaks

Important

Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Tutorial ini terhubung ke GitHub melalui tipe `CodeStarSourceConnection` tindakan.

Note

Fitur ini tidak tersedia di Wilayah Asia Pasifik (Hong Kong), Afrika (Cape Town), Timur Tengah (Bahrain), atau Eropa (Zurich). Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline. Untuk pertimbangan dengan tindakan ini di Wilayah Eropa (Milan), lihat catatan diCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Topik

- Prasyarat
- Langkah 1: Buka CloudShell dan kloning repositori Anda
- Langkah 2: Buat pipeline untuk memicu tag Git
- Langkah 3: Tandai komit Anda untuk rilis
- Langkah 4: Lepaskan perubahan dan lihat log

Prasyarat

Sebelum Anda mulai, Anda harus melakukan hal berikut:

- Buat GitHub repositori dengan akun Anda GitHub .
- Siapkan GitHub kredensialmu. Ketika Anda menggunakan Konsol Manajemen AWS untuk mengatur koneksi, Anda diminta untuk masuk dengan GitHub kredensi Anda.

Langkah 1: Buka CloudShell dan kloning repositori Anda

Anda dapat menggunakan antarmuka baris perintah untuk mengkloning repositori Anda, membuat komit, dan menambahkan tag. Tutorial ini meluncurkan CloudShell contoh untuk antarmuka baris perintah.

1. Masuk ke Konsol Manajemen AWS.
2. Di bilah navigasi atas, pilih AWS ikon. Halaman utama Konsol Manajemen AWS tampilan.
3. Di bilah navigasi atas, pilih AWS CloudShell ikon. CloudShell terbuka. Tunggu sementara CloudShell lingkungan dibuat.

   > **Note**
   >
   > Jika Anda tidak melihat CloudShell ikon, pastikan Anda berada di Wilayah yang didukung oleh CloudShell. Tutorial ini mengasumsikan Anda berada di Wilayah AS Barat (Oregon).

4. Masuk GitHub, navigasikan ke repositori Anda. Pilih Kode, lalu pilih HTTPS. Salin jalurnya. Alamat untuk mengkloning repositori Git Anda disalin ke clipboard Anda.
5. Jalankan perintah berikut untuk mengkloning repositori.

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

6. Masukkan GitHub akun Anda `Username` dan `Password` saat diminta. Untuk `Password` entri, Anda harus menggunakan token yang dibuat pengguna daripada kata sandi akun Anda.

Langkah 2: Buat pipeline untuk memicu tag Git

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan koneksi ke GitHub repositori dan tindakan Anda.
- Tahap build dengan aksi AWS CodeBuild build.

Untuk membuat alur dengan wizard

1. Masuk ke CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Alur, pilih Buat alur.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyGitHubTagsPipeline`**.
5. Dalam tipe Pipeline, pertahankan pilihan default di V2. Jenis pipa berbeda dalam karakteristik dan harga. Untuk informasi selengkapnya, lihat Jenis pipa.
6. Di Peran layanan, pilih Peran layanan baru.

   > **Note**
   >
   > Jika Anda memilih untuk menggunakan peran CodePipeline layanan yang ada, pastikan Anda telah menambahkan izin `codestar-connections:UseConnection` IAM ke kebijakan peran layanan Anda. Untuk petunjuk tentang peran CodePipeline layanan, lihat Menambahkan izin ke peran CodePipeline layanan.

7. Di bawah Pengaturan lanjutan, biarkan default. Di Penyimpanan artifact, pilih Lokasi default untuk menggunakan penyimpanan artifact default, seperti bucket artifact Amazon S3 yang ditetapkan sebagai default, untuk alur Anda di Wilayah yang Anda pilih untuk alur Anda.

   > **Note**
   >
   > Ini bukan bucket sumber untuk kode sumber Anda. Ini adalah penyimpanan artifact untuk alur Anda. Penyimpanan artifact terpisah, seperti bucket S3, diperlukan untuk setiap alur.

   Pilih Berikutnya.
8. Pada Langkah 3: Tambahkan halaman tahap sumber, tambahkan tahap sumber:

   a. Di penyedia Sumber, pilih GitHub (melalui GitHub Aplikasi).

b. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.

c. Dalam nama Repositori, pilih nama repositori Anda GitHub.

d. Di cabang Default, pilih cabang yang ingin Anda tentukan saat pipeline dimulai secara manual atau dengan peristiwa sumber yang bukan tag Git. Jika sumber perubahan bukan pemicu atau jika eksekusi pipeline dimulai secara manual, maka perubahan yang digunakan akan menjadi komit HEAD dari cabang default.

e. Di bawah acara Webhook, di Jenis filter, pilih Tag.

Di bidang Tag atau pola, masukkan`release*`.

> **Important**
>
> Pipeline yang dimulai dengan jenis pemicu tag Git akan dikonfigurasi untuk peristiwa WebHookv2 dan tidak akan menggunakan peristiwa Webhook (deteksi perubahan pada semua peristiwa push) untuk memulai pipeline.

Pilih Berikutnya.

9. Di Tambahkan tahap membangun, tambahkan sebuah tahap membangun:

a. Di Penyedia pembangunan, pilih AWS CodeBuild. Izinkan Wilayah ke default ke Wilayah alur.

b. Pilih Buat proyek.

c. Di Nama proyek, masukkan nama untuk proyek pembangunan ini.

d. Di Citra lingkungan, pilih Citra terkelola. Untuk Sistem operasi, pilih Ubuntu.

e. Untuk Waktu aktif, pilih Standar. Untuk Gambar, pilih:5.0aws/codebuild/standard.

f. Untuk Peran layanan, pilih Peran layanan baru.

> **Note**
>
> Perhatikan nama peran CodeBuild layanan Anda. Anda akan membutuhkan nama peran untuk langkah terakhir dalam tutorial ini.

g. Pada Buildspec, untuk Spesifikasi membangun, pilih Sisipkan perintah membangun. Pilih Beralih ke editor, dan tempel yang berikut ini di bawah perintah Build.

```
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. Pilih Lanjutkan ke CodePipeline. Ini kembali ke CodePipeline konsol dan membuat CodeBuild proyek yang menggunakan perintah build Anda untuk konfigurasi. Proyek build menggunakan peran layanan untuk mengelola Layanan AWS izin. Langkah ini mungkin memakan waktu beberapa menit.

i. Pilih Berikutnya.

10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

11. Pada Langkah 6: Tambahkan halaman tahap penerapan, pilih Lewati tahap penerapan, lalu terima pesan peringatan dengan memilih Lewati lagi. Pilih Berikutnya.

12. Pada Langkah 7: Tinjau, pilih Buat pipeline.

Langkah 3: Tandai komit Anda untuk rilis

Setelah Anda membuat pipeline dan menentukan tag Git, Anda dapat menandai commit di GitHub repositori Anda. Dalam langkah-langkah ini, Anda akan menandai komit dengan `release-1` tag. Setiap komit dalam repositori Git harus memiliki tag Git yang unik. Saat Anda memilih komit dan menandainya, ini memungkinkan Anda untuk memasukkan perubahan dari cabang yang berbeda ke dalam penerapan pipeline Anda. Perhatikan bahwa rilis nama tag tidak berlaku untuk konsep rilis di GitHub.

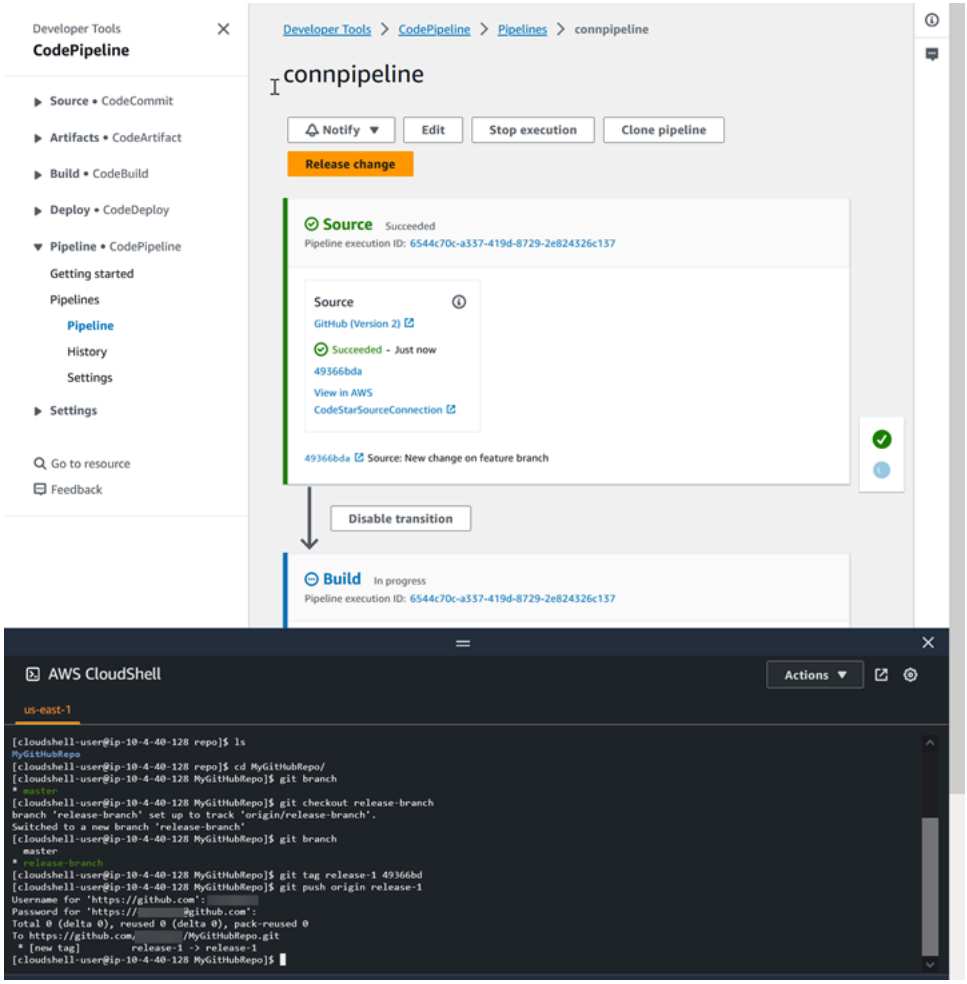
1. Referensikan komit yang disalin yang ingin IDs Anda tag. Untuk melihat komit di setiap cabang, di CloudShell terminal, masukkan perintah berikut untuk menangkap komit yang ingin IDs Anda tag:

   ```
   git log
   ```

2. Di CloudShell terminal, masukkan perintah untuk menandai komit Anda dan dorong ke asal. Setelah Anda menandai komit Anda, Anda menggunakan perintah git push untuk mendorong tag ke asal. Dalam contoh berikut, masukkan perintah berikut untuk menggunakan `release-1` tag untuk komit kedua dengan ID49366bd. Tag ini akan disaring oleh filter `release*` tag pipeline dan akan memulai pipeline.

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

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



Langkah 4: Lepaskan perubahan dan lihat log

1. Setelah pipeline berjalan dengan sukses, pada tahap build yang berhasil, pilih Lihat log.

   Di bawah Log, lihat keluaran CodeBuild build. Perintah menampilkan nilai variabel yang dimasukkan.

2. Di halaman Riwayat, lihat kolom Pemicu. Lihat tipe pemicu GitTag : release-1.

Tutorial: Filter nama cabang untuk permintaan tarik untuk memulai pipeline Anda (tipe V2)

Dalam tutorial ini, Anda akan membuat pipeline yang terhubung ke GitHub repositori.com Anda di mana tindakan sumber dikonfigurasi untuk memulai pipeline Anda dengan konfigurasi pemicu yang memfilter permintaan tarik. Ketika peristiwa permintaan tarik tertentu terjadi untuk cabang tertentu, pipeline Anda dimulai. Contoh ini menunjukkan cara membuat pipeline yang memungkinkan pemfilteran untuk nama cabang. Untuk informasi selengkapnya tentang bekerja dengan pemicu, lihatTambahkan filter untuk jenis acara permintaan push dan pull (CLI). Untuk informasi selengkapnya tentang pemfilteran dengan pola regex dalam format glob, lihat. Bekerja dengan pola glob dalam sintaks

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Tutorial ini terhubung GitHub ke.com melalui tipe `CodeStarSourceConnection` tindakan.

Topik

- Prasyarat
- Langkah 1: Buat pipeline untuk memulai permintaan tarik untuk cabang tertentu
- Langkah 2: Buat dan gabungkan permintaan tarik GitHub di.com untuk memulai eksekusi pipeline Anda

Prasyarat

Sebelum Anda mulai, Anda harus melakukan hal berikut:

- Buat GitHub repositori.com dengan akun GitHub .com Anda.
- Siapkan GitHub kredensialmu. Ketika Anda menggunakan Konsol Manajemen AWS untuk mengatur koneksi, Anda diminta untuk masuk dengan GitHub kredensi Anda.

Langkah 1: Buat pipeline untuk memulai permintaan tarik untuk cabang tertentu

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan koneksi ke repositori dan tindakan GitHub .com Anda.
- Tahap build dengan aksi AWS CodeBuild build.

Untuk membuat alur dengan wizard

1. Masuk ke CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Alur, pilih Buat alur.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyFilterBranchesPipeline`**.
5. Dalam tipe Pipeline, pertahankan pilihan default di V2. Jenis pipa berbeda dalam karakteristik dan harga. Untuk informasi selengkapnya, lihat Jenis pipa.
6. Di Peran layanan, pilih Peran layanan baru.

> **Note**
>
> Jika Anda memilih untuk menggunakan peran CodePipeline layanan yang ada, pastikan Anda telah menambahkan izin `codeconnections:UseConnection` IAM ke kebijakan peran layanan Anda. Untuk petunjuk tentang peran CodePipeline layanan, lihat Menambahkan izin ke peran CodePipeline layanan.

7. Di bawah Pengaturan lanjutan, biarkan default. Di Penyimpanan artifact, pilih Lokasi default untuk menggunakan penyimpanan artifact default, seperti bucket artifact Amazon S3 yang ditetapkan sebagai default, untuk alur Anda di Wilayah yang Anda pilih untuk alur Anda.

> **Note**
>
> Ini bukan bucket sumber untuk kode sumber Anda. Ini adalah penyimpanan artifact untuk alur Anda. Penyimpanan artifact terpisah, seperti bucket S3, diperlukan untuk setiap alur.

Pilih Berikutnya.

8. Pada Langkah 3: Tambahkan halaman tahap sumber, tambahkan tahap sumber:

   a. Di penyedia Sumber, pilih GitHub (melalui GitHub Aplikasi).

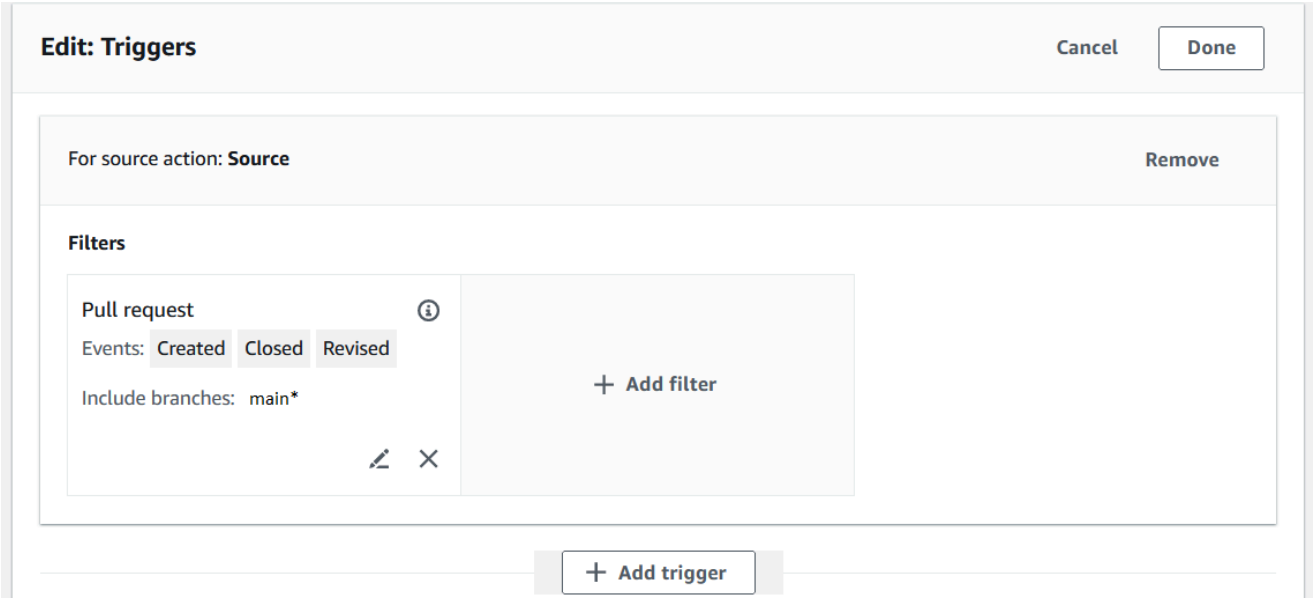
   b. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.

   c. Dalam nama Repositori, pilih nama GitHub repositori.com Anda.

   d. Di bawah Jenis pemicu, pilih Tentukan filter.

      Di bawah Jenis acara, pilih Permintaan tarik. Pilih semua peristiwa di bawah permintaan tarik sehingga peristiwa terjadi untuk permintaan tarik yang dibuat, diperbarui, atau ditutup.

      Di bawah Cabang, di bidang Sertakan, masukkan`main*`.



> **Important**
>
> Saluran pipa yang dimulai dengan jenis pemicu ini akan dikonfigurasi untuk peristiwa WebHookv2 dan tidak akan menggunakan peristiwa Webhook (deteksi perubahan pada semua peristiwa push) untuk memulai pipeline.

Pilih Berikutnya.

9. Pada Langkah 4: Tambahkan tahap build, di penyedia Build, pilih AWS CodeBuild. Izinkan Wilayah ke default ke Wilayah alur. Pilih atau buat proyek build seperti yang diinstruksikan. Tutorial: Gunakan tag Git untuk memulai pipeline Anda Tindakan ini hanya akan digunakan dalam tutorial ini sebagai tahap kedua yang diperlukan untuk membuat pipeline Anda.
10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

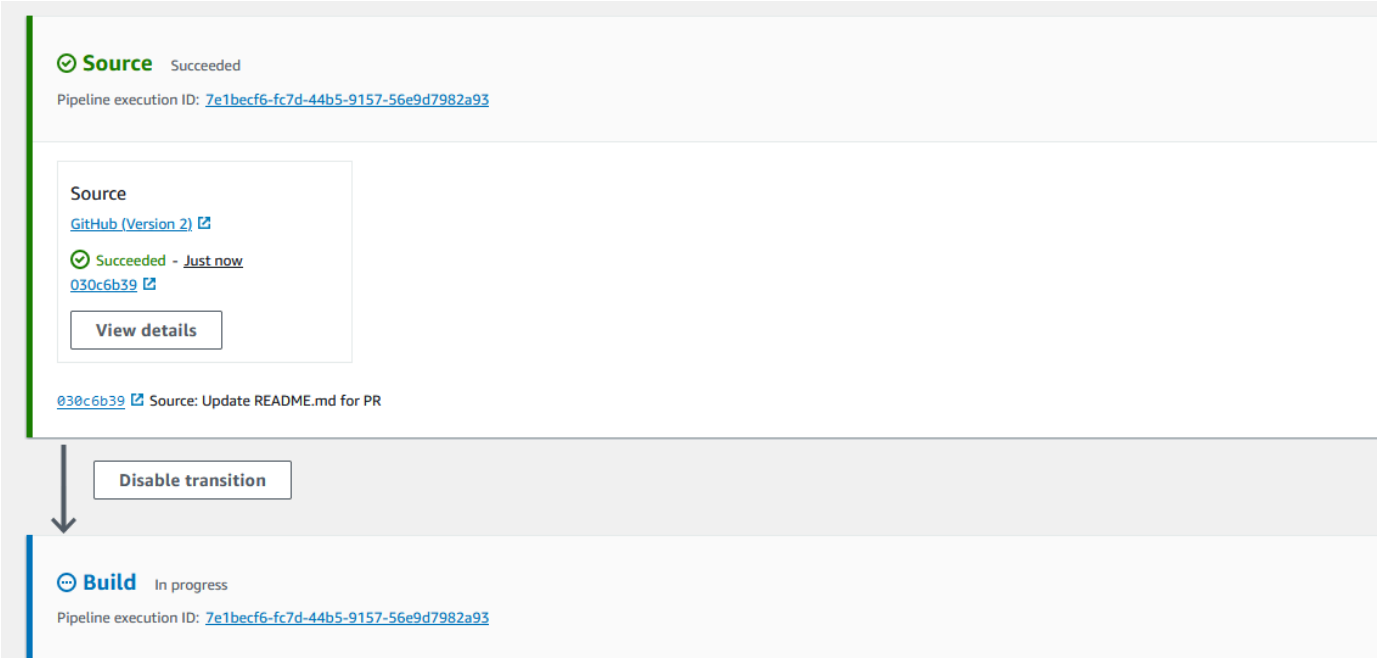
    Pilih Berikutnya.
11. Pada Langkah 6: Tambahkan halaman tahap penerapan, pilih Lewati tahap penerapan, lalu terima pesan peringatan dengan memilih Lewati lagi. Pilih Berikutnya.
12. Pada Langkah 7: Tinjau, pilih Buat pipeline.

Langkah 2: Buat dan gabungkan permintaan tarik GitHub di.com untuk memulai eksekusi pipeline Anda

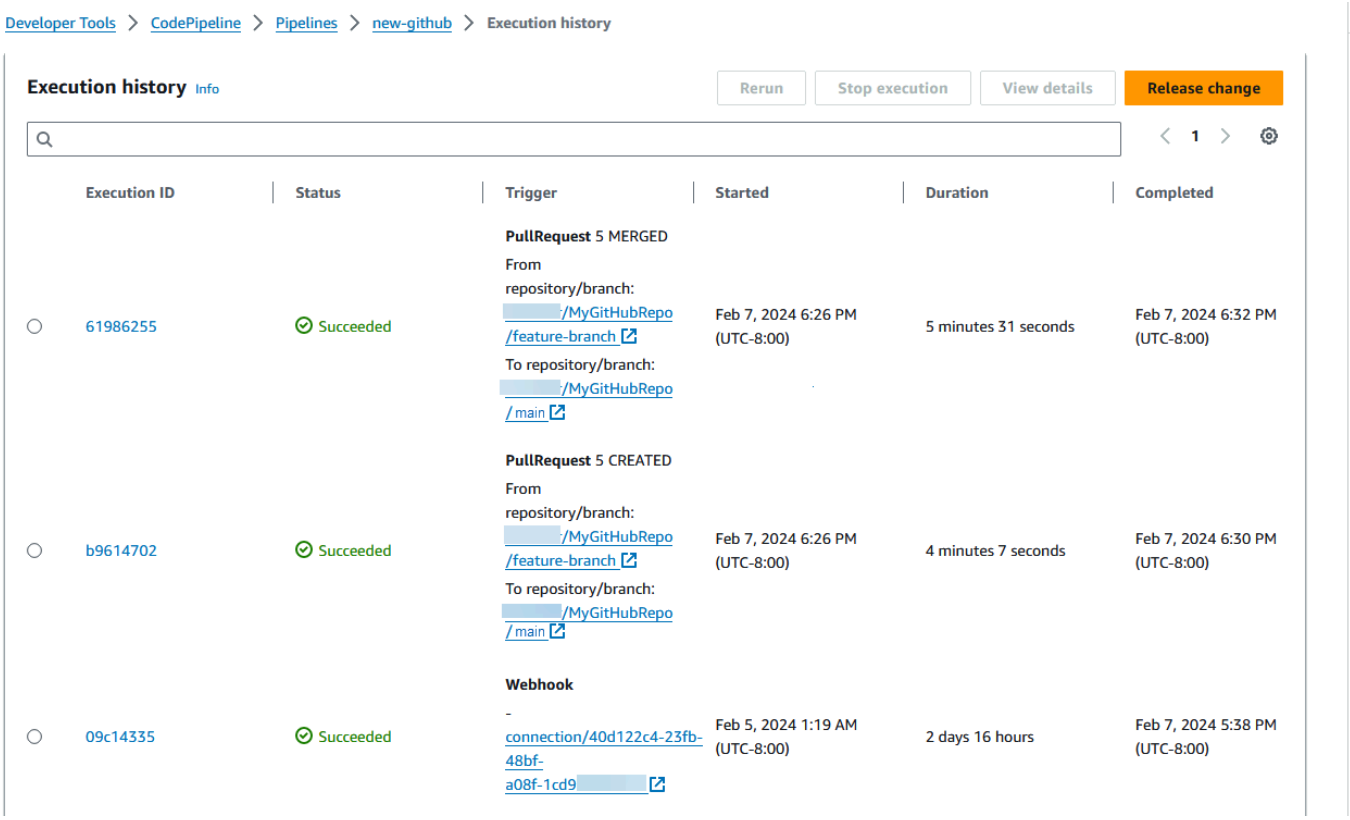
Di bagian ini, Anda membuat dan menggabungkan permintaan tarik. Ini memulai pipeline Anda, dengan satu eksekusi untuk permintaan tarik terbuka dan satu eksekusi untuk permintaan tarik tertutup.

Untuk membuat permintaan tarik dan memulai pipeline Anda

1. GitHubDi.com, buat permintaan tarik dengan membuat perubahan pada README.md pada cabang fitur dan menaikkan permintaan tarik ke cabang. `main` Lakukan perubahan dengan pesan seperti`Update README.md for PR`.
2. Pipeline dimulai dengan revisi sumber yang menampilkan pesan Sumber untuk permintaan tarik sebagai Perbarui README.md untuk PR.



3. Pilihriwayat. Dalam riwayat eksekusi pipeline, lihat peristiwa status permintaan tarik CREATED dan MERGED yang memulai eksekusi pipeline.



Tutorial: Gunakan variabel tingkat pipa

Dalam tutorial ini, Anda akan membuat pipeline di mana Anda menambahkan variabel pada tingkat pipeline dan menjalankan tindakan CodeBuild build yang mengeluarkan nilai variabel Anda.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Topik

- Prasyarat
- Langkah 1: Buat pipeline Anda dan bangun proyek
- Langkah 2: Lepaskan perubahan dan lihat log

Prasyarat

Sebelum Anda mulai, Anda harus melakukan hal berikut:

- Buat CodeCommit repositori.
- Tambahkan file.txt ke repositori.

Langkah 1: Buat pipeline Anda dan bangun proyek

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan koneksi ke CodeCommit repositori Anda.
- Tahap build dengan aksi AWS CodeBuild build.

Untuk membuat alur dengan wizard

1. Masuk ke CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Alur, pilih Buat alur.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyVariablesPipeline`**.
5. Dalam tipe Pipeline, pertahankan pilihan default di V2. Jenis pipa berbeda dalam karakteristik dan harga. Untuk informasi selengkapnya, lihat Jenis pipa.
6. Di Peran layanan, pilih Peran layanan baru.

   > **Note**
   >
   > Jika Anda memilih untuk menggunakan peran CodePipeline layanan yang ada, pastikan Anda telah menambahkan izin `codeconnections:UseConnection` IAM ke kebijakan peran layanan Anda. Untuk petunjuk tentang peran CodePipeline layanan, lihat Menambahkan izin ke peran CodePipeline layanan.

7. Di bawah Variabel, pilih Tambahkan variabel. Di Nama, masukkan `timeout`. Di Default, masukkan 1000. Dalam deskripsi, masukkan deskripsi berikut:**`Timeout`**.

   Ini akan membuat variabel di mana Anda dapat mendeklarasikan nilai ketika eksekusi pipeline dimulai. Nama variabel harus cocok `[A-Za-z0-9@\-_]+` dan bisa apa saja kecuali string kosong.
8. Di bawah Pengaturan lanjutan, biarkan default. Di Penyimpanan artifact, pilih Lokasi default untuk menggunakan penyimpanan artifact default, seperti bucket artifact Amazon S3 yang ditetapkan sebagai default, untuk alur Anda di Wilayah yang Anda pilih untuk alur Anda.

   > **Note**
   >
   > Ini bukan bucket sumber untuk kode sumber Anda. Ini adalah penyimpanan artifact untuk alur Anda. Penyimpanan artifact terpisah, seperti bucket S3, diperlukan untuk setiap alur.

   Pilih Berikutnya.
9. Pada Langkah 3: Tambahkan halaman tahap sumber, tambahkan tahap sumber:

   a. Di penyedia Sumber, pilih AWS CodeCommit.

b. Dalam nama Repositori dan nama Cabang, pilih repositori dan cabang Anda.

Pilih Berikutnya.

10. Pada Langkah 4: Tambahkan tahap build, tambahkan tahap build:

a. Di Penyedia pembangunan, pilih AWS CodeBuild. Izinkan Wilayah ke default ke Wilayah alur.

b. Pilih Buat proyek.

c. Di Nama proyek, masukkan nama untuk proyek pembangunan ini.

d. Di Citra lingkungan, pilih Citra terkelola. Untuk Sistem operasi, pilih Ubuntu.

e. Untuk Waktu aktif, pilih Standar. Untuk Gambar, pilih:5.0aws/codebuild/standard.

f. Untuk Peran layanan, pilih Peran layanan baru.

> **Note**
>
> Perhatikan nama peran CodeBuild layanan Anda. Anda akan membutuhkan nama peran untuk langkah terakhir dalam tutorial ini.

g. Pada Buildspec, untuk Spesifikasi membangun, pilih Sisipkan perintah membangun. Pilih Beralih ke editor, dan tempel yang berikut ini di bawah perintah Build. Dalam buildspec, variabel pelanggan `$CUSTOM_VAR1` akan digunakan untuk menampilkan variabel pipeline di log build. Anda akan membuat variabel `$CUSTOM_VAR1` output sebagai variabel lingkungan pada langkah berikut.

```
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. Pilih Lanjutkan ke CodePipeline. Ini kembali ke CodePipeline konsol dan membuat CodeBuild proyek yang menggunakan perintah build Anda untuk konfigurasi. Proyek build menggunakan peran layanan untuk mengelola Layanan AWS izin. Langkah ini mungkin memakan waktu beberapa menit.

i. Di bawah variabel Lingkungan - opsional, untuk membuat variabel lingkungan sebagai variabel input untuk tindakan build yang akan diselesaikan oleh variabel tingkat pipa, pilih Tambahkan variabel lingkungan. Ini akan membuat variabel yang ditentukan dalam buildspec sebagai. `$CUSTOM_VAR1` Di Nama, masukkan `CUSTOM_VAR1`. Dalam Value (Nilai), masukkan `#{variables.timeout}`. Di Type, pilih`Plaintext`.

   `#{variables.timeout}`Nilai untuk variabel lingkungan didasarkan pada namespace variabel tingkat pipa `variables` dan variabel tingkat pipa yang dibuat untuk pipeline pada langkah 7. `timeout`

j. Pilih Berikutnya.

11. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

12. Pada Langkah 6: Tambahkan halaman tahap penerapan, pilih Lewati tahap penerapan, lalu terima pesan peringatan dengan memilih Lewati lagi. Pilih Berikutnya.
13. Pada Langkah 7: Tinjau, pilih Buat pipeline.

Langkah 2: Lepaskan perubahan dan lihat log

1. Setelah pipeline berjalan dengan sukses, pada tahap build yang berhasil, pilih Lihat detail.

   Pada halaman detail, pilih tab Log. Lihat output CodeBuild build. Perintah menampilkan nilai variabel yang dimasukkan.
2. Di navigasi sebelah kiri, pilih History.

   Pilih eksekusi terbaru, lalu pilih tab Variabel. Lihat nilai yang diselesaikan untuk variabel pipeline.

Tutorial: Buat pipeline sederhana (ember S3)

Cara termudah untuk membuat pipeline adalah dengan menggunakan wizard Create pipeline di AWS CodePipeline konsol.

Dalam tutorial ini, Anda membuat pipeline dua tahap yang menggunakan bucket sumber S3 berversi dan CodeDeploy untuk merilis contoh aplikasi.

> **Note**
>
> Jika Amazon S3 adalah penyedia sumber untuk pipeline Anda, Anda dapat mem-zip file sumber atau file ke dalam satu .zip dan mengunggah.zip ke bucket sumber Anda. Anda juga dapat mengunggah satu file yang tidak di-zip; namun, tindakan hilir yang mengharapkan file.zip akan gagal.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Setelah Anda membuat pipeline sederhana ini, Anda menambahkan tahap lain dan kemudian menonaktifkan dan mengaktifkan transisi antar tahapan.

> **Important**
>
> Banyak tindakan yang Anda tambahkan ke pipeline dalam prosedur ini melibatkan AWS sumber daya yang perlu Anda buat sebelum membuat pipeline. AWS sumber daya untuk tindakan sumber Anda harus selalu dibuat di AWS Wilayah yang sama tempat Anda membuat pipeline. Misalnya, jika Anda membuat pipeline di Wilayah AS Timur (Ohio), CodeCommit repositori Anda harus berada di Wilayah AS Timur (Ohio).
> Anda dapat menambahkan tindakan lintas wilayah saat membuat pipeline. AWS sumber daya untuk tindakan lintas wilayah harus berada di AWS Wilayah yang sama di mana Anda berencana untuk menjalankan tindakan. Untuk informasi selengkapnya, lihat Menambahkan tindakan Lintas wilayah di CodePipeline.

Sebelum Anda mulai, Anda harus menyelesaikan prasyarat di. Memulai dengan CodePipeline

Topik

- Langkah 1: Buat bucket sumber S3 untuk aplikasi Anda
- Langkah 2: Buat instans Windows Amazon EC2 dan instal agen CodeDeploy
- Langkah 3: Buat aplikasi di CodeDeploy
- Langkah 4: Buat pipeline pertama Anda di CodePipeline
- (Opsional) Langkah 5: Tambahkan tahap lain ke pipeline Anda
- (Opsional) Langkah 6: Nonaktifkan dan aktifkan transisi antar tahapan CodePipeline
- Langkah 7: Bersihkan sumber daya

Langkah 1: Buat bucket sumber S3 untuk aplikasi Anda

Anda dapat menyimpan file sumber atau aplikasi Anda di lokasi berversi apa pun. Dalam tutorial ini, Anda membuat bucket S3 untuk file aplikasi sampel dan mengaktifkan pembuatan versi pada bucket itu. Setelah mengaktifkan pembuatan versi, Anda menyalin contoh aplikasi ke bucket tersebut.

Untuk membuat bucket S3

1. Masuk ke konsol di Konsol Manajemen AWS. Buka konsol S3.
2. Pilih Buat bucket.
3. Dalam nama Bucket, masukkan nama untuk bucket Anda (misalnya,**`awscodepipeline-demobucket-example-date`**).

   > **Note**
   >
   > Karena semua nama bucket di Amazon S3 harus unik, gunakan salah satu nama Anda sendiri, bukan nama yang ditunjukkan pada contoh. Anda dapat mengubah nama contoh hanya dengan menambahkan tanggal ke dalamnya. Catat nama ini karena Anda membutuhkannya untuk sisa tutorial ini.

   Di Wilayah, pilih Wilayah tempat Anda ingin membuat pipeline, seperti US West (Oregon), lalu pilih Buat bucket.
4. Setelah ember dibuat, spanduk sukses ditampilkan. Pilih Buka detail ember.
5. Pada tab Properties, pilih Versioning. Pilih Aktifkan pembuatan versi, lalu pilih Simpan.

   Saat pembuatan versi diaktifkan, Amazon S3 menyimpan setiap versi dari setiap objek di bucket.
6. Pada tab Izin, biarkan default. Untuk informasi selengkapnya tentang izin bucket dan objek S3, lihat Menentukan Izin dalam Kebijakan.
7. Selanjutnya, unduh sampel dan simpan ke folder atau direktori di komputer lokal Anda.

   a. Pilih salah satu dari berikut ini. Pilih `SampleApp_Windows.zip` apakah Anda ingin mengikuti langkah-langkah dalam tutorial ini untuk instance Windows Server.

      - Jika Anda ingin menyebarkan ke instans Amazon Linux menggunakan CodeDeploy, unduh contoh aplikasi di sini: SampleApp _Linux.zip.

- Jika Anda ingin menyebarkan ke instance Windows Server menggunakan CodeDeploy, unduh contoh aplikasi di sini: SampleApp _Windows.zip.

Aplikasi sampel berisi file-file berikut untuk digunakan dengan CodeDeploy:

- `appspec.yml`— File spesifikasi aplikasi (AppSpecfile) adalah file berformat YAMM yang digunakan oleh CodeDeploy untuk mengelola penyebaran. Untuk informasi selengkapnya tentang AppSpec file, lihat Referensi CodeDeploy AppSpec file di Panduan AWS CodeDeploy Pengguna.
- `index.html`— File indeks berisi halaman beranda untuk aplikasi sampel yang digunakan.
- `LICENSE.txt`— File lisensi berisi informasi lisensi untuk aplikasi sampel.
- File untuk skrip — Aplikasi sampel menggunakan skrip untuk menulis file teks ke lokasi pada instance Anda. Satu file ditulis untuk masing-masing dari beberapa peristiwa siklus hidup CodeDeploy penerapan sebagai berikut:
  - (Hanya sampel Linux) `scripts` folder - Folder berisi skrip shell berikut untuk menginstal dependensi dan memulai dan menghentikan aplikasi sampel untuk penerapan otomatis:`install_dependencies`,, dan. `start_server` `stop_server`
  - (Hanya sampel Windows) `before-install.bat` - Ini adalah skrip batch untuk peristiwa siklus hidup `BeforeInstall` penerapan, yang akan berjalan untuk menghapus file lama yang ditulis selama penerapan sampel ini sebelumnya dan membuat lokasi pada instance Anda untuk menulis file baru.

b. Unduh file terkompresi (zip). Jangan unzip file.

8. Di konsol Amazon S3, untuk bucket Anda, unggah file:

   a. Pilih Unggah.

   b. Seret dan lepas file atau pilih Tambahkan file dan telusuri file tersebut.

   c. Pilih Unggah.

Langkah 2: Buat instans Windows Amazon EC2 dan instal agen CodeDeploy

> **Note**
>
> Tutorial ini memberikan contoh langkah-langkah untuk membuat instans Windows Amazon EC2. Untuk contoh langkah untuk membuat instans Amazon EC2 Linux, lihat. Langkah 3: Buat instans Amazon EC2 Linux dan instal agen CodeDeploy Saat diminta jumlah instance yang akan dibuat, tentukan 2 instance.

Pada langkah ini, Anda membuat instance Windows Server Amazon EC2 yang akan Anda gunakan aplikasi sampel. Sebagai bagian dari proses ini, Anda membuat peran instans dengan kebijakan yang memungkinkan penginstalan dan pengelolaan CodeDeploy agen pada instans. CodeDeploy Agen adalah paket perangkat lunak yang memungkinkan instance untuk digunakan dalam CodeDeploy penerapan. Anda juga melampirkan kebijakan yang memungkinkan instance mengambil file yang digunakan CodeDeploy agen untuk menyebarkan aplikasi Anda dan mengizinkan instance dikelola oleh SSM.

Untuk membuat peran instance

1. Buka konsol IAM di https://console.aws.amazon.com/iam/).
2. Dari dasbor konsol, pilih Peran.
3. Pilih Buat peran.
4. Di bawah Pilih jenis entitas tepercaya, pilih Layanan AWS. Di bawah Pilih kasus penggunaan, pilih EC2, lalu pilih Berikutnya: Izin.
5. Cari dan pilih kebijakan bernama **`AmazonEC2RoleforAWSCodeDeploy`**.
6. Cari dan pilih kebijakan bernama **`AmazonSSMManagedInstanceCore`**. Pilih Berikutnya: Tanda.
7. Pilih Berikutnya: Tinjau. Masukkan nama untuk peran (misalnya,**`EC2InstanceRole`**).

> **Note**
>
> Catat nama peran Anda untuk langkah selanjutnya. Anda memilih peran ini saat membuat instance Anda.

Pilih Buat peran.

Untuk meluncurkan instance

1. Buka konsol Amazon EC2 di. https://console.aws.amazon.com/ec2/
2. Dari navigasi samping, pilih Instans, dan pilih Luncurkan instance dari bagian atas halaman.
3. Di bawah Nama dan tag, di Nama, masukkan**`MyCodePipelineDemo`**. Ini menetapkan instance kunci tag **`Name`** dan nilai tag dari. **`MyCodePipelineDemo`** Kemudian, Anda membuat CodeDeploy aplikasi yang menyebarkan aplikasi sampel ke instance. CodeDeploymemilih instance untuk menyebarkan berdasarkan tag.
4. Di bawah Application and OS Images (Amazon Machine Image), pilih opsi Windows. (AMI ini digambarkan sebagai Pangkalan Microsoft Windows Server 2019 dan diberi label “Tingkat gratis memenuhi syarat” dan dapat ditemukan di bawah Mulai Cepat..)
5. Di bawah Jenis instans, pilih `t2.micro` tipe yang memenuhi syarat tingkat gratis sebagai konfigurasi perangkat keras untuk instans Anda.
6. Di bawah Key pair (login), pilih key pair atau buat satu.

   Anda juga dapat memilih Proceed without a key pair.

   > **Note**
   >
   > Untuk keperluan tutorial ini, Anda dapat melanjutkan tanpa key pair. Untuk menggunakan SSH untuk terhubung ke instance Anda, buat atau gunakan key pair.

7. Di bawah Pengaturan jaringan, lakukan hal berikut.

   Di Auto-assign IP Publik, pastikan statusnya Aktifkan.

   - Di samping Menetapkan grup keamanan, pilih Buat grup keamanan baru.
   - Di baris untuk SSH, di bawah Jenis sumber, pilih IP Saya.
   - Pilih Tambahkan grup keamanan, pilih HTTP, lalu di bawah Jenis sumber, pilih IP Saya.
8. Perluas Detail lanjutan. Di profil instans IAM, pilih peran IAM yang Anda buat di prosedur sebelumnya (misalnya,**`EC2InstanceRole`**).
9. Di bawah Ringkasan, di bawah Jumlah instance, masukkan.. 2
10. Pilih Luncurkan instans.

11. Pilih Lihat semua instans untuk menutup halaman konfirmasi dan kembali ke konsol.
12. Anda dapat melihat status peluncuran di halaman Instans. Saat Anda meluncurkan instans, status awalnya adalah `pending`. Setelah instans dimulai, statusnya berubah menjadi `running`, dan ia menerima nama DNS publik. (Jika kolom DNS Publik tidak ditampilkan, pilih ikon Tampilkan/Sembunyikan, lalu pilih DNS Publik.)
13. Diperlukan waktu beberapa menit sampai instans siap untuk terhubung dengan DNS tersebut. Periksa apakah pesan Anda telah lulus pemeriksaan statusnya. Anda dapat melihat informasi ini di kolom Pemeriksaan Status.

Langkah 3: Buat aplikasi di CodeDeploy

Dalam CodeDeploy, aplikasi adalah pengidentifikasi, dalam bentuk nama, untuk kode yang ingin Anda terapkan. CodeDeploy menggunakan nama ini untuk memastikan kombinasi yang benar dari revisi, konfigurasi penerapan, dan grup penyebaran direferensikan selama penerapan. Anda memilih nama CodeDeploy aplikasi yang Anda buat dalam langkah ini ketika Anda membuat pipeline Anda nanti dalam tutorial ini.

Anda pertama kali membuat peran layanan CodeDeploy untuk digunakan. Jika Anda telah membuat peran layanan, Anda tidak perlu membuat yang lain.

Untuk membuat peran CodeDeploy layanan

1. Buka konsol IAM di https://console.aws.amazon.com/iam/).
2. Dari dasbor konsol, pilih Peran.
3. Pilih Buat peran.
4. Di bawah Pilih entitas tepercaya, pilih Layanan AWS. Di bawah Kasus penggunaan, pilih CodeDeploy. Pilih CodeDeploydari opsi yang tercantum. Pilih Berikutnya. Kebijakan yang `AWSCodeDeployRole` dikelola sudah melekat pada peran tersebut.
5. Pilih Berikutnya.
6. Masukkan nama untuk peran (misalnya,**`CodeDeployRole`**), lalu pilih Buat peran.

Untuk membuat aplikasi di CodeDeploy

1. Buka CodeDeploy konsol di https://console.aws.amazon.com/codedeploy.
2. Jika halaman Aplikasi tidak muncul, pada AWS CodeDeploy menu, pilih Aplikasi.
3. Pilih Create application (Buat aplikasi).

4. Dalam nama Aplikasi, masukkan`MyDemoApplication`.
5. Di Compute Platform, pilih EC2/On-premise.
6. Pilih Create application (Buat aplikasi).

Untuk membuat grup penyebaran di CodeDeploy

1. Pada halaman yang menampilkan aplikasi Anda, pilih Buat grup penyebaran.
2. Dalam nama grup Deployment, masukkan**`MyDemoDeploymentGroup`**.
3. Di peran Layanan, pilih peran layanan yang Anda buat sebelumnya. Anda harus menggunakan peran layanan yang mempercayai AWS CodeDeploy , setidaknya, kepercayaan dan izin yang dijelaskan dalam Membuat Peran Layanan untuk. CodeDeploy Untuk mendapatkan ARN peran layanan, lihat Mendapatkan Peran Layanan ARN (Konsol).
4. Di bawah Jenis Deployment, pilih In-place.
5. Di bawah Konfigurasi lingkungan, pilih Instans Amazon EC2. Pilih Nama di bidang Kunci, dan di bidang Nilai, masukkan**`MyCodePipelineDemo`**.

   > **Important**
   >
   > Anda harus memilih nilai yang sama untuk kunci Nama di sini yang Anda tetapkan ke instans EC2 saat Anda membuatnya. Jika Anda menandai instance Anda dengan sesuatu selain**`MyCodePipelineDemo`**, pastikan untuk menggunakannya di sini.

6. Di bawah Konfigurasi Agen dengan AWS Systems Manager, pilih Sekarang dan jadwalkan pembaruan. Ini menginstal agen pada instance. Instans Windows sudah dikonfigurasi dengan agen SSM dan sekarang akan diperbarui dengan CodeDeploy agen.
7. Di bawah Pengaturan Deployment, pilih`CodeDeployDefault.OneAtaTime`.
8. Di bawah Load Balancer, pastikan kotak Aktifkan load balancing tidak dipilih. Anda tidak perlu mengatur penyeimbang beban atau memilih grup target untuk contoh ini. Setelah Anda membatalkan pilihan kotak centang, opsi penyeimbang beban tidak ditampilkan.
9. Di bagian Advanced, tinggalkan default.
10. Pilih Buat grup penyebaran.

Langkah 4: Buat pipeline pertama Anda di CodePipeline

Di bagian tutorial ini, Anda membuat pipeline. Sampel berjalan secara otomatis melalui pipa.

Untuk membuat proses rilis CodePipeline otomatis

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyFirstPipeline`**.

   > **Note**
   >
   > Jika Anda memilih nama lain untuk pipeline Anda, pastikan untuk menggunakan nama itu alih-alih **`MyFirstPipeline`** untuk sisa tutorial ini. Setelah Anda membuat pipeline, Anda tidak dapat mengubah namanya. Nama pipa tunduk pada beberapa batasan. Untuk informasi selengkapnya, lihat Kuota di AWS CodePipeline.

5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
6. Dalam peran Layanan, lakukan salah satu hal berikut:

   - Pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan baru di IAM.
   - Pilih Peran layanan yang ada untuk menggunakan peran layanan yang sudah dibuat di IAM. Di Nama peran, pilih peran layanan Anda dari daftar.

7. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.
8. Pada Langkah 3: Tambahkan tahap sumber, di penyedia Sumber, pilih Amazon S3. Di Bucket, masukkan nama bucket S3 yang Anda buat. Langkah 1: Buat bucket sumber S3 untuk aplikasi Anda Dalam kunci objek S3, masukkan kunci objek dengan atau tanpa jalur file, dan ingat untuk menyertakan ekstensi file. Misalnya, untuk`SampleApp_Windows.zip`, masukkan nama file contoh seperti yang ditunjukkan dalam contoh ini:

   ```
   SampleApp_Windows.zip
   ```

   Pilih Langkah selanjutnya.

Di bawah Ubah opsi deteksi, biarkan default. Hal ini CodePipeline memungkinkan Anda menggunakan Amazon CloudWatch Events untuk mendeteksi perubahan di bucket sumber Anda.

Pilih Berikutnya.

9. Pada Langkah 4: Tambahkan tahap build, pilih Lewati tahap build, lalu terima pesan peringatan dengan memilih Lewati lagi. Pilih Berikutnya.
10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.
11. Pada Langkah 6: Tambahkan tahap penerapan, di Penyedia penyebaran, pilih. CodeDeploy Bidang Region default sama Wilayah AWS dengan pipeline Anda. Di Nama aplikasi, masukkan`MyDemoApplication`, atau pilih tombol Refresh, lalu pilih nama aplikasi dari daftar. Di grup Deployment**`MyDemoDeploymentGroup`**, masukkan, atau pilih dari daftar, lalu pilih Berikutnya.

    > **Note**
    >
    > Nama Deploy adalah nama yang diberikan secara default ke tahap yang dibuat di Langkah 4: Tambahkan langkah tahap penerapan, sama seperti Source adalah nama yang diberikan ke tahap pertama pipeline.
12. Pada Langkah 7: Tinjau, tinjau informasinya, lalu pilih Buat pipeline.
13. Pipa mulai berjalan. Anda dapat melihat pesan kemajuan dan keberhasilan dan kegagalan saat CodePipeline sampel menyebarkan halaman web ke setiap instans Amazon EC2 dalam penerapan. CodeDeploy

Selamat! Anda baru saja membuat pipa sederhana di CodePipeline. Pipa memiliki dua tahap:

- Tahap sumber bernama Source, yang mendeteksi perubahan dalam aplikasi sampel berversi yang disimpan dalam bucket S3 dan menarik perubahan tersebut ke dalam pipeline.
- Tahap Deploy yang menyebarkan perubahan tersebut ke instans EC2 dengan. CodeDeploy

Sekarang, verifikasi hasilnya.

Untuk memverifikasi pipeline Anda berjalan dengan sukses

1. Lihat kemajuan awal pipa. Status setiap tahap berubah dari Tidak ada eksekusi yang belum ada dalam proses, dan kemudian menjadi Berhasil atau Gagal. Pipa harus menyelesaikan proses pertama dalam beberapa menit.
2. Setelah Berhasil ditampilkan untuk status tindakan, di area status untuk tahap Deploy, pilih Detail. Ini membuka CodeDeploy konsol.
3. Di tab grup Deployment, di bawah peristiwa siklus hidup Deployment, pilih ID instance. Ini membuka konsol EC2.
4. Pada tab Deskripsi, di DNS Publik, salin alamatnya, lalu tempelkan ke bilah alamat browser web Anda. Lihat halaman indeks untuk contoh aplikasi yang Anda unggah ke bucket S3 Anda.

   Halaman web ditampilkan untuk contoh aplikasi yang Anda unggah ke bucket S3 Anda.

Untuk informasi selengkapnya tentang tahapan, tindakan, dan cara kerja saluran pipa, lihatCodePipeline konsep .

(Opsional) Langkah 5: Tambahkan tahap lain ke pipeline Anda

Sekarang tambahkan tahap lain dalam pipeline untuk menyebarkan dari server pementasan ke server produksi menggunakan. CodeDeploy Pertama, Anda membuat grup penerapan lain di CodePipelineDemoApplication dalam CodeDeploy. Kemudian Anda menambahkan tahapan yang menyertakan tindakan yang menggunakan grup penyebaran ini. Untuk menambahkan tahap lain, Anda menggunakan CodePipeline konsol atau AWS CLI untuk mengambil dan mengedit struktur pipeline secara manual dalam file JSON, lalu jalankan update-pipeline perintah untuk memperbarui pipeline dengan perubahan Anda.

Topik

- Buat grup penerapan kedua di CodeDeploy
- Tambahkan grup penerapan sebagai tahap lain dalam pipeline Anda

Buat grup penerapan kedua di CodeDeploy

> **Note**
>
> Di bagian tutorial ini, Anda membuat grup penyebaran kedua, tetapi menyebarkan ke instans Amazon EC2 yang sama seperti sebelumnya. Ini hanya untuk tujuan demonstrasi. Ini sengaja

dirancang untuk gagal menunjukkan kepada Anda bagaimana kesalahan ditampilkan. CodePipeline

Untuk membuat grup penyebaran kedua di CodeDeploy

1. Buka CodeDeploy konsol di https://console.aws.amazon.com/codedeploy.
2. Pilih Aplikasi, dan dalam daftar aplikasi, pilih`MyDemoApplication`.
3. Pilih tab grup Deployment, lalu pilih Create deployment group.
4. Pada halaman Buat grup penyebaran, dalam nama grup Deployment, masukkan nama untuk grup penyebaran kedua (misalnya,). **`CodePipelineProductionFleet`**
5. Di Peran Layanan, pilih peran CodeDeploy layanan yang sama yang Anda gunakan untuk penerapan awal (bukan peran CodePipeline layanan).
6. Di bawah Jenis Deployment, pilih In-place.
7. Di bawah Konfigurasi lingkungan, pilih Instans Amazon EC2. Pilih Nama di kotak Kunci, dan di kotak Nilai, pilih `MyCodePipelineDemo` dari daftar. Tinggalkan konfigurasi default untuk pengaturan Deployment.
8. Di bawah konfigurasi Deployment, pilih`CodeDeployDefault.OneAtaTime`.
9. Di bawah Load Balancer, hapus Aktifkan penyeimbangan beban.
10. Pilih Buat grup penyebaran.

Tambahkan grup penerapan sebagai tahap lain dalam pipeline Anda

Sekarang setelah Anda memiliki grup penyebaran lain, Anda dapat menambahkan tahapan yang menggunakan grup penerapan ini untuk menyebarkan ke instans EC2 yang sama dengan yang Anda gunakan sebelumnya. Anda dapat menggunakan CodePipeline konsol atau AWS CLI untuk menambahkan tahap ini.

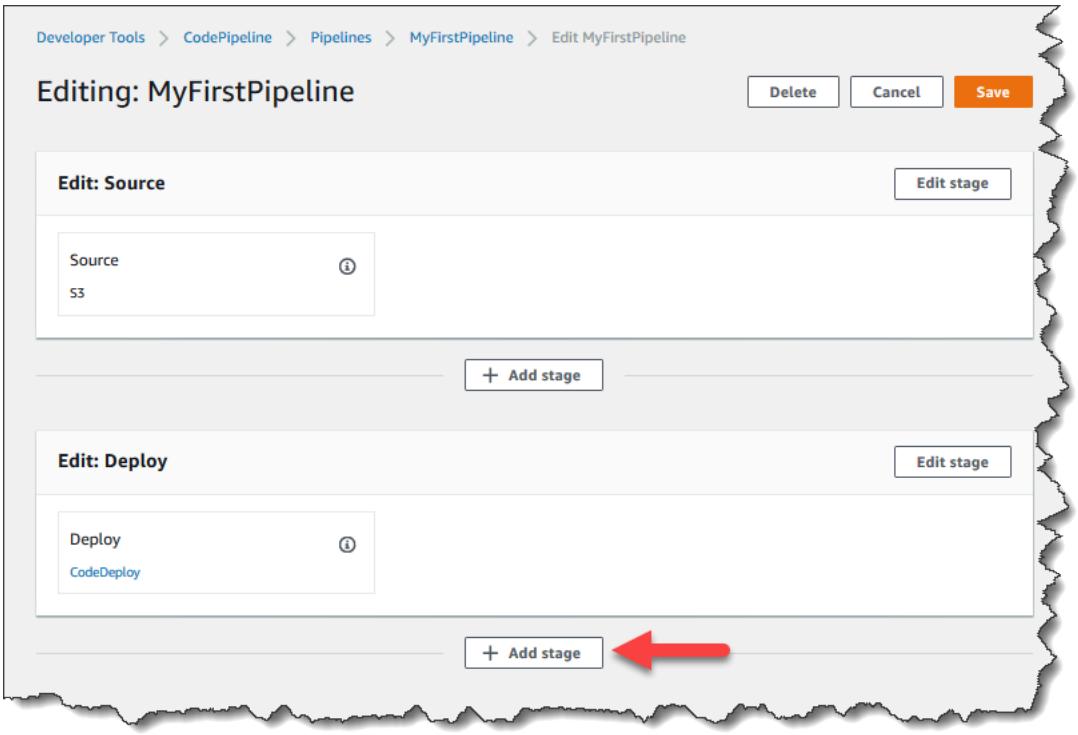
Topik

- Buat tahap ketiga (konsol)
- Buat tahap ketiga (CLI)

Buat tahap ketiga (konsol)

Anda dapat menggunakan CodePipeline konsol untuk menambahkan tahap baru yang menggunakan grup penyebaran baru. Karena grup penerapan ini menerapkan ke instans EC2 yang telah Anda gunakan, tindakan penerapan pada tahap ini gagal.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Di Nama, pilih nama pipeline yang Anda buat, MyFirstPipeline.
3. Pada halaman detail pipeline, pilih Edit.
4. Pada halaman Edit, pilih + Tambah tahap untuk menambahkan tahap segera setelah tahap Deploy.



5. Di Tambahkan tahap, dalam nama Panggung, masukkan**`Production`**. Pilih Tambahkan tahap.
6. Di tahap baru, pilih + Tambahkan grup tindakan.
7. Dalam Edit tindakan, dalam nama Tindakan, masukkan**`Deploy-Second-Deployment`**. Di Penyedia tindakan, di bawah Deploy, pilih CodeDeploy.
8. Di CodeDeploy bagian, di Nama aplikasi, pilih `MyDemoApplication` dari daftar drop-down, seperti yang Anda lakukan saat membuat pipeline. Di grup Deployment, pilih grup penyebaran yang baru saja Anda buat. **`CodePipelineProductionFleet`** Di artefak Input, pilih artefak input dari aksi sumber. Pilih Simpan.

9. Pada halaman Edit, pilih Simpan. Di Simpan perubahan pipeline, pilih Simpan.
10. Meskipun tahap baru telah ditambahkan ke pipeline Anda, status Tidak ada eksekusi belum ditampilkan karena tidak ada perubahan yang memicu proses pipeline lainnya. Anda harus menjalankan ulang revisi terakhir secara manual untuk melihat bagaimana pipeline yang diedit berjalan. Pada halaman detail pipeline, pilih Rilis ubah, lalu pilih Rilis saat diminta. Ini menjalankan revisi terbaru yang tersedia di setiap lokasi sumber yang ditentukan dalam aksi sumber melalui pipeline.

    Atau, untuk menggunakan AWS CLI untuk menjalankan kembali pipeline, dari terminal di Linux, macOS, atau mesin Unix lokal Anda, atau prompt perintah pada mesin Windows lokal Anda, jalankan perintah, start-pipeline-execution dengan menentukan nama pipeline. Ini menjalankan aplikasi di bucket sumber Anda melalui pipeline untuk kedua kalinya.

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

    Perintah ini mengembalikan `pipelineExecutionId` objek.
11. Kembali ke CodePipeline konsol dan dalam daftar pipeline, pilih MyFirstPipelineuntuk membuka halaman tampilan.

    Pipa menunjukkan tiga tahap dan keadaan artefak yang berjalan melalui tiga tahap tersebut. Mungkin diperlukan waktu hingga lima menit agar pipa berjalan melalui semua tahapan. Anda melihat penerapan berhasil pada dua tahap pertama, seperti sebelumnya, tetapi tahap Produksi menunjukkan tindakan Deploy-Second-Deployment gagal.
12. Dalam tindakan Deploy-Second-Deployment, pilih Detail. Anda diarahkan ke halaman untuk CodeDeploy penyebaran. Dalam kasus ini, kegagalan adalah hasil dari penyebaran grup instans pertama ke semua instans EC2, sehingga tidak ada instance untuk grup penerapan kedua.

    > **Note**
    >
    > Kegagalan ini dirancang, untuk menunjukkan apa yang terjadi ketika ada kegagalan dalam tahap pipa.

Buat tahap ketiga (CLI)

Meskipun menggunakan AWS CLI untuk menambahkan tahap ke pipeline Anda lebih kompleks daripada menggunakan konsol, ini memberikan lebih banyak visibilitas ke dalam struktur pipa.

Untuk membuat tahap ketiga untuk pipeline Anda

1. Buka sesi terminal di Linux, macOS, atau mesin Unix lokal Anda, atau prompt perintah di mesin Windows lokal Anda, dan jalankan get-pipeline perintah untuk menampilkan struktur pipeline yang baru saja Anda buat. Untuk**MyFirstPipeline**, Anda akan mengetik perintah berikut:

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

   Perintah ini mengembalikan struktur MyFirstPipeline. Bagian pertama dari output harus terlihat mirip dengan yang berikut:

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

   Bagian akhir dari output mencakup metadata pipa dan akan terlihat mirip dengan yang berikut:

   ```
       ...
           ],
           "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. Salin dan tempel struktur ini ke editor teks biasa, dan simpan file sebagai. **pipeline.json** Untuk kenyamanan, simpan file ini di direktori yang sama tempat Anda menjalankan aws codepipeline perintah.

> **Note**
>
> Anda dapat menyalurkan JSON langsung ke file dengan get-pipeline perintah sebagai berikut:
>
> ```
> aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
> ```

3. Salin bagian tahap Deploy dan tempel setelah dua tahap pertama. Karena ini adalah tahap penerapan, seperti tahap Deploy, Anda menggunakannya sebagai template untuk tahap ketiga.
4. Ubah nama panggung dan detail grup penyebaran.

   Contoh berikut menunjukkan JSON yang Anda tambahkan ke file pipeline.json setelah tahap Deploy. Edit elemen yang ditekankan dengan nilai-nilai baru. Ingatlah untuk menyertakan koma untuk memisahkan definisi tahap Deploy dan Produksi.

```
,
{
    "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. Jika Anda bekerja dengan struktur pipa yang diambil menggunakan get-pipeline perintah, Anda harus menghapus `metadata` baris dari file JSON. Jika tidak, update-pipeline perintah tidak dapat menggunakannya. Hapus `"metadata": { }` garis dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang.

   Misalnya, hapus baris berikut dari struktur:

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

   Simpan file tersebut.

6. Jalankan update-pipeline perintah, tentukan file JSON pipeline, mirip dengan yang berikut ini:

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

   Perintah ini mengembalikan seluruh struktur pipa yang diperbarui.

   > **Important**
   >
   > Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

7. Jalankan start-pipeline-execution perintah, tentukan nama pipa. Ini menjalankan aplikasi di bucket sumber Anda melalui pipeline untuk kedua kalinya.

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

   Perintah ini mengembalikan `pipelineExecutionId` objek.

8. Buka CodePipeline konsol dan pilih MyFirstPipelinedari daftar saluran pipa.

   Pipa menunjukkan tiga tahap dan keadaan artefak yang berjalan melalui tiga tahap tersebut. Mungkin diperlukan waktu hingga lima menit agar pipa berjalan melalui semua tahapan. Meskipun penerapan berhasil pada dua tahap pertama, seperti sebelumnya, tahap Produksi menunjukkan bahwa tindakan Deploy-Second-Deployment gagal.

9. Dalam tindakan Deploy-Second-Deployment, pilih Detail untuk melihat detail kegagalan. Anda diarahkan ke halaman detail untuk CodeDeploy penyebaran. Dalam kasus ini, kegagalan adalah hasil dari penyebaran grup instans pertama ke semua instans EC2, sehingga tidak ada instance untuk grup penerapan kedua.

> **Note**
>
> Kegagalan ini dirancang, untuk menunjukkan apa yang terjadi ketika ada kegagalan dalam tahap pipa.

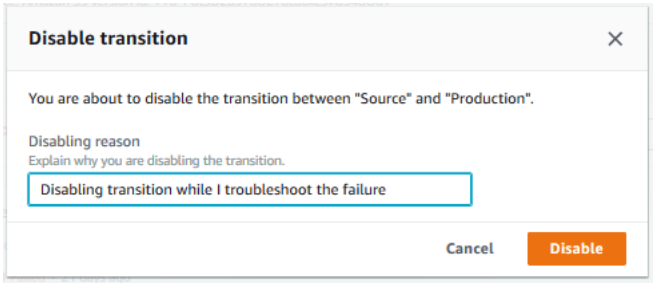
(Opsional) Langkah 6: Nonaktifkan dan aktifkan transisi antar tahapan CodePipeline

Anda dapat mengaktifkan atau menonaktifkan transisi antar tahapan dalam pipeline. Menonaktifkan transisi antar tahapan memungkinkan Anda mengontrol transisi secara manual antara satu tahap dan tahap lainnya. Misalnya, Anda mungkin ingin menjalankan dua tahap pertama dari pipeline, tetapi menonaktifkan transisi ke tahap ketiga hingga Anda siap untuk menerapkan ke produksi, atau saat Anda memecahkan masalah atau kegagalan dengan tahap itu.

Untuk menonaktifkan dan mengaktifkan transisi antar tahapan dalam pipeline CodePipeline

1. Buka CodePipeline konsol dan pilih MyFirstPipelinedari daftar saluran pipa.
2. Pada halaman detail untuk pipeline, pilih tombol Nonaktifkan transisi antara tahap kedua (Deploy) dan tahap ketiga yang Anda tambahkan di bagian sebelumnya (Produksi).
3. Di Nonaktifkan transisi, masukkan alasan untuk menonaktifkan transisi antar tahapan, lalu pilih Nonaktifkan.

   Panah di antara tahapan menampilkan ikon dan perubahan warna, dan tombol Aktifkan transisi.



4. Unggah sampel Anda lagi ke bucket S3. Karena bucket berversi, perubahan ini memulai pipeline.

5. Kembali ke halaman detail untuk pipeline Anda dan perhatikan status tahapannya. Tampilan pipeline berubah untuk menunjukkan kemajuan dan keberhasilan pada dua tahap pertama, tetapi tidak ada perubahan yang terjadi pada tahap ketiga. Proses ini mungkin memakan waktu beberapa menit.
6. Aktifkan transisi dengan memilih tombol Aktifkan transisi antara dua tahap. Dalam kotak dialog Aktifkan transisi, pilih Aktifkan. Tahap mulai berjalan dalam beberapa menit dan mencoba untuk memproses artefak yang telah dijalankan melalui dua tahap pertama dari pipa.

> **Note**
>
> Jika Anda ingin tahap ketiga ini berhasil, edit grup CodePipelineProductionFleet penyebaran sebelum Anda mengaktifkan transisi, dan tentukan kumpulan instans EC2 yang berbeda di mana aplikasi digunakan. Untuk informasi selengkapnya tentang cara melakukannya, lihat Mengubah setelan grup penerapan. Jika Anda membuat lebih banyak instans EC2, Anda mungkin dikenakan biaya tambahan.

Langkah 7: Bersihkan sumber daya

Anda dapat menggunakan beberapa sumber daya yang Anda buat dalam tutorial ini untukTutorial: Buat pipeline empat tahap. Misalnya, Anda dapat menggunakan kembali CodeDeploy aplikasi dan penyebaran. Anda dapat mengonfigurasi tindakan build dengan penyedia seperti CodeBuild, yang merupakan layanan build yang dikelola sepenuhnya di cloud. Anda juga dapat mengonfigurasi tindakan build yang menggunakan penyedia dengan server atau sistem build, seperti Jenkins.

Namun, setelah Anda menyelesaikan ini dan tutorial lainnya, Anda harus menghapus pipeline dan sumber daya yang digunakannya, sehingga Anda tidak dikenakan biaya untuk terus menggunakan sumber daya tersebut. Pertama, hapus pipeline, lalu CodeDeploy aplikasi dan instans Amazon EC2 yang terkait, dan terakhir, bucket S3.

Untuk membersihkan sumber daya yang digunakan dalam tutorial ini

1. Untuk membersihkan CodePipeline sumber daya Anda, ikuti petunjuk di Hapus pipeline di AWS CodePipeline.
2. Untuk membersihkan CodeDeploy sumber daya Anda, ikuti petunjuk di Untuk membersihkan sumber daya (konsol).

3. Untuk menghapus bucket S3, ikuti petunjuk di Menghapus atau mengosongkan ember. Jika Anda tidak bermaksud membuat lebih banyak saluran pipa, hapus bucket S3 yang dibuat untuk menyimpan artefak pipa Anda. Untuk informasi lebih lanjut tentang ember ini, lihatCodePipeline konsep .

Tutorial: Buat pipeline sederhana (CodeCommit repositori)

Dalam tutorial ini, Anda gunakan CodePipeline untuk menyebarkan kode yang dikelola dalam CodeCommit repositori ke satu instans Amazon EC2. Pipeline Anda dipicu saat Anda mendorong perubahan ke CodeCommit repositori. Pipeline menyebarkan perubahan Anda ke instans Amazon EC2 yang CodeDeploy digunakan sebagai layanan penerapan.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Pipa memiliki dua tahap:

- Tahap sumber (Sumber) untuk tindakan CodeCommit sumber Anda.
- Tahap penerapan (Deploy) untuk tindakan CodeDeploy penerapan Anda.

Cara termudah untuk memulai AWS CodePipeline adalah dengan menggunakan wizard Create Pipeline di CodePipeline konsol.

> **Note**
>
> Sebelum memulai, pastikan Anda telah menyiapkan klien Git Anda untuk bekerja dengannya CodeCommit. Untuk petunjuk, lihat Menyiapkan untuk CodeCommit.

Langkah 1: Buat CodeCommit repositori

Pertama, Anda membuat repositori di. CodeCommit Pipeline Anda mendapatkan kode sumber dari repositori ini saat dijalankan. Anda juga membuat repositori lokal tempat Anda memelihara dan memperbarui kode sebelum Anda mendorongnya ke repositori. CodeCommit

Untuk membuat CodeCommit repositori

1. Buka CodeCommit konsol di https://console.aws.amazon.com/codecommit/.
2. Di pemilih Region, pilih Wilayah AWS tempat Anda ingin membuat repositori dan pipeline. Untuk informasi lebih lanjut, lihat Wilayah AWS dan Titik Akhir.
3. Pada halaman Repositori, pilih Buat repositori.
4. Pada halaman Buat repositori, dalam Nama Repositori, ketikkan nama untuk repositori Anda, (misalnya **MyDemoRepo**).
5. Pilih Buat.

> **Note**
>
> Langkah-langkah yang tersisa dalam tutorial ini digunakan **MyDemoRepo** untuk nama CodeCommit repositori Anda. Jika Anda memilih nama yang berbeda, pastikan untuk menggunakannya di seluruh tutorial ini.

Untuk menyiapkan repositori lokal

Pada langkah ini, Anda mengatur repositori lokal untuk terhubung ke repositori jarak jauh CodeCommit Anda.

> **Note**
>
> Anda tidak diharuskan untuk menyiapkan repositori lokal. Anda juga dapat menggunakan konsol untuk mengunggah file seperti yang dijelaskan dalamLangkah 2: Tambahkan kode sampel ke CodeCommit repositori Anda.

1. Dengan repositori baru Anda terbuka di konsol, pilih Clone URL di kanan atas halaman, lalu pilih Clone SSH. Alamat untuk mengkloning repositori Git Anda disalin ke clipboard Anda.

2. Di terminal atau baris perintah Anda, navigasikan ke direktori lokal tempat Anda ingin repositori lokal Anda disimpan. Dalam tutorial ini, kita gunakan`/tmp`.
3. Jalankan perintah berikut untuk mengkloning repositori, mengganti alamat SSH dengan yang Anda salin pada langkah sebelumnya. Perintah ini membuat direktori yang disebut`MyDemoRepo`. Anda menyalin contoh aplikasi ke direktori ini.

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

Langkah 2: Tambahkan kode sampel ke CodeCommit repositori Anda

Pada langkah ini, Anda mengunduh kode untuk aplikasi sampel yang dibuat untuk panduan CodeDeploy sampel, dan menambahkannya ke repositori Anda CodeCommit .

1. Selanjutnya, unduh sampel dan simpan ke folder atau direktori di komputer lokal Anda.
   a. Pilih salah satu dari berikut ini. Pilih `SampleApp_Linux.zip` apakah Anda ingin mengikuti langkah-langkah dalam tutorial ini untuk instance Linux.
      - Jika Anda ingin menyebarkan ke instans Amazon Linux menggunakan CodeDeploy, unduh contoh aplikasi di sini: SampleApp _Linux.zip.
      - Jika Anda ingin menyebarkan ke instance Windows Server menggunakan CodeDeploy, unduh contoh aplikasi di sini: SampleApp _Windows.zip.

      Aplikasi sampel berisi file-file berikut untuk digunakan dengan CodeDeploy:

      - `appspec.yml`— File spesifikasi aplikasi (AppSpecfile) adalah file berformat YAMM yang digunakan oleh CodeDeploy untuk mengelola penyebaran. Untuk informasi selengkapnya tentang AppSpec file, lihat Referensi CodeDeploy AppSpec file di Panduan AWS CodeDeploy Pengguna.
      - `index.html`— File indeks berisi halaman beranda untuk aplikasi sampel yang digunakan.
      - `LICENSE.txt`— File lisensi berisi informasi lisensi untuk aplikasi sampel.
      - File untuk skrip — Aplikasi sampel menggunakan skrip untuk menulis file teks ke lokasi pada instance Anda. Satu file ditulis untuk masing-masing dari beberapa peristiwa siklus hidup CodeDeploy penerapan sebagai berikut:

- (Hanya sampel Linux) `scripts` folder - Folder berisi skrip shell berikut untuk menginstal dependensi dan memulai dan menghentikan aplikasi sampel untuk penerapan otomatis:`install_dependencies`,, dan. `start_server` `stop_server`
- (Hanya sampel Windows) `before-install.bat` - Ini adalah skrip batch untuk peristiwa siklus hidup `BeforeInstall` penerapan, yang akan berjalan untuk menghapus file lama yang ditulis selama penerapan sampel ini sebelumnya dan membuat lokasi pada instance Anda untuk menulis file baru.

b. Unduh file terkompresi (zip).

2. Buka zip file dari SampleApp_Linux.zip ke direktori lokal yang Anda buat sebelumnya (misalnya, `/tmp/MyDemoRepo` atau`c:\temp\MyDemoRepo`).

   Pastikan untuk menempatkan file langsung ke repositori lokal Anda. Jangan sertakan `SampleApp_Linux` folder. Di Linux, macOS, atau mesin Unix lokal Anda, misalnya, direktori dan hierarki file Anda akan terlihat seperti ini:

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

3. Untuk mengunggah file ke repositori Anda, gunakan salah satu metode berikut.

   a. Untuk menggunakan CodeCommit konsol untuk mengunggah file Anda:

      i. Buka CodeCommit konsol, dan pilih repositori Anda dari daftar Repositori.

      ii. Pilih Tambahkan file, lalu pilih Unggah file.

      iii. Pilih Pilih file, lalu telusuri file Anda. Untuk menambahkan file di bawah folder, pilih Buat file dan kemudian masukkan nama folder dengan nama file, seperti`scripts/install_dependencies`. Tempel konten file ke file baru.

      Lakukan perubahan dengan memasukkan nama pengguna dan alamat email Anda.

      Pilih Perubahan commit.

iv. Ulangi langkah ini untuk setiap file.

Isi repositori Anda akan terlihat seperti ini:

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

b. Untuk menggunakan perintah git untuk mengunggah file Anda:

i. Ubah direktori ke repo lokal Anda:

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

ii. Jalankan perintah berikut untuk mementaskan semua file Anda sekaligus:

```
git add -A
```

iii. Jalankan perintah berikut untuk mengkomit file dengan pesan komit:

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

iv. Jalankan perintah berikut untuk mendorong file dari repo lokal Anda ke CodeCommit repositori Anda:

```
git push
```

4. File yang Anda unduh dan tambahkan ke repo lokal Anda sekarang telah ditambahkan ke `main` cabang di CodeCommit `MyDemoRepo` repositori Anda dan siap untuk disertakan dalam pipeline.

Langkah 3: Buat instans Amazon EC2 Linux dan instal agen CodeDeploy

Pada langkah ini, Anda membuat instans Amazon EC2 tempat Anda menerapkan aplikasi sampel. Sebagai bagian dari proses ini, buat peran instance yang memungkinkan penginstalan dan pengelolaan CodeDeploy agen pada instance. CodeDeploy Agen adalah paket perangkat lunak yang

memungkinkan instance untuk digunakan dalam CodeDeploy penerapan. Anda juga melampirkan kebijakan yang memungkinkan instance mengambil file yang digunakan CodeDeploy agen untuk menyebarkan aplikasi Anda dan mengizinkan instance dikelola oleh SSM.

Untuk membuat peran instance

1. Buka konsol IAM di https://console.aws.amazon.com/iam/).
2. Dari dasbor konsol, pilih Peran.
3. Pilih Buat peran.
4. Di bawah Pilih jenis entitas tepercaya, pilih Layanan AWS. Di bawah Pilih kasus penggunaan, pilih EC2. Di bawah Pilih kasus penggunaan Anda, pilih EC2. Pilih Berikutnya: Izin.
5. Cari dan pilih kebijakan yang diberi nama **`AmazonEC2RoleforAWSCodeDeploy`**.
6. Cari dan pilih kebijakan yang diberi nama **`AmazonSSMManagedInstanceCore`**. Pilih Berikutnya: Tanda.
7. Pilih Berikutnya: Tinjau. Masukkan nama untuk peran (misalnya,**`EC2InstanceRole`**).

   > **Note**
   >
   > Catat nama peran Anda untuk langkah selanjutnya. Anda memilih peran ini saat membuat instance Anda.

   Pilih Buat peran.

Untuk meluncurkan sebuah instans

1. Buka konsol Amazon EC2 di. https://console.aws.amazon.com/ec2/
2. Dari navigasi samping, pilih Instans, dan pilih Luncurkan instance dari bagian atas halaman.
3. Di Nama, masukkan **`MyCodePipelineDemo`**. Ini memberikan contoh tag Key dari **`Name`** dan tag Nilai dari. **`MyCodePipelineDemo`** Kemudian, Anda membuat CodeDeploy aplikasi yang menyebarkan aplikasi sampel ke instance ini. CodeDeploymemilih instance untuk menyebarkan berdasarkan tag.
4. Di bawah Application and OS Images (Amazon Machine Image), cari opsi Amazon Linux AMI dengan AWS logo, dan pastikan itu dipilih. (AMI ini digambarkan sebagai Amazon Linux 2 AMI (HVM) dan diberi label “Free tier eligible”.)

5. Di bawah Jenis instans, pilih `t2.micro` tipe yang memenuhi syarat tingkat gratis sebagai konfigurasi perangkat keras untuk instans Anda.
6. Di bawah Key pair (login), pilih key pair atau buat satu.

   Anda juga dapat memilih Proceed without a key pair.

   > **Note**
   >
   > Untuk keperluan tutorial ini, Anda dapat melanjutkan tanpa key pair. Untuk menggunakan SSH untuk terhubung ke instance Anda, buat atau gunakan key pair.

7. Di bawah Pengaturan jaringan, lakukan hal berikut.

   Di Auto-assign IP Publik, pastikan statusnya Aktifkan.

   Untuk grup keamanan yang dibuat, pilih HTTP, lalu di bawah Jenis sumber, pilih IP Saya.
8. Perluas Detail lanjutan. Di profil instans IAM, pilih peran IAM yang Anda buat di prosedur sebelumnya (misalnya,**`EC2InstanceRole`**).
9. Di bawah Ringkasan, di bawah Jumlah instance, masukkan.. 1
10. Pilih Luncurkan instans.
11. Anda dapat melihat status peluncuran di halaman Instans. Saat Anda meluncurkan instans, status awalnya adalah `pending`. Setelah instans dimulai, statusnya berubah menjadi `running`, dan ia menerima nama DNS publik. (Jika kolom DNS Publik tidak ditampilkan, pilih ikon Tampilkan/Sembunyikan, lalu pilih DNS Publik.)

Langkah 4: Buat aplikasi di CodeDeploy

Di CodeDeploy, aplikasi adalah sumber daya yang berisi aplikasi perangkat lunak yang ingin Anda gunakan. Kemudian, Anda menggunakan aplikasi ini CodePipeline untuk mengotomatiskan penerapan aplikasi sampel ke instans Amazon EC2 Anda.

Pertama, Anda membuat peran yang memungkinkan CodeDeploy untuk melakukan penerapan. Kemudian, Anda membuat CodeDeploy aplikasi.

Untuk membuat peran CodeDeploy layanan

1. Buka konsol IAM di https://console.aws.amazon.com/iam/).
2. Dari dasbor konsol, pilih Peran.

3. Pilih Buat peran.
4. Di bawah Pilih entitas tepercaya, pilih Layanan AWS. Di bawah Kasus penggunaan, pilih CodeDeploy. Pilih CodeDeploydari opsi yang tercantum. Pilih Berikutnya. Kebijakan yang `AWSCodeDeployRole` dikelola sudah melekat pada peran tersebut.
5. Pilih Berikutnya.
6. Masukkan nama untuk peran (misalnya,**`CodeDeployRole`**), lalu pilih Buat peran.

Untuk membuat aplikasi di CodeDeploy

1. Buka CodeDeploy konsol di https://console.aws.amazon.com/codedeploy.
2. Jika halaman Aplikasi tidak muncul, pada menu, pilih Aplikasi.
3. Pilih Create application (Buat aplikasi).
4. Dalam nama Aplikasi, masukkan**`MyDemoApplication`**.
5. Di Compute Platform, pilih EC2/On-premise.
6. Pilih Create application (Buat aplikasi).

Untuk membuat grup penyebaran di CodeDeploy

Grup penerapan adalah sumber daya yang mendefinisikan setelan terkait penerapan seperti instance mana yang akan digunakan dan seberapa cepat menerapkannya.

1. Pada halaman yang menampilkan aplikasi Anda, pilih Buat grup penyebaran.
2. Dalam nama grup Deployment, masukkan**`MyDemoDeploymentGroup`**.
3. Dalam peran Layanan, pilih ARN dari peran layanan yang Anda buat sebelumnya (misalnya,**`arn:aws:iam::`*`account_ID`*`:role/CodeDeployRole`**).
4. Di bawah Jenis Deployment, pilih In-place.
5. Di bawah Konfigurasi lingkungan, pilih Instans Amazon EC2. Di bidang Key, masukkan**`Name`**. Di bidang Nilai, masukkan nama yang Anda gunakan untuk menandai instance (misalnya,**`MyCodePipelineDemo`**).
6. Di bawah Konfigurasi Agen dengan AWS Systems Manager, pilih Sekarang dan jadwalkan pembaruan. Ini menginstal agen pada instance. Instance Linux sudah dikonfigurasi dengan agen SSM dan sekarang akan diperbarui dengan CodeDeploy agen.
7. Di bawah konfigurasi Deployment, pilih`CodeDeployDefault.OneAtaTime`.

8. Di bawah Load Balancer, pastikan Aktifkan load balancing tidak dipilih. Anda tidak perlu mengatur penyeimbang beban atau memilih grup target untuk contoh ini.
9. Pilih Buat grup penyebaran.

Langkah 5: Buat pipeline pertama Anda CodePipeline

Anda sekarang siap untuk membuat dan menjalankan pipeline pertama Anda. Pada langkah ini, Anda membuat pipeline yang berjalan secara otomatis saat kode didorong ke CodeCommit repositori Anda.

Untuk membuat CodePipeline pipa

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyFirstPipeline`**.
5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
6. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.
7. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.
8. Pada Langkah 3: Tambahkan tahap sumber, di penyedia Sumber, pilih CodeCommit. Dalam nama Repositori, pilih nama CodeCommit repositori yang Anda buat. Langkah 1: Buat CodeCommit repositori Di Nama cabang, pilih`main`, lalu pilih Langkah berikutnya.

   Setelah Anda memilih nama repositori dan cabang, pesan akan menampilkan aturan Amazon CloudWatch Events yang akan dibuat untuk pipeline ini.

Di bawah Ubah opsi deteksi, biarkan default. Hal ini memungkinkan CodePipeline untuk menggunakan Amazon CloudWatch Events untuk mendeteksi perubahan dalam repositori sumber Anda.

Pilih Berikutnya.

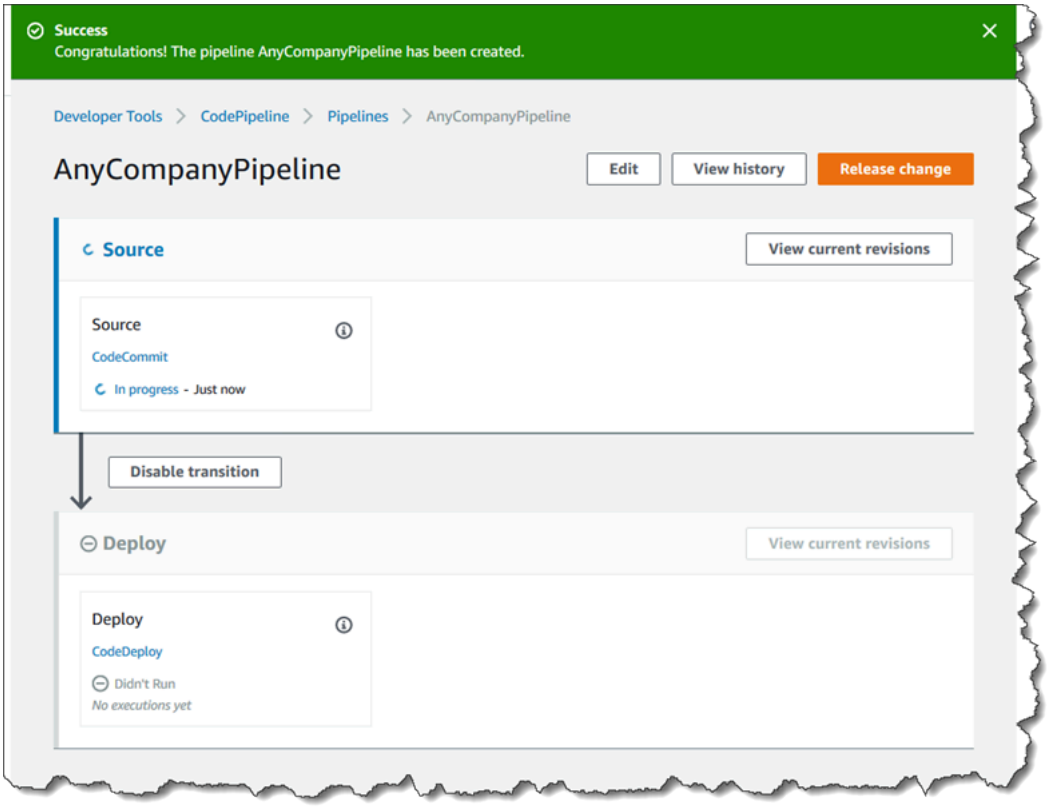
9. Pada Langkah 4: Tambahkan tahap build, pilih Lewati tahap build, lalu terima pesan peringatan dengan memilih Lewati lagi. Pilih Berikutnya.

> **Note**
>
> Dalam tutorial ini, Anda menerapkan kode yang tidak memerlukan layanan build, sehingga Anda dapat melewati langkah ini. Namun, jika kode sumber Anda perlu dibangun sebelum diterapkan ke instance, Anda dapat mengonfigurasinya CodeBilddi langkah ini.

10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

11. Pada Langkah 6: Tambahkan tahap penerapan, di Penyedia penyebaran, pilih. CodeDeploy Di Nama aplikasi, pilih**`MyDemoApplication`**. Di grup Deployment, pilih**`MyDemoDeploymentGroup`**, lalu pilih Langkah berikutnya.
12. Pada Langkah 7: Tinjau, tinjau informasinya, lalu pilih Buat pipeline.
13. Pipa mulai berjalan setelah dibuat. Ini mengunduh kode dari CodeCommit repositori Anda dan membuat CodeDeploy penerapan ke instans EC2 Anda. Anda dapat melihat pesan kemajuan dan keberhasilan dan kegagalan saat CodePipeline sampel menyebarkan halaman web ke instans Amazon EC2 dalam penerapan. CodeDeploy



Selamat! Anda baru saja membuat pipa sederhana di CodePipeline.

Selanjutnya, Anda memverifikasi hasilnya.

Untuk memverifikasi bahwa pipeline Anda berhasil berjalan

1. Lihat kemajuan awal pipa. Status setiap tahap berubah dari Tidak ada eksekusi yang belum ada dalam proses, dan kemudian menjadi Berhasil atau Gagal. Pipa harus menyelesaikan proses pertama dalam beberapa menit.
2. Setelah Succeeded ditampilkan untuk status pipeline, di area status untuk tahap Deploy, pilih. CodeDeploy Ini membuka CodeDeploy konsol. Jika Berhasil tidak ditampilkan lihatPemecahan masalah CodePipeline.
3. Pada tab Deployment, pilih ID deployment. Pada halaman untuk penerapan, di bawah peristiwa siklus hidup Deployment, pilih ID instance. Ini membuka konsol EC2.
4. Pada tab Deskripsi, di DNS Publik, salin alamat (misalnya,`ec2-192-0-2-1.us-west-2.compute.amazonaws.com`), lalu tempelkan ke bilah alamat browser web Anda.

Halaman web ditampilkan untuk contoh aplikasi yang Anda unduh dan dorong ke CodeCommit repositori Anda.

Untuk informasi selengkapnya tentang tahapan, tindakan, dan cara kerja saluran pipa, lihatCodePipeline konsep .

Langkah 6: Ubah kode di CodeCommit repositori Anda

Pipeline Anda dikonfigurasi untuk berjalan setiap kali perubahan kode dilakukan ke CodeCommit repositori Anda. Pada langkah ini, Anda membuat perubahan pada file HTML yang merupakan bagian dari CodeDeploy aplikasi sampel di CodeCommit repositori. Saat Anda mendorong perubahan ini, pipeline Anda berjalan lagi, dan perubahan yang Anda buat terlihat di alamat web yang Anda akses sebelumnya.

1. Ubah direktori ke repo lokal Anda:

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

2. Gunakan editor teks untuk memodifikasi `index.html` file:

```
(For Linux or Unix)gedit index.html
(For OS X)open -e index.html
(For Windows)notepad index.html
```

3. Merevisi isi `index.html` file untuk mengubah warna latar belakang dan beberapa teks pada halaman web, dan kemudian menyimpan file.

```
<!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. Komit dan dorong perubahan Anda ke CodeCommit repositori Anda dengan menjalankan perintah berikut, satu per satu:

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

```
git push
```

Untuk memverifikasi pipeline Anda berjalan dengan sukses

1. Lihat kemajuan awal pipa. Status setiap tahap berubah dari Tidak ada eksekusi yang belum ada dalam proses, dan kemudian menjadi Berhasil atau Gagal. Pengerjaan pipa harus selesai dalam beberapa menit.

2. Setelah Berhasil ditampilkan untuk status tindakan, segarkan halaman demo yang Anda akses sebelumnya di browser Anda.

   Halaman web yang diperbarui ditampilkan.

Langkah 7: Bersihkan sumber daya

Anda dapat menggunakan beberapa sumber daya yang Anda buat dalam tutorial ini untuk tutorial lain dalam panduan ini. Misalnya, Anda dapat menggunakan kembali CodeDeploy aplikasi dan penyebaran. Namun, setelah Anda menyelesaikan ini dan tutorial lainnya, Anda harus menghapus pipeline dan sumber daya yang digunakannya sehingga Anda tidak dikenakan biaya untuk terus menggunakan sumber daya tersebut. Pertama, hapus pipeline, lalu CodeDeploy aplikasi dan instans Amazon EC2 yang terkait, dan terakhir, repositori. CodeCommit

Untuk membersihkan sumber daya yang digunakan dalam tutorial ini

1. Untuk membersihkan CodePipeline sumber daya Anda, ikuti petunjuk di Hapus pipeline di AWS CodePipeline.
2. Untuk membersihkan CodeDeploy sumber daya Anda, ikuti petunjuk di Clean Up Deployment Walkthrough Resources.
3. Untuk menghapus CodeCommit repositori, ikuti instruksi di Hapus repositori. CodeCommit

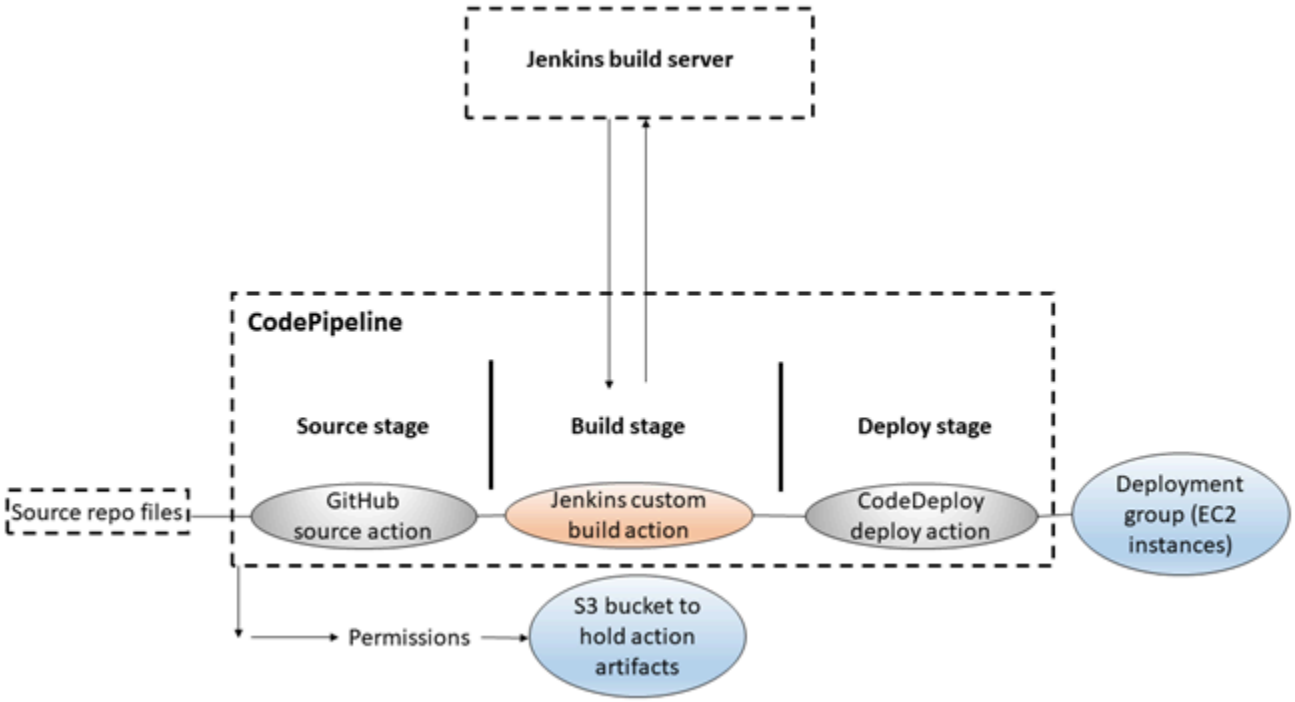
Langkah 8: Bacaan lebih lanjut

Pelajari lebih lanjut tentang cara CodePipeline kerja:

- Untuk informasi selengkapnya tentang tahapan, tindakan, dan cara kerja saluran pipa, lihatCodePipeline konsep .
- Untuk informasi tentang tindakan yang dapat Anda lakukan dengan menggunakan CodePipeline, lihatIntegrasi dengan tipe CodePipeline tindakan.
- Coba tutorial yang lebih canggih ini,Tutorial: Buat pipeline empat tahap. Ini menciptakan pipeline multi-tahap yang mencakup langkah yang membangun kode sebelum diterapkan.

Tutorial: Buat pipeline empat tahap

Sekarang setelah Anda membuat pipeline pertama di Tutorial: Buat pipeline sederhana (ember S3) atauTutorial: Buat pipeline sederhana (CodeCommit repositori), Anda dapat mulai membuat pipeline yang lebih kompleks. Tutorial ini akan memandu Anda melalui pembuatan pipeline empat tahap yang menggunakan GitHub repositori untuk sumber Anda, server build Jenkins untuk membangun proyek, dan CodeDeploy aplikasi untuk menyebarkan kode yang dibangun ke server pementasan. Diagram berikut menunjukkan pipa tiga tahap awal.



Setelah pipeline dibuat, Anda akan mengeditnya untuk menambahkan tahap dengan tindakan pengujian untuk menguji kode, juga menggunakan Jenkins.

Sebelum Anda dapat membuat pipeline ini, Anda harus mengkonfigurasi sumber daya yang diperlukan. Misalnya, jika Anda ingin menggunakan GitHub repositori untuk kode sumber Anda, Anda harus membuat repositori sebelum Anda dapat menambahkannya ke pipeline. Sebagai bagian dari pengaturan, tutorial ini memandu Anda melalui pengaturan Jenkins pada instance EC2 untuk tujuan demonstrasi.

> **Important**
>
> Banyak tindakan yang Anda tambahkan ke pipeline dalam prosedur ini melibatkan AWS sumber daya yang perlu Anda buat sebelum membuat pipeline. AWS sumber daya untuk

tindakan sumber Anda harus selalu dibuat di AWS Wilayah yang sama tempat Anda membuat pipeline. Misalnya, jika Anda membuat pipeline di Wilayah AS Timur (Ohio), CodeCommit repositori Anda harus berada di Wilayah AS Timur (Ohio).
Anda dapat menambahkan tindakan lintas wilayah saat membuat pipeline. AWS sumber daya untuk tindakan lintas wilayah harus berada di AWS Wilayah yang sama di mana Anda berencana untuk menjalankan tindakan. Untuk informasi selengkapnya, lihat Menambahkan tindakan Lintas wilayah di CodePipeline.

Important

Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Sebelum Anda memulai tutorial ini, Anda seharusnya sudah menyelesaikan prasyarat umum di. Memulai dengan CodePipeline

Topik

- Langkah 1: Prasyarat lengkap
- Langkah 2: Buat pipeline di CodePipeline
- Langkah 3: Tambahkan tahap lain ke pipeline Anda
- Langkah 4: Bersihkan Sumber Daya

Langkah 1: Prasyarat lengkap

Untuk mengintegrasikan dengan Jenkins, AWS CodePipeline mengharuskan Anda untuk menginstal CodePipeline Plugin untuk Jenkins pada setiap instance Jenkins yang ingin Anda gunakan. CodePipeline Anda juga harus mengonfigurasi pengguna atau peran IAM khusus yang akan digunakan untuk izin antara proyek Jenkins Anda dan. CodePipeline Cara termudah untuk mengintegrasikan Jenkins dan CodePipeline menginstal Jenkins pada instans EC2 yang menggunakan peran instans IAM yang Anda buat untuk integrasi Jenkins. Agar tautan dalam pipeline agar tindakan Jenkins berhasil terhubung, Anda harus mengonfigurasi pengaturan proxy dan firewall

di server atau instans EC2 untuk memungkinkan koneksi masuk ke port yang digunakan oleh proyek Jenkins Anda. Pastikan Anda telah mengonfigurasi Jenkins untuk mengautentikasi pengguna dan menerapkan kontrol akses sebelum Anda mengizinkan koneksi pada port tersebut (misalnya, 443 dan 8443 jika Anda telah mengamankan Jenkins untuk hanya menggunakan koneksi HTTPS, atau 80 dan 8080 jika Anda mengizinkan koneksi HTTP). Untuk informasi lebih lanjut, lihat Mengamankan Jenkins.

> **Note**
>
> Tutorial ini menggunakan contoh kode dan mengkonfigurasi langkah-langkah build yang mengonversi sampel dari Haml ke HTML. Anda dapat mengunduh kode sampel sumber terbuka dari GitHub repositori dengan mengikuti langkah-langkahnya. Salin atau kloning sampel ke dalam repositori GitHub Anda akan membutuhkan seluruh sampel di GitHub repositori Anda, bukan hanya file.zip.
> Tutorial ini juga mengasumsikan bahwa:
>
> - Anda akrab dengan menginstal dan mengelola Jenkins dan membuat proyek Jenkins.
> - Anda telah menginstal Rake dan permata Haml untuk Ruby di komputer atau instance yang sama yang meng-host proyek Jenkins Anda.
> - Anda telah mengatur variabel lingkungan sistem yang diperlukan sehingga perintah Rake dapat dijalankan dari terminal atau baris perintah (misalnya, pada sistem Windows, memodifikasi variabel PATH untuk menyertakan direktori tempat Anda menginstal Rake).

Topik

- Salin atau kloning sampel ke dalam repositori GitHub
- Buat peran IAM untuk digunakan untuk integrasi Jenkins
- Instal dan konfigurasikan Jenkins dan CodePipeline Plugin untuk Jenkins

Salin atau kloning sampel ke dalam repositori GitHub

Untuk mengkloning sampel dan mendorong ke repositori GitHub

1. Unduh kode sampel dari GitHub repositori, atau kloning repositori ke komputer lokal Anda. Ada dua paket sampel:

- Jika Anda akan menerapkan sampel Anda ke instans Amazon Linux, RHEL, atau Ubuntu Server, pilih _linux.zip. codepipeline-jenkins-aws-codedeploy
- Jika Anda akan menerapkan sampel Anda ke instance Windows Server, pilih CodePipeline-Jenkins- .zip. AWSCodeDeploy_Windows

2. Dari repositori, pilih Fork untuk mengkloning repo sampel menjadi repo di akun Github Anda. Lihat informasi yang lebih lengkap dalam dokumentasi GitHub.

Buat peran IAM untuk digunakan untuk integrasi Jenkins

Sebagai praktik terbaik, pertimbangkan untuk meluncurkan instans EC2 untuk meng-host server Jenkins Anda dan menggunakan peran IAM untuk memberikan instance izin yang diperlukan untuk berinteraksi dengannya. CodePipeline

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di konsol IAM, di panel navigasi, pilih Peran, lalu pilih Buat peran.
3. Di bawah Pilih jenis entitas tepercaya, pilih Layanan AWS. Di bawah Pilih layanan yang akan menggunakan peran ini, pilih EC2. Di bawah Pilih kasus penggunaan Anda, pilih EC2.
4. Pilih Berikutnya: Izin. Pada halaman Lampirkan kebijakan izin, pilih kebijakan `AWSCodePipelineCustomActionAccess` terkelola, lalu pilih Berikutnya: Tag. Pilih Berikutnya: Tinjauan.
5. Pada halaman Tinjauan, dalam nama Peran, masukkan nama peran yang akan dibuat khusus untuk integrasi Jenkins (misalnya,*`JenkinsAccess`*), lalu pilih Buat peran.

Saat Anda membuat instans EC2 di mana Anda akan menginstal Jenkins, di Langkah 3: Konfigurasikan Detail Instance, pastikan Anda memilih peran instance (misalnya,*`JenkinsAccess`*).

Untuk informasi selengkapnya tentang peran instans dan Amazon EC2, lihat peran IAM untuk Amazon EC2, Menggunakan Peran IAM untuk Memberikan Izin ke Aplikasi yang Berjalan di Instans Amazon EC2, dan Membuat peran untuk mendelegasikan izin ke. Layanan AWS

Instal dan konfigurasikan Jenkins dan CodePipeline Plugin untuk Jenkins

Untuk menginstal Jenkins dan CodePipeline Plugin untuk Jenkins

1. Buat instans EC2 tempat Anda akan menginstal Jenkins, dan di Langkah 3: Konfigurasikan Detail Instance, pastikan Anda memilih peran instans yang Anda buat (misalnya,*JenkinsAccess*). Untuk informasi selengkapnya tentang membuat instans EC2, lihat Meluncurkan instans Amazon EC2 di Panduan Pengguna Amazon EC2.

   > **Note**
   >
   > Jika Anda sudah memiliki sumber daya Jenkins yang ingin Anda gunakan, Anda dapat melakukannya, tetapi Anda harus membuat pengguna IAM khusus, menerapkan kebijakan `AWSCodePipelineCustomActionAccess` terkelola ke pengguna itu, dan kemudian mengonfigurasi dan menggunakan kredensil akses untuk pengguna tersebut di sumber daya Jenkins Anda. Jika Anda ingin menggunakan UI Jenkins untuk menyediakan kredensialnya, konfigurasikan Jenkins agar hanya mengizinkan HTTPS. Untuk informasi selengkapnya, lihat Pemecahan masalah CodePipeline.

2. Instal Jenkins pada instans EC2. Untuk informasi lebih lanjut, lihat dokumentasi Jenkins untuk menginstal Jenkins dan memulai dan mengakses Jenkins, serta di. details of integration with Jenkins Integrasi produk dan layanan dengan CodePipeline
3. Luncurkan Jenkins, dan di halaman beranda, pilih Kelola Jenkins.
4. Pada halaman Kelola Jenkins, pilih Kelola Plugin.
5. Pilih tab Tersedia, dan di kotak pencarian Filter, masukkan**`AWS CodePipeline`**. Pilih CodePipeline Plugin untuk Jenkins dari daftar dan pilih Unduh sekarang dan instal setelah restart.
6. Pada halaman Instalasi Plugin/Upgrade, pilih Restart Jenkins ketika instalasi selesai dan tidak ada pekerjaan yang berjalan.
7. Pilih Kembali ke Dasbor.
8. Di halaman utama, pilih Item Baru.
9. Di Nama Item, masukkan nama untuk proyek Jenkins (misalnya,*MyDemoProject*). Pilih Freestyle project, lalu pilih OK.

> **Note**
>
> Pastikan bahwa nama untuk proyek Anda memenuhi persyaratan untuk CodePipeline. Untuk informasi selengkapnya, lihat Kuota di AWS CodePipeline.

10. Pada halaman konfigurasi untuk proyek, pilih kotak centang Execute concurrent build jika perlu. Di Manajemen Kode Sumber, pilih AWS CodePipeline. Jika Anda telah menginstal Jenkins pada instans EC2 dan mengonfigurasi AWS CLI dengan profil untuk pengguna IAM yang Anda buat untuk integrasi antara CodePipeline dan Jenkins, biarkan semua bidang lainnya kosong.
11. Pilih Advanced, dan di Provider, masukkan nama untuk penyedia tindakan seperti yang akan muncul di CodePipeline (misalnya,*MyJenkinsProviderName*). Pastikan nama ini unik dan mudah diingat. Anda akan menggunakannya ketika Anda menambahkan tindakan build ke pipeline Anda nanti dalam tutorial ini, dan lagi ketika Anda menambahkan tindakan pengujian.

> **Note**
>
> Nama tindakan ini harus memenuhi persyaratan penamaan untuk tindakan di CodePipeline. Untuk informasi selengkapnya, lihat Kuota di AWS CodePipeline.

12. Di Build Triggers, kosongkan kotak centang apa pun, lalu pilih Poll SCM. Dalam Jadwal, masukkan lima tanda bintang yang dipisahkan oleh spasi, sebagai berikut:

```
* * * * *
```

    Jajak pendapat ini dilakukan CodePipeline setiap menit.

13. Di Build, pilih Add build step. Pilih Execute shell (Amazon Linux, RHEL, atau Ubuntu Server) Jalankan perintah batch (Windows Server), lalu masukkan yang berikut ini:

```
rake
```

> **Note**
>
> Pastikan lingkungan Anda dikonfigurasi dengan variabel dan pengaturan yang diperlukan untuk menjalankan rake; jika tidak, build akan gagal.

14. Pilih Add post-build action, lalu pilih AWS CodePipeline Publisher. Pilih Tambah, dan di Build Output Locations, biarkan lokasi kosong. Konfigurasi ini adalah default. Ini akan membuat file terkompresi di akhir proses pembuatan.
15. Pilih Simpan untuk menyimpan proyek Jenkins Anda.

Langkah 2: Buat pipeline di CodePipeline

Di bagian tutorial ini, Anda membuat pipeline menggunakan wizard Create Pipeline.

Untuk membuat proses rilis CodePipeline otomatis

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Jika perlu, gunakan pemilih Wilayah untuk mengubah Wilayah ke wilayah tempat sumber daya pipa Anda berada. Misalnya, jika Anda membuat sumber daya untuk tutorial sebelumnya di `us-east-2`, pastikan pemilih Region diatur ke US East (Ohio).

   Untuk informasi selengkapnya tentang Wilayah dan titik akhir yang tersedia CodePipeline, lihat AWS CodePipeline titik akhir dan kuota.
3. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
4. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
5. Pada Langkah 2: Pilih halaman pengaturan pipeline, dalam nama Pipeline, masukkan nama untuk pipeline Anda.
6. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
7. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.
8. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, dan pilih Berikutnya.
9. Pada Langkah 3: Tambahkan halaman tahap sumber, di penyedia Sumber, pilih GitHub.
10. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.
11. Pada Langkah 4: Tambahkan tahap build, pilih Add Jenkins. Di nama Provider, masukkan nama tindakan yang Anda berikan di CodePipeline Plugin untuk Jenkins

(misalnya*MyJenkinsProviderName*). Nama ini harus sama persis dengan nama di CodePipeline Plugin untuk Jenkins. Di URL Server, masukkan URL instance EC2 tempat Jenkins diinstal. Dalam nama Proyek, masukkan nama proyek yang Anda buat di Jenkins, seperti*MyDemoProject*, dan kemudian pilih Berikutnya.

12. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

13. Pada Langkah 6: Tambahkan tahap penerapan, gunakan kembali grup CodeDeploy aplikasi dan penyebaran yang Anda buat. Tutorial: Buat pipeline sederhana (ember S3) Di Penyedia Deploy, pilih CodeDeploy. Di Nama aplikasi**`CodePipelineDemoApplication`**, masukkan, atau pilih tombol refresh, lalu pilih nama aplikasi dari daftar. Di grup Deployment**`CodePipelineDemoFleet`**, masukkan, atau pilih dari daftar, lalu pilih Berikutnya.

    > **Note**
    >
    > Anda dapat menggunakan CodeDeploy sumber daya Anda sendiri atau membuat yang baru, tetapi Anda mungkin dikenakan biaya tambahan.

14. Pada Langkah 7: Tinjau, tinjau informasinya, lalu pilih Buat pipeline.
15. Pipa secara otomatis memulai dan menjalankan sampel melalui pipa. Anda dapat melihat pesan kemajuan dan keberhasilan dan kegagalan saat pipeline membangun sampel Haml ke HTML dan menerapkannya sebagai halaman web ke setiap instans Amazon EC2 dalam penerapan. CodeDeploy

Langkah 3: Tambahkan tahap lain ke pipeline Anda

Sekarang Anda akan menambahkan tahap pengujian dan kemudian tindakan pengujian ke tahap yang menggunakan tes Jenkins yang disertakan dalam sampel untuk menentukan apakah halaman web memiliki konten apa pun. Tes ini hanya untuk tujuan demonstrasi.

> **Note**
>
> Jika Anda tidak ingin menambahkan tahap lain ke pipeline, Anda dapat menambahkan tindakan pengujian ke tahap Pementasan pipeline, sebelum atau sesudah tindakan penerapan.

Tambahkan tahap pengujian ke pipeline Anda

Topik

- Cari alamat IP dari sebuah instans
- Buat proyek Jenkins untuk menguji penerapan
- Buat tahap keempat

Cari alamat IP dari sebuah instans

Untuk memverifikasi alamat IP dari sebuah instance tempat Anda menerapkan kode

1. Setelah Berhasil ditampilkan untuk status pipeline, di area status untuk tahap Pementasan, pilih Detail.
2. Di bagian Detail Deployment, di ID Instance, pilih ID instans dari salah satu instance yang berhasil diterapkan.
3. Salin alamat IP instance (misalnya,*192.168.0.4*). Anda akan menggunakan alamat IP ini dalam tes Jenkins Anda.

Buat proyek Jenkins untuk menguji penerapan

Untuk membuat proyek Jenkins

1. Pada contoh di mana Anda menginstal Jenkins, buka Jenkins dan dari halaman utama, pilih Item Baru.
2. Di Nama Item, masukkan nama untuk proyek Jenkins (misalnya,*MyTestProject*). Pilih Freestyle project, lalu pilih OK.

   > **Note**
   >
   > Pastikan bahwa nama untuk proyek Anda memenuhi CodePipeline persyaratan. Untuk informasi selengkapnya, lihat Kuota di AWS CodePipeline.

3. Pada halaman konfigurasi untuk proyek, pilih kotak centang Execute concurrent build jika perlu. Di Manajemen Kode Sumber, pilih AWS CodePipeline. Jika Anda telah menginstal Jenkins pada instans EC2 dan mengkonfigurasi AWS CLI dengan profil untuk pengguna IAM yang Anda buat untuk integrasi antara CodePipeline dan Jenkins, biarkan semua bidang lainnya kosong.

> **Important**
>
> Jika Anda mengonfigurasi proyek Jenkins dan tidak diinstal pada instans Amazon EC2, atau diinstal pada instans EC2 yang menjalankan sistem operasi Windows, lengkapi bidang seperti yang dipersyaratkan oleh host proxy dan pengaturan port Anda, dan berikan kredensi pengguna IAM atau peran yang Anda konfigurasikan untuk integrasi antara Jenkins dan. CodePipeline

4. Pilih Advanced, dan di Kategori, pilih Test.
5. Di Provider, masukkan nama yang sama dengan yang Anda gunakan untuk proyek build (misalnya,*MyJenkinsProviderName*). Anda akan menggunakan nama ini ketika Anda menambahkan tindakan pengujian ke pipeline Anda nanti dalam tutorial ini.

> **Note**
>
> Nama ini harus memenuhi persyaratan CodePipeline penamaan untuk tindakan. Untuk informasi selengkapnya, lihat Kuota di AWS CodePipeline.

6. Di Build Triggers, kosongkan kotak centang apa pun, lalu pilih Poll SCM. Dalam Jadwal, masukkan lima tanda bintang yang dipisahkan oleh spasi, sebagai berikut:

```
* * * * *
```

Jajak pendapat ini dilakukan CodePipeline setiap menit.

7. Di Build, pilih Add build step. Jika Anda menggunakan instance Amazon Linux, RHEL, atau Ubuntu Server, pilih Execute shell. Kemudian masukkan yang berikut ini, di mana alamat IP adalah alamat instans EC2 yang Anda salin sebelumnya:

```
TEST_IP_ADDRESS=192.168.0.4 rake test
```

Jika Anda menyebarkan ke instance Windows Server, pilih Jalankan perintah batch, lalu masukkan yang berikut ini, di mana alamat IP adalah alamat instans EC2 yang Anda salin sebelumnya:

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

> **Note**
>
> Tes ini mengasumsikan port default 80. Jika Anda ingin menentukan port yang berbeda, tambahkan pernyataan port uji, sebagai berikut:
>
> ```
> TEST_IP_ADDRESS=192.168.0.4 TEST_PORT=8000 rake test
> ```

8. Pilih Add post-build action, lalu pilih AWS CodePipeline Publisher. Jangan pilih Tambah.
9. Pilih Simpan untuk menyimpan proyek Jenkins Anda.

Buat tahap keempat

Untuk menambahkan panggung ke pipeline Anda yang mencakup tindakan uji Jenkins

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Di Nama, pilih nama pipeline yang Anda buat, MySecondPipeline.
3. Pada halaman detail pipeline, pilih Edit.
4. Pada halaman Edit, pilih + Stage untuk menambahkan stage segera setelah tahap Build.
5. Di bidang nama untuk tahap baru, masukkan nama (misalnya,**`Testing`**), lalu pilih + Tambahkan grup tindakan.
6. Dalam nama Action, masukkan*`MyJenkinsTest-Action`*. Di Penyedia uji, pilih nama penyedia yang Anda tentukan di Jenkins (misalnya,*`MyJenkinsProviderName`*). Dalam nama Proyek, masukkan nama proyek yang Anda buat di Jenkins (misalnya,*`MyTestProject`*). Di artefak Input, pilih artefak dari build Jenkins yang nama defaultnya***`BuildArtifact`***, lalu pilih Selesai.

   > **Note**
   >
   > Karena tindakan pengujian Jenkins beroperasi pada aplikasi yang dibangun pada langkah pembuatan Jenkins, gunakan artefak build untuk artefak input ke tindakan pengujian.

   Untuk informasi lebih lanjut tentang artefak input dan output dan struktur pipa, lihat. CodePipeline referensi struktur pipa

7. Pada halaman Edit, pilih Simpan perubahan pipeline. Dalam kotak dialog Simpan perubahan pipeline, pilih Simpan dan lanjutkan.
8. Meskipun tahap baru telah ditambahkan ke pipeline Anda, status Tidak ada eksekusi belum ditampilkan untuk tahap itu karena tidak ada perubahan yang memicu proses pipeline lainnya. Untuk menjalankan sampel melalui pipeline yang direvisi, pada halaman detail pipeline, pilih Rilis perubahan.

   Tampilan pipeline menunjukkan tahapan dan tindakan dalam pipeline Anda dan status revisi yang berjalan melalui empat tahap tersebut. Waktu yang dibutuhkan pipa untuk berjalan melalui semua tahapan akan tergantung pada ukuran artefak, kompleksitas tindakan pembuatan dan pengujian Anda, dan faktor lainnya.

Langkah 4: Bersihkan Sumber Daya

Setelah Anda menyelesaikan tutorial ini, Anda harus menghapus pipeline dan sumber daya yang digunakannya sehingga Anda tidak akan dikenakan biaya untuk terus menggunakan sumber daya tersebut. Jika Anda tidak berniat untuk terus menggunakan CodePipeline, hapus pipeline, lalu CodeDeploy aplikasi dan instans Amazon EC2 yang terkait, dan terakhir, ember Amazon S3 digunakan untuk menyimpan artefak. Anda juga harus mempertimbangkan apakah akan menghapus sumber daya lain, seperti GitHub repositori, jika Anda tidak berniat untuk terus menggunakannya.

Untuk membersihkan sumber daya yang digunakan dalam tutorial ini

1. Buka sesi terminal di Linux, macOS, atau mesin Unix lokal Anda, atau prompt perintah di mesin Windows lokal Anda, dan jalankan delete-pipeline perintah untuk menghapus pipeline yang Anda buat. Untuk**`MySecondPipeline`**, Anda akan memasukkan perintah berikut:

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

   Perintah ini tidak mengembalikan apa pun.
2. Untuk membersihkan CodeDeploy sumber daya Anda, ikuti petunjuk di Cleaning Up.
3. Untuk membersihkan sumber daya instans Anda, hapus instans EC2 tempat Anda menginstal Jenkins. Untuk informasi lebih lanjut, lihat Membersihkan instans Anda.
4. Jika Anda tidak bermaksud membuat lebih banyak saluran pipa atau menggunakan CodePipeline lagi, hapus bucket Amazon S3 yang digunakan untuk menyimpan artefak untuk pipeline Anda. Untuk menghapus ember, ikuti petunjuk di Menghapus ember.

5. Jika Anda tidak berniat menggunakan sumber daya lain untuk pipeline ini lagi, pertimbangkan untuk menghapusnya dengan mengikuti panduan untuk sumber daya tertentu. Misalnya, jika Anda ingin menghapus GitHub repositori, ikuti instruksi di Menghapus repositori di situs web. GitHub

Tutorial: Menyiapkan aturan CloudWatch Acara untuk menerima pemberitahuan email untuk perubahan status pipeline

Setelah menyiapkan pipeline AWS CodePipeline, Anda dapat menyiapkan aturan CloudWatch Acara untuk mengirim notifikasi setiap kali ada perubahan pada status eksekusi pipeline Anda, atau dalam tahapan atau tindakan di pipeline Anda. Untuk informasi selengkapnya tentang penggunaan CloudWatch Acara untuk mengatur notifikasi perubahan status pipeline, lihatMemantau CodePipeline peristiwa.

Dalam tutorial ini, Anda mengonfigurasi notifikasi untuk mengirim email saat status pipeline berubah menjadi GAGAL. Tutorial ini menggunakan metode transformator masukan saat membuat aturan CloudWatch Events. Ini mengubah rincian skema pesan untuk menyampaikan pesan dalam teks yang dapat dibaca manusia.

> **Note**
>
> Saat Anda membuat sumber daya untuk tutorial ini, seperti notifikasi Amazon SNS dan aturan CloudWatch Acara, pastikan sumber daya dibuat di AWS Wilayah yang sama dengan pipeline Anda.

Topik

- Langkah 1: Siapkan notifikasi email menggunakan Amazon SNS
- Langkah 2: Buat aturan dan tambahkan topik SNS sebagai target
- Langkah 3: Bersihkan Sumber Daya

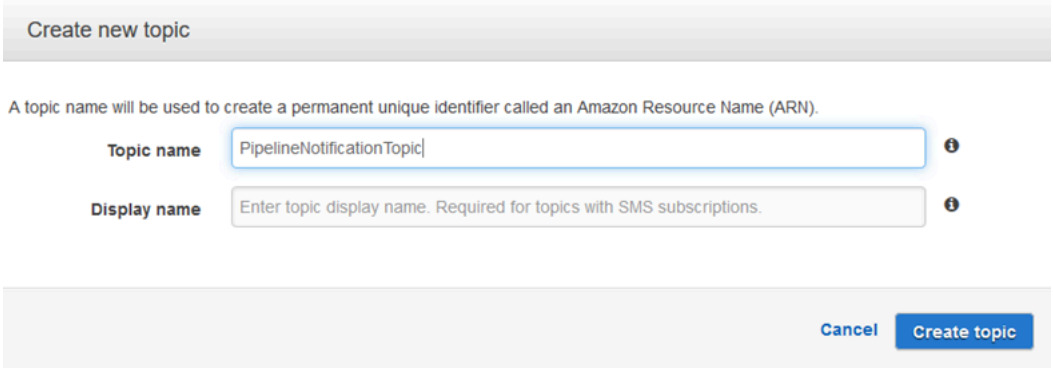
Langkah 1: Siapkan notifikasi email menggunakan Amazon SNS

Amazon SNS mengoordinasikan penggunaan topik untuk mengirimkan pesan ke titik akhir atau klien berlangganan. Gunakan Amazon SNS untuk membuat topik notifikasi dan kemudian berlangganan

topik menggunakan alamat email Anda. Topik Amazon SNS akan ditambahkan sebagai target ke aturan CloudWatch Acara Anda. Untuk informasi lebih lanjut, lihat Panduan Developer Layanan Notifikasi Sederhana Amazon.

Buat atau identifikasi topik di Amazon SNS. CodePipeline akan menggunakan CloudWatch Acara untuk mengirim pemberitahuan ke topik ini melalui Amazon SNS. Untuk membuat topik:

1. Buka konsol Amazon SNS di https://console.aws.amazon.com /sns.
2. Pilih Buat topik.
3. Dalam kotak dialog Buat topik baru, untuk nama Topik, ketikkan nama untuk topik (misalnya,**`PipelineNotificationTopic`**).



4. Pilih Buat topik.

   Untuk informasi selengkapnya, lihat Membuat Topik di Panduan Pengembang Amazon SNS.

Berlangganan satu atau beberapa penerima ke topik untuk menerima pemberitahuan email. Untuk berlangganan penerima ke suatu topik:

1. Di konsol Amazon SNS, dari daftar Topik, pilih kotak centang di sebelah topik baru Anda. Pilih Tindakan, Berlangganan topik.
2. Di kotak dialog Buat langganan, verifikasi bahwa ARN muncul di Topik ARN.
3. Untuk Protokol, pilih Email.
4. Untuk Endpoint, ketik alamat email lengkap penerima.
5. Pilih Buat Langganan.
6. Amazon SNS mengirimkan email konfirmasi berlangganan ke penerima. Untuk menerima pemberitahuan email, penerima harus memilih tautan Konfirmasi langganan di email ini. Setelah penerima mengklik tautan, jika berhasil berlangganan, Amazon SNS menampilkan pesan konfirmasi di browser web penerima.

Untuk informasi selengkapnya, lihat Berlangganan Topik di Panduan Pengembang Amazon SNS.

Langkah 2: Buat aturan dan tambahkan topik SNS sebagai target

Buat aturan pemberitahuan CloudWatch Acara dengan CodePipeline sebagai sumber acara.

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi, pilih Peristiwa.
3. Pilih Buat aturan. Di bawah Sumber acara, pilih AWS CodePipeline. Untuk Jenis Peristiwa, pilih Perubahan Status Eksekusi Pipeline.
4. Pilih Status spesifik, dan pilih**`FAILED`**.
5. Pilih Edit untuk membuka editor JSON untuk panel Pratinjau Pola Acara. Tambahkan **`pipeline`** parameter dengan nama pipeline Anda seperti yang ditunjukkan pada contoh berikut untuk pipeline bernama “MyPipeline.”

   Anda dapat menyalin pola acara di sini dan menempelkannya ke konsol:

   ```
   {
     "source": [
       "aws.codepipeline"
     ],
     "detail-type": [
       "CodePipeline Pipeline Execution State Change"
     ],
     "detail": {
       "state": [
         "FAILED"
       ],
       "pipeline": [
         "myPipeline"
       ]
     }
   }
   ```

6. Untuk Targets (Target), pilih Add target (Tambahkan target).
7. Dalam daftar target, pilih topik SNS. Untuk Topik, masukkan topik yang Anda buat.
8. Perluas Konfigurasi input, lalu pilih Input Transformer.

9. Dalam kotak Input Path, ketik pasangan kunci-nilai berikut.

```
{ "pipeline" : "$.detail.pipeline" }
```

   Dalam kotak Template Input, ketik berikut ini:

```
"The Pipeline <pipeline> has failed."
```

10. Pilih Konfigurasikan detail.
11. Pada halaman Konfigurasi detail aturan, ketikkan nama dan deskripsi opsional. Untuk Status, biarkan kotak Diaktifkan dipilih.
12. Pilih Buat aturan.
13. Konfirmasikan bahwa CodePipeline sekarang mengirim pemberitahuan build. Misalnya, periksa untuk melihat apakah email pemberitahuan build sekarang ada di kotak masuk Anda.
14. Untuk mengubah perilaku aturan, di CloudWatch konsol, pilih aturan, lalu pilih Tindakan, Edit. Edit aturan, pilih Konfigurasikan detail, lalu pilih Perbarui aturan.

    Untuk berhenti menggunakan aturan untuk mengirim pemberitahuan build, di CloudWatch konsol, pilih aturan, lalu pilih Tindakan, Nonaktifkan.

    Untuk menghapus aturan, di CloudWatch konsol, pilih aturan, lalu pilih Tindakan, Hapus.

Langkah 3: Bersihkan Sumber Daya

Setelah Anda menyelesaikan tutorial ini, Anda harus menghapus pipeline dan sumber daya yang digunakannya sehingga Anda tidak akan dikenakan biaya untuk terus menggunakan sumber daya tersebut.

Untuk informasi tentang cara membersihkan notifikasi SNS dan menghapus aturan Amazon CloudWatch Events, lihat Membersihkan (Berhenti berlangganan dari Topik Amazon SNS) dan referensi `DeleteRule` di Referensi API Acara CloudWatch Amazon.

Tutorial: Membuat pipeline yang membangun dan menguji aplikasi Android Anda AWS Device Farm

Anda dapat menggunakannya AWS CodePipeline untuk mengonfigurasi alur integrasi berkelanjutan di mana aplikasi Anda dibuat dan diuji setiap kali komit didorong. Tutorial ini menunjukkan cara

membuat dan mengonfigurasi pipeline untuk membangun dan menguji aplikasi Android Anda dengan kode sumber di GitHub repositori. Pipeline mendeteksi kedatangan GitHub komit baru dan kemudian digunakan CodeBuilduntuk membangun aplikasi dan Device Farm untuk mengujinya.

> **Important**
>
> Sebagai bagian dari pembuatan pipa di konsol, ember artefak S3 akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

> **Important**
>
> Banyak tindakan yang Anda tambahkan ke pipeline dalam prosedur ini melibatkan AWS sumber daya yang perlu Anda buat sebelum membuat pipeline. AWS sumber daya untuk tindakan sumber Anda harus selalu dibuat di AWS Wilayah yang sama tempat Anda membuat pipeline. Misalnya, jika Anda membuat pipeline di Wilayah AS Timur (Ohio), CodeCommit repositori Anda harus berada di Wilayah AS Timur (Ohio).
> Anda dapat menambahkan tindakan lintas wilayah saat membuat pipeline. AWS sumber daya untuk tindakan lintas wilayah harus berada di AWS Wilayah yang sama di mana Anda berencana untuk menjalankan tindakan. Untuk informasi selengkapnya, lihat Menambahkan tindakan Lintas wilayah di CodePipeline.

Anda dapat mencobanya menggunakan aplikasi Android dan definisi pengujian yang ada, atau Anda dapat menggunakan aplikasi sampel dan definisi pengujian yang disediakan oleh Device Farm.

Sebelum Anda memulai

1. Masuk ke AWS Device Farm konsol dan pilih Buat proyek baru.
2. Pilih proyek Anda. Di browser, salin URL proyek baru Anda. URL berisi ID proyek.
3. Salin dan simpan ID proyek ini. Anda menggunakannya saat Anda membuat pipeline di CodePipeline.

   Berikut adalah contoh URL untuk sebuah proyek. Untuk mengekstrak ID proyek, salin nilainya setelahnya`projects/`. Dalam contoh ini, ID proyek adalah`eec4905f-98f8-40aa-9afc-4c1cfexample`.

```
https://<region-URL>/devicefarm/home?region=us-west-2#/projects/
eec4905f-98f8-40aa-9afc-4c1cfexample/runs
```

Konfigurasikan CodePipeline untuk menggunakan pengujian Device Farm

1. Tambahkan dan komit file yang dipanggil `buildspec.yml` di root kode aplikasi Anda, dan dorong ke repositori Anda. CodeBuild menggunakan file ini untuk menjalankan perintah dan mengakses artefak yang diperlukan untuk membangun aplikasi Anda.

   ```
   version: 0.2

   phases:
     build:
       commands:
         - chmod +x ./gradlew
         - ./gradlew assembleDebug
   artifacts:
     files:
        - './android/app/build/outputs/**/*.apk'
     discard-paths: yes
   ```

2. (Opsional) Jika Anda menggunakan Calabash atau Appium untuk menguji aplikasi Anda, tambahkan file definisi pengujian ke repositori Anda. Pada langkah selanjutnya, Anda dapat mengonfigurasi Device Farm untuk menggunakan definisi untuk menjalankan rangkaian pengujian Anda.

   Jika Anda menggunakan pengujian bawaan Device Farm, Anda dapat melewati langkah ini.

3. Untuk membuat pipeline dan menambahkan tahap sumber, lakukan hal berikut:

   a. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

   b. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.

   c. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.

d. Pada Langkah 2: Pilih halaman pengaturan pipeline, dalam nama Pipeline, masukkan nama untuk pipeline Anda.

e. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.

f. Dalam peran Layanan, biarkan peran layanan baru dipilih, dan biarkan nama Peran tidak berubah. Anda juga dapat memilih untuk menggunakan peran layanan yang ada, jika Anda memilikinya.

> **Note**
>
> Jika Anda menggunakan peran CodePipeline layanan yang dibuat sebelum Juli 2018, Anda perlu menambahkan izin untuk Device Farm. Untuk melakukannya, buka konsol IAM, temukan peran, lalu tambahkan izin berikut ke kebijakan peran. Untuk informasi selengkapnya, lihat Menambahkan izin ke peran CodePipeline layanan.
>
> ```
> {
>      "Effect": "Allow",
>      "Action": [
>         "devicefarm:ListProjects",
>         "devicefarm:ListDevicePools",
>         "devicefarm:GetRun",
>         "devicefarm:GetUpload",
>         "devicefarm:CreateUpload",
>         "devicefarm:ScheduleRun"
>      ],
>      "Resource": "*"
> }
> ```

g. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.

h. Pada Langkah 3: Tambahkan halaman tahap sumber, di penyedia Sumber, pilih GitHub (melalui GitHub Aplikasi).

i. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.

j. Di Repositori, pilih repositori sumber.

k. Di Branch, pilih cabang yang ingin Anda gunakan.

l. Biarkan default yang tersisa untuk tindakan sumber. Pilih Berikutnya.

4. Pada Langkah 4: Tambahkan tahap build, tambahkan tahap build:

   a. Di Penyedia build, pilih Penyedia build lain, lalu pilih AWS CodeBuild. Izinkan Wilayah ke default ke Wilayah alur.

   b. Pilih Buat proyek.

   c. Di Nama proyek, masukkan nama untuk proyek pembangunan ini.

   d. Di Citra lingkungan, pilih Citra terkelola. Untuk Sistem operasi, pilih Ubuntu.

   e. Untuk Waktu aktif, pilih Standar. Untuk Gambar, pilih:5.0aws/codebuild/standard.

      CodeBuild menggunakan image OS ini, yang telah menginstal Android Studio, untuk membangun aplikasi Anda.

   f. Untuk peran Layanan, pilih peran CodeBuild layanan yang ada atau buat yang baru.

   g. Untuk spesifikasi Build, pilih Gunakan file buildspec.

   h. Pilih Lanjutkan ke CodePipeline. Ini kembali ke CodePipeline konsol dan membuat CodeBuild proyek yang menggunakan `buildspec.yml` di repositori Anda untuk konfigurasi. Proyek build menggunakan peran layanan untuk mengelola Layanan AWS izin. Langkah ini mungkin memakan waktu beberapa menit.

   i. Pilih Berikutnya.

5. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

   Pilih Berikutnya.

6. Pada Langkah 6: Tambahkan halaman tahap penerapan, pilih Lewati tahap penerapan, lalu terima pesan peringatan dengan memilih Lewati lagi. Pilih Berikutnya.

7. Pada Langkah 7: Tinjau, pilih Buat pipeline. Anda akan melihat diagram yang menunjukkan sumber dan tahap build.

8. Tambahkan tindakan pengujian Device Farm ke pipeline Anda:

   a. Di kanan atas, pilih Edit.

   b. Di bagian bawah diagram, pilih + Tambahkan tahap. Dalam nama Panggung, masukkan nama, seperti**`Test`**.

   c. Pilih + Tambahkan grup tindakan.

   d. Di Nama tindakan, masukkan nama.

e. Di penyedia Action, pilih AWS Device Farm. Izinkan Wilayah ke default ke Wilayah alur.

f. Dalam artefak Input, pilih artefak input yang cocok dengan artefak keluaran dari tahap yang datang sebelum tahap pengujian, seperti. `BuildArtifact`

   Di AWS CodePipeline konsol, Anda dapat menemukan nama artefak keluaran untuk setiap tahap dengan mengarahkan kursor ke ikon informasi di diagram pipa. Jika pipeline menguji aplikasi langsung dari tahap Sumber, pilih SourceArtifact. Jika pipeline menyertakan tahap Build, pilih BuildArtifact.

g. Masuk ProjectId, masukkan ID proyek Device Farm Anda. Gunakan langkah-langkah di awal tutorial ini untuk mengambil ID proyek Anda.

h. Masuk DevicePoolArn, masukkan ARN untuk kumpulan perangkat. Untuk mendapatkan kumpulan perangkat yang tersedia ARNs untuk proyek, termasuk ARN untuk Perangkat Teratas, gunakan AWS CLI untuk memasukkan perintah berikut:

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

i. Masuk AppType, masukkan Android.

   Berikut ini adalah daftar nilai yang valid untuk AppType:

   - iOS
   - Android
   - Web

j. Di App, masukkan jalur paket aplikasi yang dikompilasi. Jalur relatif terhadap akar artefak input untuk tahap pengujian. Biasanya, jalur ini mirip dengan`app-release.apk`.

k. Masuk TestType, masukkan jenis pengujian Anda, lalu di Uji, masukkan jalur file definisi pengujian. Jalur relatif terhadap akar artefak input untuk pengujian Anda.

   Berikut ini adalah daftar nilai yang valid untuk TestType:

   - APPIUM_JAVA_JUNIT
   - APPIUM_JAVA_TESTNG
   - APPIUM_NODE
   - APPIUM_RUBY
   - APPIUM_PYTHON

- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- BUILTIN_FUZZ
- INSTRUMENTASI
- XCTEST
- XCTEST_UI

> **Note**
>
> Node lingkungan khusus tidak didukung.

l. Di bidang yang tersisa, berikan konfigurasi yang sesuai untuk pengujian dan jenis aplikasi Anda.

m. (Opsional) Di Advanced, berikan informasi konfigurasi untuk uji coba Anda.

n. Pilih Simpan.

o. Di panggung yang Anda edit, pilih Selesai. Di AWS CodePipeline panel, pilih Simpan, lalu pilih Simpan pada pesan peringatan.

p. Untuk mengirimkan perubahan dan memulai pembuatan pipeline, pilih Rilis perubahan, lalu pilih Rilis.

Tutorial: Buat pipeline yang menguji aplikasi iOS Anda AWS Device Farm

Anda dapat menggunakannya AWS CodePipeline untuk dengan mudah mengonfigurasi alur integrasi berkelanjutan di mana aplikasi Anda diuji setiap kali bucket sumber berubah. Tutorial ini menunjukkan cara membuat dan mengonfigurasi pipeline untuk menguji aplikasi iOS bawaan Anda dari bucket S3. Pipeline mendeteksi kedatangan perubahan yang disimpan melalui Amazon CloudWatch Events, dan kemudian menggunakan Device Farm untuk menguji aplikasi yang dibuat.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

> **Important**
>
> Banyak tindakan yang Anda tambahkan ke pipeline dalam prosedur ini melibatkan AWS sumber daya yang perlu Anda buat sebelum membuat pipeline. AWS sumber daya untuk tindakan sumber Anda harus selalu dibuat di AWS Wilayah yang sama tempat Anda membuat pipeline. Misalnya, jika Anda membuat pipeline di Wilayah AS Timur (Ohio), CodeCommit repositori Anda harus berada di Wilayah AS Timur (Ohio).
> Anda dapat menambahkan tindakan lintas wilayah saat membuat pipeline. AWS sumber daya untuk tindakan lintas wilayah harus berada di AWS Wilayah yang sama di mana Anda berencana untuk menjalankan tindakan. Untuk informasi selengkapnya, lihat Menambahkan tindakan Lintas wilayah di CodePipeline.

Anda dapat mencobanya menggunakan aplikasi iOS yang ada, atau Anda dapat menggunakan contoh aplikasi iOS.

Sebelum Anda memulai

1. Masuk ke AWS Device Farm konsol dan pilih Buat proyek baru.
2. Pilih proyek Anda. Di browser, salin URL proyek baru Anda. URL berisi ID proyek.
3. Salin dan simpan ID proyek ini. Anda menggunakannya saat Anda membuat pipeline di CodePipeline.

   Berikut adalah contoh URL untuk sebuah proyek. Untuk mengekstrak ID proyek, salin nilainya setelahnya`projects/`. Dalam contoh ini, ID proyek adalah`eec4905f-98f8-40aa-9afc-4c1cfexample`.

```
https://<region-URL>/devicefarm/home?region=us-west-2#/projects/
eec4905f-98f8-40aa-9afc-4c1cfexample/runs
```

Konfigurasikan CodePipeline untuk menggunakan pengujian Device Farm Anda (contoh Amazon S3)

1. Buat atau gunakan bucket S3 dengan versi diaktifkan. Ikuti instruksi Langkah 1: Buat bucket sumber S3 untuk aplikasi Anda untuk membuat ember S3.
2. Di konsol Amazon S3 untuk bucket Anda, pilih Unggah, dan ikuti petunjuk untuk mengunggah file.zip Anda.

   Contoh aplikasi Anda harus dikemas dalam file.zip.
3. Untuk membuat pipeline dan menambahkan tahap sumber, lakukan hal berikut:

   a. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

   b. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.

   c. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.

   d. Pada Langkah 2: Pilih halaman pengaturan pipeline, dalam nama Pipeline, masukkan nama untuk pipeline Anda.

   e. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.

   f. Dalam peran Layanan, biarkan peran layanan baru dipilih, dan biarkan nama Peran tidak berubah. Anda juga dapat memilih untuk menggunakan peran layanan yang ada, jika Anda memilikinya.

   > **Note**
   >
   > Jika Anda menggunakan peran CodePipeline layanan yang dibuat sebelum Juli 2018, Anda harus menambahkan izin untuk Device Farm. Untuk melakukannya, buka konsol IAM, temukan peran, lalu tambahkan izin berikut ke kebijakan peran.

Untuk informasi selengkapnya, lihat Menambahkan izin ke peran CodePipeline layanan.

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

g. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.

h. Pada Langkah 3: Tambahkan halaman tahap sumber, di penyedia Sumber, pilih Amazon S3.

i. Di lokasi Amazon S3, masukkan bucket, seperti, dan kunci objek`my-storage-bucket`, seperti `s3-ios-test-1.zip` untuk file.zip Anda.

j. Pilih Berikutnya.

4. Pada Langkah 4: Tambahkan tahap build, buat tahap build placeholder untuk pipeline Anda. Ini memungkinkan Anda untuk membuat pipeline di wizard. Setelah Anda menggunakan wizard untuk membuat pipeline dua tahap, Anda tidak lagi memerlukan tahap pembuatan placeholder ini. Setelah pipeline selesai, tahap kedua ini dihapus dan tahap pengujian baru ditambahkan pada langkah 5.

   a. Di penyedia Build, pilih Add Jenkins. Pilihan build ini adalah placeholder. Atribut ini tidak digunakan.

   b. Di nama Penyedia, masukkan nama. Namanya adalah placeholder. Atribut ini tidak digunakan.

   c. Di URL Server, masukkan teks. Teks adalah placeholder. Atribut ini tidak digunakan.

   d. Dalam nama Proyek, masukkan nama. Namanya adalah placeholder. Atribut ini tidak digunakan.

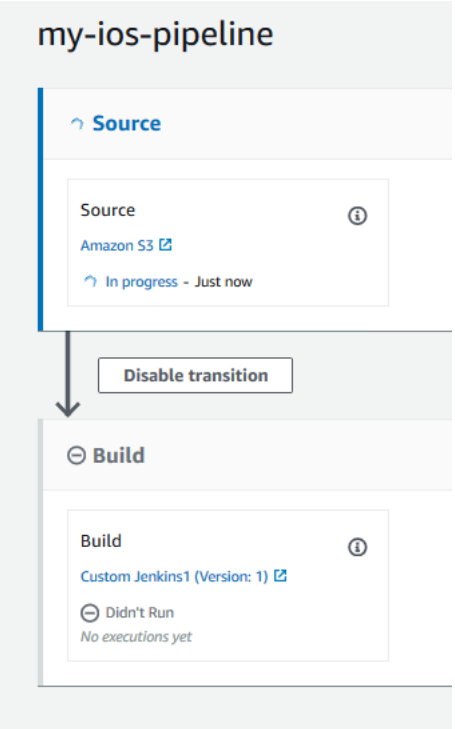
   e. Pilih Berikutnya.

f. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

   Pilih Berikutnya.

g. Pada Langkah 6: Tambahkan halaman tahap penerapan, pilih Lewati tahap penerapan, lalu terima pesan peringatan dengan memilih Lewati lagi.

h. Pada Langkah 7: Tinjau, pilih Buat pipeline. Anda akan melihat diagram yang menunjukkan sumber dan tahap build.



5. Tambahkan tindakan pengujian Device Farm ke pipeline Anda sebagai berikut:

   a. Di kanan atas, pilih Edit.

   b. Pilih Edit tahap. Pilih Hapus. Ini menghapus tahap placeholder sekarang karena Anda tidak lagi membutuhkannya untuk pembuatan pipeline.

   c. Di bagian bawah diagram, pilih + Tambahkan tahap.

   d. Dalam nama Stage, masukkan nama untuk panggung, seperti Test, dan kemudian pilih Add stage.

   e. Pilih + Tambahkan grup tindakan.

   f. Dalam nama Action, masukkan nama, seperti DeviceFarmTest.

   g. Di penyedia Action, pilih AWS Device Farm. Izinkan Wilayah ke default ke Wilayah alur.

h. Dalam artefak Input, pilih artefak input yang cocok dengan artefak keluaran dari tahap yang datang sebelum tahap pengujian, seperti. `SourceArtifact`

   Di AWS CodePipeline konsol, Anda dapat menemukan nama artefak keluaran untuk setiap tahap dengan mengarahkan kursor ke ikon informasi di diagram pipa. Jika pipeline menguji aplikasi langsung dari tahap Sumber, pilih SourceArtifact. Jika pipeline menyertakan tahap Build, pilih BuildArtifact.

i. Di ProjectId, pilih ID proyek Device Farm Anda. Gunakan langkah-langkah di awal tutorial ini untuk mengambil ID proyek Anda.

j. Masuk DevicePoolArn, masukkan ARN untuk kumpulan perangkat. Untuk mendapatkan kumpulan perangkat yang tersedia ARNs untuk proyek, termasuk ARN untuk Perangkat Teratas, gunakan AWS CLI untuk memasukkan perintah berikut:

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

k. Masuk AppType, masukkan iOS.

   Berikut ini adalah daftar nilai yang valid untuk AppType:

   - iOS
   - Android
   - Web

l. Di App, masukkan jalur paket aplikasi yang dikompilasi. Jalur relatif terhadap akar artefak input untuk tahap pengujian. Biasanya, jalur ini mirip dengan`ios-test.ipa`.

m. Masuk TestType, masukkan jenis pengujian Anda, lalu di Uji, masukkan jalur file definisi pengujian. Jalur relatif terhadap akar artefak input untuk pengujian Anda.

   Jika Anda menggunakan salah satu pengujian Device Farm bawaan, masukkan jenis pengujian yang dikonfigurasi dalam proyek Device Farm Anda, seperti BUILTIN_FUZZ. Masuk FuzzEventCount, masukkan waktu dalam milidetik, seperti 6000. Masuk FuzzEventThrottle, masukkan waktu dalam milidetik, seperti 50.

   Jika Anda tidak menggunakan salah satu pengujian Device Farm bawaan, masukkan jenis pengujian, lalu di Test, masukkan path file definisi pengujian. Jalur relatif terhadap akar artefak input untuk pengujian Anda.

   Berikut ini adalah daftar nilai yang valid untuk TestType:

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG
- APPIUM_NODE
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- BUILTIN_FUZZ
- INSTRUMENTASI
- XCTEST
- XCTEST_UI

> **Note**
>
> Node lingkungan khusus tidak didukung.

n. Di bidang yang tersisa, berikan konfigurasi yang sesuai untuk pengujian dan jenis aplikasi Anda.

o. (Opsional) Di Advanced, berikan informasi konfigurasi untuk uji coba Anda.

p. Pilih Simpan.

q. Di panggung yang Anda edit, pilih Selesai. Di AWS CodePipeline panel, pilih Simpan, lalu pilih Simpan pada pesan peringatan.

r. Untuk mengirimkan perubahan dan memulai eksekusi pipeline, pilih Rilis perubahan, lalu pilih Rilis.

Tutorial: Buat pipeline yang di-deploy ke Service Catalog

Service Catalog memungkinkan Anda untuk membuat dan menyediakan produk berdasarkan AWS CloudFormation template.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Tutorial ini menunjukkan cara membuat dan mengonfigurasi pipeline untuk menyebarkan template produk Anda ke Service Catalog dan memberikan perubahan yang telah Anda buat di repositori sumber Anda (sudah dibuat di GitHub, CodeCommit, atau Amazon S3).

> **Note**
>
> Jika Amazon S3 adalah penyedia sumber untuk pipeline Anda, Anda harus mengunggah ke bucket semua file sumber yang dikemas sebagai satu file.zip. Jika tidak, tindakan sumber gagal.

Pertama, Anda membuat produk di Service Catalog, dan kemudian Anda membuat pipeline AWS CodePipeline. Tutorial ini menyediakan dua opsi untuk mengatur konfigurasi penyebaran:

- Buat produk di Service Catalog dan unggah file template ke repositori sumber Anda. Menyediakan versi produk dan konfigurasi penerapan di CodePipeline konsol (tanpa file konfigurasi terpisah). Lihat Opsi 1: Terapkan ke Service Catalog tanpa file konfigurasi.

  > **Note**
  >
  > File template dapat dibuat dalam format YAMAL atau JSON.

- Buat produk di Service Catalog dan unggah file template ke repositori sumber Anda. Menyediakan versi produk dan konfigurasi penyebaran dalam file konfigurasi terpisah. Lihat Opsi 2: Terapkan ke Service Catalog menggunakan file konfigurasi.

Opsi 1: Terapkan ke Service Catalog tanpa file konfigurasi

Dalam contoh ini, Anda mengunggah file AWS CloudFormation template sampel untuk bucket S3, lalu membuat produk Anda di Service Catalog. Selanjutnya, Anda membuat pipeline dan menentukan konfigurasi penerapan di CodePipeline konsol.

Langkah 1: Unggah file templat sampel ke repositori sumber

1. Buka editor teks. Buat template sampel dengan menempelkan berikut ini ke dalam file. Simpan file sebagai `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"
    }
  }
}
```

   Template ini memungkinkan AWS CloudFormation untuk membuat bucket S3 yang dapat digunakan oleh Service Catalog.

2. Unggah `S3_template.json` file ke AWS CodeCommit repositori Anda.

Langkah 2: Buat produk di Service Catalog

1. Sebagai administrator TI, masuk ke konsol Service Catalog, buka halaman Produk, lalu pilih Unggah produk baru.
2. Pada halaman Unggah produk baru, lengkapi yang berikut ini:
   a. Di Nama Produk, masukkan nama yang ingin Anda gunakan untuk produk baru Anda.
   b. Dalam Deskripsi, masukkan deskripsi katalog produk. Deskripsi ini ditampilkan dalam daftar produk untuk membantu pengguna dalam memilih produk yang benar.
   c. Di Disediakan oleh, masukkan nama departemen atau administrator TI Anda.
   d. Pilih Berikutnya.
3. (Opsional) Di Masukkan detail dukungan, masukkan informasi kontak untuk dukungan produk, dan pilih Berikutnya.
4. Dalam detail Versi, lengkapi yang berikut ini:
   a. Pilih Mengunggah file template. Jelajahi `S3_template.json` file Anda dan unggah.
   b. Dalam judul Versi, masukkan nama versi produk (misalnya,**`devops S3 v2`**).
   c. Dalam Deskripsi, masukkan detail yang membedakan versi ini dari versi lain.
   d. Pilih Berikutnya.
5. Pada halaman Tinjauan, verifikasi bahwa informasinya benar, lalu pilih Buat.
6. Pada halaman Produk, di browser, salin URL produk baru Anda. Ini berisi ID produk. Salin dan simpan ID produk ini. Anda menggunakannya saat Anda membuat pipeline di CodePipeline.

   Berikut adalah URL untuk produk yang diberi nama`my-product`. Untuk mengekstrak ID produk, salin nilai antara tanda sama dengan (=) dan ampersand (). & Dalam contoh ini, ID produk adalah`prod-example123456`.

   ```
   https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
   productCreated=prod-example123456&createdProductTitle=my-product
   ```

   > **Note**
   >
   > Salin URL untuk produk Anda sebelum Anda menjauh dari halaman. Setelah Anda menavigasi jauh dari halaman ini, Anda harus menggunakan CLI untuk mendapatkan ID produk Anda.

Setelah beberapa detik, produk Anda muncul di halaman Produk. Anda mungkin perlu menyegarkan browser Anda untuk melihat produk dalam daftar.

Langkah 3: Buat pipeline Anda

1. Untuk memberi nama pipeline Anda dan memilih parameter untuk pipeline Anda, lakukan hal berikut:

   a. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

   b. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.

   c. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.

   d. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan nama untuk pipeline Anda.

   e. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.

   f. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.

   g. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.

2. Untuk menambahkan tahap sumber pada Langkah 3: Tambahkan halaman tahap sumber, lakukan hal berikut:

   a. Di penyedia Sumber, pilih AWS CodeCommit.

   b. Dalam nama Repositori dan nama Cabang, masukkan repositori dan cabang yang ingin Anda gunakan untuk tindakan sumber Anda.

   c. Pilih Berikutnya.

3. Pada Langkah 4: Tambahkan tahap build, pilih Lewati tahap build, lalu terima pesan peringatan dengan memilih Lewati lagi.

4. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

Pilih Berikutnya.

5. Pada Langkah 6: Tambahkan tahap penerapan, selesaikan yang berikut ini:

   a. Di Penyedia Deploy, pilih AWS Service Catalog.

   b. Untuk konfigurasi deployment, pilih Enter deployment configuration.

   c. Di Product ID, tempel ID produk yang Anda salin dari konsol Service Catalog.

   d. Di jalur file Template, masukkan jalur relatif tempat file template disimpan.

   e. Di Jenis produk, pilih CloudFormation template.

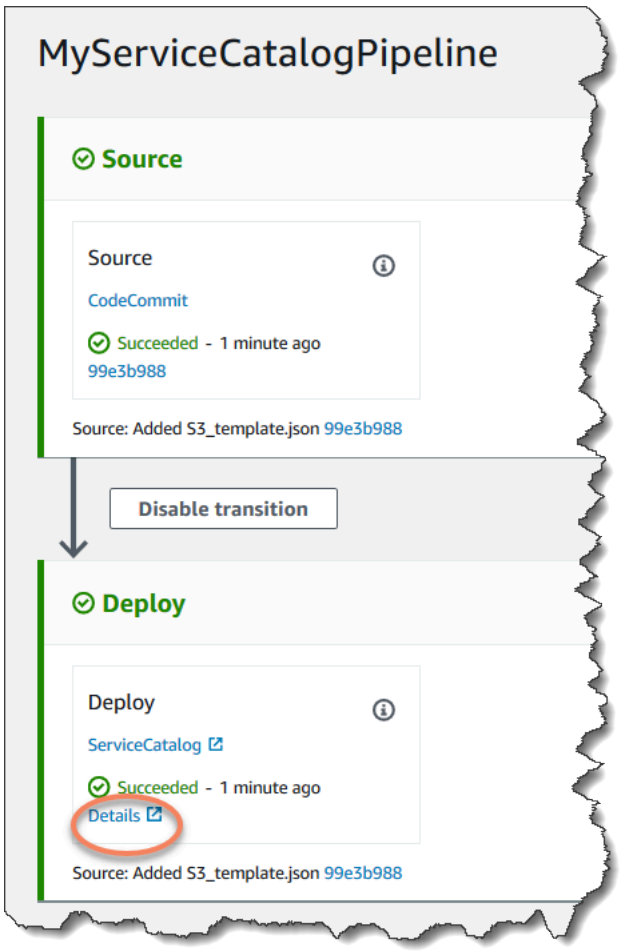
   f. Dalam nama versi Produk, masukkan nama versi produk yang Anda tentukan di Service Catalog. Jika Anda ingin perubahan template disebarkan ke versi produk baru, masukkan nama versi produk yang belum digunakan untuk versi produk sebelumnya dalam produk yang sama.

   g. Untuk artefak Input, pilih artefak input sumber.

   h. Pilih Berikutnya.

6. Pada Langkah 7: Tinjau, tinjau pengaturan pipeline Anda, lalu pilih Buat.

7. Setelah pipeline Anda berhasil berjalan, pada tahap penerapan, pilih Detail. Ini membuka produk Anda di Service Catalog.



8. Di bawah informasi produk Anda, pilih nama versi Anda untuk membuka templat produk. Lihat penerapan template.

Langkah 4: Dorong perubahan dan verifikasi produk Anda di Service Catalog

1. Lihat pipeline Anda di CodePipeline konsol, dan pada tahap sumber Anda, pilih Detail. AWS CodeCommit Repositori sumber Anda terbuka di konsol. Pilih Edit, dan buat perubahan pada file (misalnya, ke deskripsi).

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

2. Berkomitmen dan dorong perubahan Anda. Pipeline Anda dimulai setelah Anda mendorong perubahan. Saat proses pipeline selesai, pada tahap deployment, pilih Detail untuk membuka produk Anda di Service Catalog.
3. Di bawah informasi produk Anda, pilih nama versi baru untuk membuka template produk. Lihat perubahan template yang diterapkan.

Opsi 2: Terapkan ke Service Catalog menggunakan file konfigurasi

Dalam contoh ini, Anda mengunggah file AWS CloudFormation template sampel untuk bucket S3, lalu membuat produk Anda di Service Catalog. Anda juga mengunggah file konfigurasi terpisah yang menentukan konfigurasi penerapan Anda. Selanjutnya, Anda membuat pipeline dan menentukan lokasi file konfigurasi Anda.

Langkah 1: Unggah file templat sampel ke repositori sumber

1. Buka editor teks. Buat template sampel dengan menempelkan berikut ini ke dalam file. Simpan file sebagai `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"
    }
  }
}
```

   Template ini memungkinkan AWS CloudFormation untuk membuat bucket S3 yang dapat digunakan oleh Service Catalog.

2. Unggah `S3_template.json` file ke AWS CodeCommit repositori Anda.

Langkah 2: Buat file konfigurasi penyebaran produk Anda

1. Buka editor teks. Buat file konfigurasi untuk produk Anda. File konfigurasi digunakan untuk menentukan parameter/preferensi penyebaran Service Catalog Anda. Anda menggunakan file ini saat membuat pipeline.

   Sampel ini menyediakan “devops S3 v2" dan a `ProductVersionName` of. `ProductVersionDescription` `MyProductVersionDescription` Jika Anda ingin perubahan template disebarkan ke versi produk baru, cukup masukkan nama versi produk yang belum digunakan untuk versi produk sebelumnya dalam produk yang sama.

   Simpan file sebagai `sample_config.json`.

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

   File ini membuat informasi versi produk untuk Anda setiap kali pipeline Anda berjalan.

2. Unggah `sample_config.json` file ke AWS CodeCommit repositori Anda. Pastikan Anda mengunggah file ini ke repositori sumber Anda.

Langkah 3: Buat produk di Service Catalog

1. Sebagai administrator TI, masuk ke konsol Service Catalog, buka halaman Produk, lalu pilih Unggah produk baru.
2. Pada halaman Unggah produk baru, lengkapi yang berikut ini:

   a. Di Nama Produk, masukkan nama yang ingin Anda gunakan untuk produk baru Anda.

   b. Dalam Deskripsi, masukkan deskripsi katalog produk. Deskripsi ini muncul di daftar produk untuk membantu pengguna memilih produk yang benar.

   c. Di Disediakan oleh, masukkan nama departemen atau administrator TI Anda.

   d. Pilih Berikutnya.

3. (Opsional) Di Masukkan detail dukungan, masukkan informasi kontak dukungan produk, lalu pilih Berikutnya.
4. Dalam detail Versi, lengkapi yang berikut ini:
   a. Pilih Mengunggah file template. Jelajahi `S3_template.json` file Anda dan unggah.
   b. Dalam judul Versi, masukkan nama versi produk (misalnya, “devops S3 v2").
   c. Dalam Deskripsi, masukkan detail yang membedakan versi ini dari versi lain.
   d. Pilih Berikutnya.
5. Pada halaman Tinjauan, verifikasi bahwa informasi sudah benar, lalu pilih Konfirmasi dan unggah.
6. Pada halaman Produk, di browser, salin URL produk baru Anda. Ini berisi ID produk. Salin dan simpan ID produk ini. Anda gunakan saat membuat pipeline di CodePipeline.

   Berikut adalah URL untuk produk yang diberi nama`my-product`. Untuk mengekstrak ID produk, salin nilai antara tanda sama dengan (=) dan ampersand (). & Dalam contoh ini, ID produk adalah`prod-example123456`.

   ```
   https://<region-URL>/servicecatalog/home?region=<region>#/admin-products?
   productCreated=prod-example123456&createdProductTitle=my-product
   ```

   > **Note**
   >
   > Salin URL untuk produk Anda sebelum Anda menjauh dari halaman. Setelah Anda menavigasi jauh dari halaman ini, Anda harus menggunakan CLI untuk mendapatkan ID produk Anda.

   Setelah beberapa detik, produk Anda muncul di halaman Produk. Anda mungkin perlu menyegarkan browser Anda untuk melihat produk dalam daftar.

Langkah 4: Buat pipeline Anda

1. Untuk memberi nama pipeline Anda dan memilih parameter untuk pipeline Anda, lakukan hal berikut:

a. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

b. Pilih Memulai. Pilih Buat pipeline, lalu masukkan nama untuk pipeline Anda.

c. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.

d. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.

2. Untuk menambahkan tahap sumber, lakukan hal berikut:

a. Di penyedia Sumber, pilih AWS CodeCommit.

b. Dalam nama Repositori dan nama Cabang, masukkan repositori dan cabang yang ingin Anda gunakan untuk tindakan sumber Anda.

c. Pilih Berikutnya.

3. Di Add build stage, pilih Lewati tahap build, lalu terima pesan peringatan dengan memilih Lewati lagi.

4. Di tahap Add deploy, selesaikan yang berikut ini:

a. Di Penyedia Deploy, pilih AWS Service Catalog.

b. Pilih Gunakan file konfigurasi.

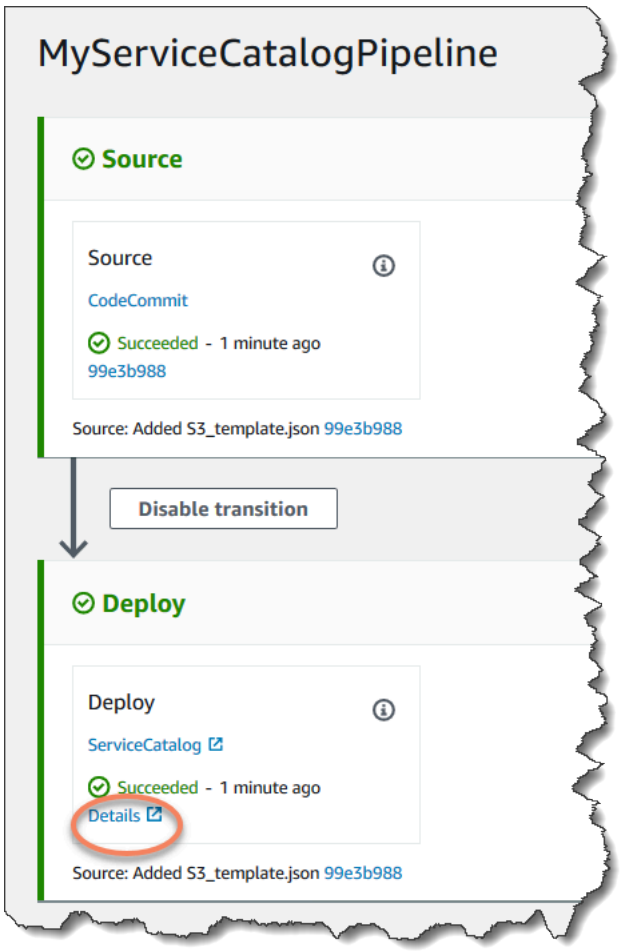
c. Di Product ID, tempel ID produk yang Anda salin dari konsol Service Catalog.

d. Di jalur file Konfigurasi, masukkan jalur file dari file konfigurasi di repositori Anda.

e. Pilih Berikutnya.

5. Di Tinjau, tinjau setelan pipeline Anda, lalu pilih Buat.

6. Setelah pipeline berhasil berjalan, pada tahap penerapan, pilih Detail untuk membuka produk di Service Catalog.



7. Di bawah informasi produk Anda, pilih nama versi Anda untuk membuka templat produk. Lihat penerapan template.

Langkah 5: Dorong perubahan dan verifikasi produk Anda di Service Catalog

1. Lihat pipeline Anda di CodePipeline konsol, dan pada tahap sumber, pilih Detail. AWS CodeCommit Repositori sumber Anda terbuka di konsol. Pilih Edit, lalu buat perubahan pada file (misalnya, ke deskripsi).

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

2. Berkomitmen dan dorong perubahan Anda. Pipeline Anda dimulai setelah Anda mendorong perubahan. Saat proses pipeline selesai, pada tahap deployment, pilih Detail untuk membuka produk Anda di Service Catalog.
3. Di bawah informasi produk Anda, pilih nama versi baru untuk membuka template produk. Lihat perubahan template yang diterapkan.

Tutorial: Buat pipeline dengan AWS CloudFormation

Contohnya menyediakan contoh templat yang memungkinkan Anda gunakan AWS CloudFormation untuk membuat pipeline yang menyebarkan aplikasi ke instance Anda setiap kali kode sumber berubah. Template sampel membuat pipeline yang dapat Anda lihat AWS CodePipeline. Pipeline mendeteksi kedatangan perubahan yang disimpan melalui Amazon CloudWatch Events.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Topik

- Contoh 1: Buat AWS CodeCommit pipeline dengan AWS CloudFormation
- Contoh 2: Buat pipeline Amazon S3 dengan AWS CloudFormation

Contoh 1: Buat AWS CodeCommit pipeline dengan AWS CloudFormation

Panduan ini menunjukkan cara menggunakan AWS CloudFormation konsol untuk membuat infrastruktur yang menyertakan pipeline yang terhubung ke repositori CodeCommit sumber. Dalam tutorial ini, Anda menggunakan file template sampel yang disediakan untuk membuat tumpukan sumber daya, yang mencakup penyimpanan artefak, pipeline, dan sumber daya deteksi perubahan, seperti aturan Amazon CloudWatch Events Anda. Setelah membuat tumpukan sumber daya AWS CloudFormation, Anda dapat melihat pipeline di AWS CodePipeline konsol. Pipa adalah pipa dua tahap dengan tahap CodeCommit sumber dan tahap CodeDeploy penyebaran.

Prasyarat:

Anda harus telah membuat sumber daya berikut untuk digunakan dengan template AWS CloudFormation sampel:

- Anda harus telah membuat repositori sumber. Anda dapat menggunakan AWS CodeCommit repositori yang Anda buat. Tutorial: Buat pipeline sederhana (CodeCommit repositori)

- Anda harus telah membuat grup CodeDeploy aplikasi dan penyebaran. Anda dapat menggunakan CodeDeploy sumber daya yang Anda buatTutorial: Buat pipeline sederhana (CodeCommit repositori).
- Pilih salah satu tautan ini untuk mengunduh file AWS CloudFormation templat sampel untuk membuat pipeline: YAMAL | JSON

  Buka zip file dan letakkan di komputer lokal Anda.
- Unduh file aplikasi sampel SampleApp_Linux.zip.

Buat pipeline Anda di AWS CloudFormation

1. Buka zip file dari SampleApp_Linux.zip dan unggah file ke AWS CodeCommit repositori Anda. Anda harus mengunggah file yang tidak di-zip ke direktori root repositori Anda. Anda dapat mengikuti instruksi Langkah 2: Tambahkan kode sampel ke CodeCommit repositori Anda untuk mendorong file ke repositori Anda.
2. Buka AWS CloudFormation konsol dan pilih Create Stack. Pilih Dengan sumber daya baru (standar).
3. Di bawah Tentukan templat, pilih Unggah templat. Pilih Pilih file dan kemudian pilih file template dari komputer lokal Anda. Pilih Berikutnya.
4. Dalam nama Stack, masukkan nama untuk pipeline Anda. Parameter yang ditentukan oleh template sampel ditampilkan. Masukkan parameter berikut:

   a. Masuk ApplicationName, masukkan nama CodeDeploy aplikasi Anda.

   b. Masuk BetaFleet, masukkan nama grup CodeDeploy penyebaran Anda.

   c. Masuk BranchName, masukkan cabang repositori yang ingin Anda gunakan.

   d. Masuk RepositoryName, masukkan nama repositori CodeCommit sumber Anda.
5. Pilih Berikutnya. Terima default pada halaman berikut, lalu pilih Berikutnya.
6. Di Capabilities, pilih I accept yang AWS CloudFormation mungkin membuat resource IAM, lalu pilih Create stack.
7. Setelah pembuatan tumpukan Anda selesai, lihat daftar acara untuk memeriksa kesalahan apa pun.

   Pemecahan Masalah

Pengguna IAM yang membuat pipeline AWS CloudFormation mungkin memerlukan izin tambahan untuk membuat sumber daya untuk pipeline. Izin berikut diperlukan dalam kebijakan AWS CloudFormation untuk memungkinkan pembuatan sumber daya CloudWatch Acara Amazon yang diperlukan untuk CodeCommit pipeline:

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

8. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

   Di bawah Pipelines, pilih pipeline Anda dan pilih View. Diagram menunjukkan sumber pipa dan tahapan penerapan Anda.

   > **Note**
   >
   > Untuk melihat pipeline yang telah dibuat, cari kolom Logical ID di bawah tab Resources untuk tumpukan Anda CloudFormation. Perhatikan nama di kolom Physical ID untuk pipeline. Di CodePipeline, Anda dapat melihat pipeline dengan ID Fisik (nama pipeline) yang sama di Wilayah tempat Anda membuat tumpukan.

9. Di repositori sumber Anda, komit dan dorong perubahan. Sumber daya deteksi perubahan Anda mengambil perubahan, dan pipeline Anda dimulai.

Contoh 2: Buat pipeline Amazon S3 dengan AWS CloudFormation

Panduan ini menunjukkan cara menggunakan AWS CloudFormation konsol untuk membuat infrastruktur yang menyertakan pipeline yang terhubung ke bucket sumber Amazon S3. Dalam tutorial ini, Anda menggunakan file template sampel yang disediakan untuk membuat tumpukan

sumber daya, yang mencakup bucket sumber, penyimpanan artefak, pipeline, dan sumber daya deteksi perubahan, seperti aturan dan jejak CloudWatch Acara Amazon Anda. CloudTrail Setelah membuat tumpukan sumber daya AWS CloudFormation, Anda dapat melihat pipeline di AWS CodePipeline konsol. Pipeline adalah pipa dua tahap dengan tahap sumber Amazon S3 dan tahap penerapan CodeDeploy.

Prasyarat:

Anda harus memiliki sumber daya berikut untuk digunakan dengan template AWS CloudFormation sampel:

- Anda harus telah membuat instans Amazon EC2, tempat Anda menginstal CodeDeploy agen pada instans. Anda harus telah membuat grup CodeDeploy aplikasi dan penyebaran. Gunakan Amazon EC2 dan CodeDeploy sumber daya yang Anda buat. Tutorial: Buat pipeline sederhana (CodeCommit repositori)
- Pilih tautan berikut untuk mengunduh file AWS CloudFormation templat sampel untuk membuat pipeline dengan sumber Amazon S3:
  - Unduh contoh template untuk pipeline Anda: YAMAL | JSON
  - Unduh template sampel untuk CloudTrail bucket dan trail Anda: YAML| JSON
  - Buka zip file dan letakkan di komputer lokal Anda.
- Unduh aplikasi sampel dari SampleApp_Linux.zip.

  Simpan file.zip di komputer lokal Anda. Anda mengunggah file.zip setelah tumpukan dibuat.

Buat pipeline Anda di AWS CloudFormation

1. Buka AWS CloudFormation konsol, dan pilih Create Stack. Pilih Dengan sumber daya baru (standar).
2. Di Pilih templat, pilih Unggah templat. Pilih Pilih file, lalu pilih file template dari komputer lokal Anda. Pilih Berikutnya.
3. Dalam nama Stack, masukkan nama untuk pipeline Anda. Parameter yang ditentukan oleh template sampel ditampilkan. Masukkan parameter berikut:

   a. Masuk ApplicationName, masukkan nama CodeDeploy aplikasi Anda. Anda dapat mengganti nama `DemoApplication` default.

   b. Masuk BetaFleet, masukkan nama grup CodeDeploy penyebaran Anda. Anda dapat mengganti nama `DemoFleet` default.

c. Masuk SourceObjectKey, masuk`SampleApp_Linux.zip`. Anda mengunggah file ini ke bucket setelah template membuat bucket dan pipeline.

4. Pilih Berikutnya. Terima default pada halaman berikut, lalu pilih Berikutnya.
5. Di Capabilities, pilih I accept yang AWS CloudFormation mungkin membuat resource IAM, lalu pilih Create stack.
6. Setelah pembuatan tumpukan Anda selesai, lihat daftar acara untuk memeriksa kesalahan apa pun.

   Pemecahan Masalah

   Pengguna IAM yang membuat pipeline AWS CloudFormation mungkin memerlukan izin tambahan untuk membuat sumber daya untuk pipeline. Izin berikut diperlukan dalam kebijakan AWS CloudFormation untuk memungkinkan membuat sumber daya CloudWatch Acara Amazon yang diperlukan untuk pipeline Amazon S3:

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

7. Di CloudFormation, di tab Sumber Daya untuk tumpukan Anda, lihat sumber daya yang dibuat untuk tumpukan Anda.

   > **Note**
   >
   > Untuk melihat pipeline yang telah dibuat, cari kolom Logical ID di bawah tab Resources untuk tumpukan Anda CloudFormation. Perhatikan nama di kolom Physical ID untuk pipeline. Di CodePipeline, Anda dapat melihat pipeline dengan ID Fisik (nama pipeline) yang sama di Wilayah tempat Anda membuat tumpukan.

Pilih bucket S3 dengan `sourcebucket` label di namanya, seperti `s3-cfn-codepipeline-sourcebucket-y04EXAMPLE.` Jangan pilih bucket artefak pipa.

Bucket sumber kosong karena sumber daya baru dibuat oleh CloudFormation. Buka konsol Amazon S3 dan temukan bucket Anda`sourcebucket`. Pilih Unggah, dan ikuti petunjuk untuk mengunggah `SampleApp_Linux.zip` file.zip Anda.

> **Note**
>
> Jika Amazon S3 adalah penyedia sumber untuk pipeline Anda, Anda harus mengunggah ke bucket semua file sumber yang dikemas sebagai satu file.zip. Jika tidak, tindakan sumber gagal.

8. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

   Di bawah Pipelines, pilih pipeline Anda, lalu pilih View. Diagram menunjukkan sumber pipa dan tahapan penerapan Anda.

9. Selesaikan langkah-langkah dalam prosedur berikut untuk membuat AWS CloudTrail sumber daya Anda.

Buat AWS CloudTrail sumber daya Anda di AWS CloudFormation

1. Buka AWS CloudFormation konsol, dan pilih Create Stack.
2. Di Pilih templat, pilih Unggah templat ke Amazon S3. Pilih Browse, lalu pilih file template untuk AWS CloudTrail sumber daya dari komputer lokal Anda. Pilih Berikutnya.
3. Dalam nama Stack, masukkan nama untuk tumpukan sumber daya Anda. Parameter yang ditentukan oleh template sampel ditampilkan. Masukkan parameter berikut:

   - Dalam SourceObjectKey, terima default untuk file zip aplikasi sampel.

4. Pilih Berikutnya. Terima default pada halaman berikut, lalu pilih Berikutnya.
5. Di Capabilities, pilih I accept yang AWS CloudFormation mungkin membuat resource IAM, lalu pilih Create.
6. Setelah pembuatan tumpukan Anda selesai, lihat daftar acara untuk memeriksa kesalahan apa pun.

Izin berikut diperlukan dalam kebijakan AWS CloudFormation untuk memungkinkan pembuatan CloudTrail sumber daya yang diperlukan untuk pipeline Amazon S3:

```
{
     "Effect": "Allow",
     "Action": [
        "cloudtrail:CreateTrail",
        "cloudtrail:DeleteTrail",
        "cloudtrail:StartLogging",
        "cloudtrail:StopLogging",
        "cloudtrail:PutEventSelectors"
     ],
     "Resource": "resource_ARN"
}
```

7. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

   Di bawah Pipelines, pilih pipeline Anda, lalu pilih View. Diagram menunjukkan sumber pipa dan tahapan penerapan Anda.

8. Di ember sumber Anda, komit dan dorong perubahan. Sumber daya deteksi perubahan Anda mengambil perubahan dan pipeline Anda dimulai.

Tutorial: Buat pipeline yang menggunakan variabel dari AWS CloudFormation tindakan penerapan

Dalam tutorial ini, Anda menggunakan AWS CodePipeline konsol untuk membuat pipeline dengan tindakan penerapan. Saat pipeline berjalan, template membuat tumpukan dan juga membuat `outputs` file. Output yang dihasilkan oleh template tumpukan adalah variabel yang dihasilkan oleh AWS CloudFormation tindakan di CodePipeline.

Dalam tindakan di mana Anda membuat tumpukan dari template, Anda menunjuk namespace variabel. Variabel yang dihasilkan oleh `outputs` file kemudian dapat dikonsumsi oleh tindakan selanjutnya. Dalam contoh ini, Anda membuat set perubahan berdasarkan `StackName` variabel yang dihasilkan oleh AWS CloudFormation tindakan. Setelah persetujuan manual, Anda menjalankan set perubahan dan kemudian membuat tindakan menghapus tumpukan yang menghapus tumpukan berdasarkan `StackName` variabel.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Topik

- Prasyarat: Buat peran AWS CloudFormation layanan dan repositori CodeCommit
- Langkah 1: Unduh, edit, dan unggah AWS CloudFormation templat sampel
- Langkah 2: Buat alur Anda
- Langkah 3: Tambahkan tindakan CloudFormation penerapan untuk membuat set perubahan
- Langkah 4: Tambahkan tindakan persetujuan manual
- Langkah 5: Tambahkan tindakan CloudFormation penerapan untuk menjalankan set perubahan
- Langkah 6: Tambahkan tindakan CloudFormation penerapan untuk menghapus tumpukan

Prasyarat: Buat peran AWS CloudFormation layanan dan repositori CodeCommit

Anda harus sudah memiliki yang berikut:

- Sebuah CodeCommit repositori. Anda dapat menggunakan AWS CodeCommit repositori yang Anda buat. Tutorial: Buat pipeline sederhana (CodeCommit repositori)
- Contoh ini membuat tumpukan Amazon DocumentDB dari template. Anda harus menggunakan AWS Identity and Access Management (IAM) untuk membuat peran AWS CloudFormation layanan dengan izin berikut untuk Amazon DocumentDB.

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

Langkah 1: Unduh, edit, dan unggah AWS CloudFormation templat sampel

Unduh file AWS CloudFormation template sampel dan unggah ke CodeCommit repositori Anda.

1. Arahkan ke template sampel untuk Wilayah Anda. Misalnya, gunakan tabel di https://docs.aws.amazon.com/documentdb/latest/developerguide/quick_start_cfn.html#quick_start_cfn-launch_stack untuk memilih Wilayah dan unduh templat. Unduh template untuk Cluster Amazon DocumentDB. Nama filenya adalah`documentdb_full_stack.yaml`.
2. Buka zip `documentdb_full_stack.yaml` file, dan buka di editor teks. Buat perubahan berikut.

   a. Untuk contoh ini, tambahkan `Purpose:` parameter berikut ke `Parameters` bagian Anda di template.

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

   b. Untuk contoh ini, tambahkan `StackName` output berikut ke `Outputs:` bagian Anda di template.

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

3. Unggah file template ke AWS CodeCommit repositori Anda. Anda harus mengunggah file template yang tidak di-zip dan diedit ke direktori root repositori Anda.

   Untuk menggunakan CodeCommit konsol untuk mengunggah file Anda:

   a. Buka CodeCommit konsol, dan pilih repositori Anda dari daftar Repositori.

   b. Pilih Tambahkan file, lalu pilih Unggah file.

   c. Pilih Pilih file, lalu telusuri file Anda. Lakukan perubahan dengan memasukkan nama pengguna dan alamat email Anda. Pilih Perubahan commit.

   File Anda akan terlihat seperti ini di tingkat root di repositori Anda:

```
documentdb_full_stack.yaml
```

Langkah 2: Buat alur Anda

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan CodeCommit tindakan di mana artefak sumber adalah file template Anda.
- Tahap penyebaran dengan tindakan CloudFormation penerapan.

Setiap tindakan dalam tahap sumber dan penerapan yang dibuat oleh wizard diberi namespace variabel, `SourceVariables` dan`DeployVariables`, masing-masing. Karena tindakan memiliki namespace yang ditetapkan, variabel yang dikonfigurasi dalam contoh ini tersedia untuk tindakan hilir. Untuk informasi selengkapnya, lihat Referensi variabel.

Untuk membuat alur dengan wizard

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyCFNDeployPipeline`**.
5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
6. Dalam peran Layanan, lakukan salah satu hal berikut:
   - Pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.
   - Pilih Peran layanan yang ada. Di Nama peran, pilih peran layanan Anda dari daftar.
7. Di toko Artifact:
   a. Pilih Lokasi default untuk menggunakan penyimpanan artefak default, seperti bucket artefak Amazon S3 yang ditetapkan sebagai default, untuk pipeline di Wilayah yang dipilih untuk pipeline.

b. Pilih Lokasi khusus jika Anda sudah memiliki toko artefak, seperti bucket artefak Amazon S3, di Wilayah yang sama dengan pipeline Anda.

> **Note**
>
> Ini bukan bucket sumber untuk kode sumber Anda. Ini adalah penyimpanan artifact untuk alur Anda. Penyimpanan artifact terpisah, seperti bucket S3, diperlukan untuk setiap alur. Saat membuat atau mengedit pipeline, Anda harus memiliki bucket artefak di Wilayah pipeline dan satu bucket artefak per AWS Wilayah tempat Anda menjalankan tindakan. Untuk informasi selengkapnya, lihat Artefak input dan output dan CodePipeline referensi struktur pipa.

Pilih Berikutnya.

8. Pada Langkah 3: Tambahkan tahap sumber:

   a. Di penyedia Sumber, pilih AWS CodeCommit.

   b. Dalam nama Repositori, pilih nama CodeCommit repositori yang Anda buat. Langkah 1: Buat CodeCommit repositori

   c. Di Nama cabang, pilih nama cabang yang berisi pembaruan kode terbaru Anda.

   Setelah Anda memilih nama repositori dan cabang, aturan Amazon CloudWatch Events yang akan dibuat untuk pipeline ini akan ditampilkan.

   Pilih Berikutnya.

9. Pada Langkah 4: Tambahkan tahap build, pilih Lewati tahap build, lalu terima pesan peringatan dengan memilih Lewati lagi.

   Pilih Berikutnya.

10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

11. Pada Langkah 6: Tambahkan tahap penerapan:

    a. Di Nama tindakan, pilih Deploy. Di Penyedia Deploy, pilih CloudFormation.

b. Dalam mode Tindakan, pilih Buat atau perbarui tumpukan.

c. Dalam nama Stack, masukkan nama untuk tumpukan. Ini adalah nama tumpukan yang akan dibuat template.

d. Dalam nama file Output, masukkan nama untuk file output, seperti**`outputs`**. Ini adalah nama file yang akan dibuat oleh tindakan setelah tumpukan dibuat.

e. Perluas Lanjutan. Di bawah penggantian Parameter, masukkan penggantian template Anda sebagai pasangan nilai kunci. Misalnya, template ini memerlukan penggantian berikut.

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

Jika Anda tidak memasukkan penggantian, template akan membuat tumpukan dengan nilai default.

f. Pilih Berikutnya.

g. Pada Langkah 7: Tinjau, pilih Buat pipeline. Anda akan melihat diagram yang menunjukkan tahapan pipa. Biarkan pipeline Anda berjalan. Pipa dua tahap Anda selesai dan siap untuk tahapan tambahan yang akan ditambahkan.

Langkah 3: Tambahkan tindakan CloudFormation penerapan untuk membuat set perubahan

Buat tindakan selanjutnya di pipeline Anda yang CloudFormation memungkinkan Anda membuat set perubahan sebelum tindakan persetujuan manual.

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

   Di bawah Pipelines, pilih pipeline Anda dan pilih View. Diagram menunjukkan sumber pipa dan tahapan penerapan Anda.

2. Pilih untuk mengedit pipeline, atau terus menampilkan pipeline dalam mode Edit.

3. Pilih untuk mengedit tahap Deploy.

4. Tambahkan tindakan penerapan yang akan membuat set perubahan untuk tumpukan yang dibuat dalam tindakan sebelumnya. Anda menambahkan tindakan ini setelah tindakan yang ada di panggung.

   a. Dalam nama Tindakan, masukkan Change_Set. Di penyedia Action, pilih AWS CloudFormation .

   b. Di artefak Input, pilih SourceArtifact.

   c. Dalam mode Tindakan, pilih Buat atau ganti set perubahan.

   d. Dalam nama Stack, masukkan sintaks variabel seperti yang ditunjukkan. Ini adalah nama tumpukan tempat set perubahan dibuat untuk, di mana namespace default `DeployVariables` ditetapkan untuk tindakan.

   ```
   #{DeployVariables.StackName}
   ```

   e. Di Ubah nama set, masukkan nama set perubahan.

   ```
   my-changeset
   ```

   f. Di Parameter Overrides, ubah `Purpose` parameter dari `testing` ke. `production`

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

   g. Pilih Selesai untuk menyimpan tindakan.

Langkah 4: Tambahkan tindakan persetujuan manual

Buat tindakan persetujuan manual di pipeline Anda.

1. Pilih untuk mengedit pipeline, atau terus menampilkan pipeline dalam mode Edit.
2. Pilih untuk mengedit tahap Deploy.

3. Tambahkan tindakan persetujuan manual setelah tindakan penerapan yang membuat set perubahan. Tindakan ini memungkinkan Anda memverifikasi perubahan sumber daya yang dibuat CloudFormation sebelum pipeline mengeksekusi set perubahan.

Langkah 5: Tambahkan tindakan CloudFormation penerapan untuk menjalankan set perubahan

Buat tindakan berikutnya di pipeline Anda yang memungkinkan CloudFormation untuk menjalankan set perubahan setelah tindakan persetujuan manual.

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

   Di bawah Pipelines, pilih pipeline Anda dan pilih View. Diagram menunjukkan sumber pipa dan tahapan penerapan Anda.
2. Pilih untuk mengedit pipeline, atau terus menampilkan pipeline dalam mode Edit.
3. Pilih untuk mengedit tahap Deploy.
4. Tambahkan tindakan penerapan yang akan menjalankan set perubahan yang disetujui dalam tindakan manual sebelumnya:
   a. Dalam nama Action, masukkan Execute_Change_Set. Di penyedia Action, pilih AWS CloudFormation.
   b. Di artefak Input, pilih SourceArtifact.
   c. Dalam mode Tindakan, pilih Jalankan set perubahan.
   d. Dalam nama Stack, masukkan sintaks variabel seperti yang ditunjukkan. Ini adalah nama tumpukan tempat set perubahan dibuat untuk.

      ```
      #{DeployVariables.StackName}
      ```

   e. Di Ubah nama set, masukkan nama set perubahan yang Anda buat dalam tindakan sebelumnya.

      ```
      my-changeset
      ```

   f. Pilih Selesai untuk menyimpan tindakan.
   g. Lanjutkan proses pipa.

Langkah 6: Tambahkan tindakan CloudFormation penerapan untuk menghapus tumpukan

Buat tindakan akhir di pipeline Anda yang memungkinkan CloudFormation untuk mendapatkan nama tumpukan dari variabel dalam file output dan menghapus tumpukan.

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

   Di bawah Pipelines, pilih pipeline Anda dan pilih View. Diagram menunjukkan sumber pipa dan tahapan penerapan Anda.
2. Pilih untuk mengedit pipeline.
3. Pilih untuk mengedit tahap Deploy.
4. Tambahkan tindakan penerapan yang akan menghapus tumpukan:

   a. Dalam nama Action, pilih DeleteStack. Di Penyedia Deploy, pilih CloudFormation.

   b. Dalam mode Tindakan, pilih Hapus tumpukan.

   c. Dalam nama Stack, masukkan sintaks variabel seperti yang ditunjukkan. Ini adalah nama tumpukan yang akan dihapus oleh tindakan.

   d. Pilih Selesai untuk menyimpan tindakan.

   e. Pilih Simpan untuk menyimpan pipa.

   Pipa berjalan saat disimpan.

Tutorial: Penerapan Standar Amazon ECS dengan CodePipeline

Tutorial ini membantu Anda membuat pipeline deployment (CD) yang lengkap dan end-to-end berkelanjutan dengan Amazon ECS. CodePipeline

> **Important**
>
> Sebagai bagian dari pembuatan pipa di konsol, ember artefak S3 akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

> **Note**
>
> Tutorial ini untuk tindakan penerapan standar Amazon ECS untuk. CodePipeline Untuk tutorial yang menggunakan Amazon ECS untuk tindakan penerapan CodeDeploy biru/hijau, lihat. CodePipeline Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR

> **Note**
>
> Tutorial ini untuk tindakan penerapan standar Amazon ECS untuk CodePipeline dengan aksi sumber. Untuk tutorial yang menggunakan tindakan ECSstandard penyebaran Amazon bersama dengan tindakan ECRBuild AndPublish build CodePipeline untuk mendorong gambar Anda, lihatTutorial: Buat dan dorong gambar Docker ke Amazon ECR dengan CodePipeline (tipe V2).

Prasyarat

Ada beberapa sumber daya yang harus Anda miliki sebelum Anda dapat menggunakan tutorial ini untuk membuat pipeline CD Anda. Berikut adalah hal-hal yang Anda butuhkan untuk memulai:

> **Note**
>
> Semua sumber daya ini harus dibuat dalam AWS Wilayah yang sama.

- Sebuah repositori kontrol sumber (tutorial ini menggunakan CodeCommit) dengan Dockerfile dan sumber aplikasi Anda. Untuk informasi selengkapnya, lihat Membuat CodeCommit Repositori di AWS CodeCommit Panduan Pengguna.
- Sebuah repositori gambar Docker (tutorial ini menggunakan Amazon ECR) yang berisi gambar yang telah Anda buat dari Dockerfile dan sumber aplikasi Anda. Untuk informasi selengkapnya, lihat Membuat Repositori dan Mendorong Gambar di Panduan Pengguna Amazon Elastic Container Registry.
- Definisi tugas Amazon ECS yang mereferensikan gambar Docker yang dihosting di repositori gambar Anda. Untuk informasi selengkapnya, lihat Membuat Definisi Tugas di Panduan Pengembang Layanan Amazon Elastic Container.

> **Important**
>
> Tindakan penerapan standar Amazon ECS untuk CodePipeline membuat revisi definisi tugas sendiri berdasarkan revisi yang digunakan oleh layanan Amazon ECS. Jika Anda membuat revisi baru untuk definisi tugas tanpa memperbarui layanan Amazon ECS, tindakan penerapan akan mengabaikan revisi tersebut.

Di bawah ini adalah contoh definisi tugas yang digunakan untuk tutorial ini. Nilai yang Anda gunakan untuk `name` dan `family` akan digunakan pada langkah berikutnya untuk file spesifikasi build Anda.

```
{
  "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": []
}
```

- Cluster Amazon ECS yang menjalankan layanan yang menggunakan definisi tugas yang Anda sebutkan sebelumnya. Untuk informasi selengkapnya, lihat Membuat Cluster dan Membuat Layanan di Panduan Pengembang Layanan Amazon Elastic Container.

Setelah Anda memenuhi prasyarat ini, Anda dapat melanjutkan dengan tutorial dan membuat pipeline CD Anda.

Langkah 1: Tambahkan File Spesifikasi Build ke Repositori Sumber Anda

Tutorial ini digunakan CodeBuild untuk membangun gambar Docker Anda dan mendorong gambar ke Amazon ECR. Tambahkan `buildspec.yml` file ke repositori kode sumber Anda untuk memberi tahu CodeBuild cara melakukannya. Contoh spesifikasi build di bawah ini melakukan hal berikut:

- Tahap pra-bangun:
  - Masuk ke Amazon ECR.
  - Atur URI repositori ke gambar ECR Anda dan tambahkan tag gambar dengan tujuh karakter pertama dari ID komit Git dari sumber.
- Membangun panggung:
  - Buat gambar Docker dan beri tag gambar baik sebagai `latest` maupun dengan ID komit Git.
- Tahap pasca-pembangunan:
  - Dorong gambar ke repositori ECR Anda dengan kedua tag.
  - Tulis file yang dipanggil `imagedefinitions.json` di root build yang memiliki nama penampung layanan Amazon ECS Anda serta gambar dan tag. Tahap penyebaran pipeline CD Anda menggunakan informasi ini untuk membuat revisi baru definisi tugas layanan Anda, dan kemudian memperbarui layanan untuk menggunakan definisi tugas baru. `imagedefinitions.json`File ini diperlukan untuk pekerja kerja ECS.

Rekatkan teks contoh ini untuk membuat `buildspec.yml` file Anda, dan ganti nilai untuk definisi gambar dan tugas Anda. Teks ini menggunakan contoh ID akun 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
```

Spesifikasi build ditulis untuk definisi tugas sampel yang disediakanPrasyarat, digunakan oleh layanan Amazon ECS untuk tutorial ini. `REPOSITORY_URI`Nilai sesuai dengan `image` repositori (tanpa tag gambar apa pun), dan *`hello-world`* nilai di dekat akhir file sesuai dengan nama wadah dalam definisi tugas layanan.

Untuk menambahkan **`buildspec.yml`** file ke repositori sumber Anda

1. Buka editor teks lalu salin dan tempel spesifikasi build di atas ke file baru.

2. Ganti `REPOSITORY_URI` value (*`012345678910.dkr.ecr.us-west-2.amazonaws.com/hello-world`*) dengan URI repositori Amazon ECR Anda (tanpa tag gambar apa pun) untuk image Docker Anda. Ganti *`hello-world`* dengan nama kontainer dalam definisi tugas layanan Anda yang mereferensikan gambar Docker Anda.
3. Komit dan dorong `buildspec.yml` file Anda ke repositori sumber Anda.

   a. Tambahkan file.

   ```
   git add .
   ```

   b. Lakukan perubahan.

   ```
   git commit -m "Adding build specification."
   ```

   c. Dorong komit.

   ```
   git push
   ```

Langkah 2: Membuat Pipeline Deployment Berkelanjutan

Gunakan CodePipeline wizard untuk membuat tahapan pipeline Anda dan menghubungkan repositori sumber Anda ke layanan ECS Anda.

Untuk membuat alur Anda

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pada halaman Selamat Datang, pilih Buat pipeline.

   Jika ini adalah pertama kalinya Anda menggunakan CodePipeline, halaman pengantar muncul alih-alih Selamat Datang. Pilih Mulai Sekarang.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, ketikkan nama untuk pipeline Anda. Untuk tutorial ini, nama pipeline adalah hello-world.
5. Dalam tipe Pipeline, pertahankan pilihan default di V2. Jenis pipa berbeda dalam karakteristik dan harga. Untuk informasi selengkapnya, lihat Jenis pipa. Pilih Berikutnya.

6. Pada Langkah 3: Tambahkan halaman tahap sumber, untuk penyedia Sumber, pilih AWS CodeCommit.

   a. Untuk nama Repositori, pilih nama CodeCommit repositori yang akan digunakan sebagai lokasi sumber untuk pipeline Anda.

   b. Untuk nama Branch, pilih cabang yang akan digunakan dan pilih Next.

7. Pada Langkah 4: Tambahkan halaman tahap build, untuk penyedia Build pilih AWS CodeBuild, lalu pilih Create project.

   a. Untuk nama Project, pilih nama unik untuk proyek build Anda. Untuk tutorial ini, nama proyeknya adalah hello-world.

   b. Untuk gambar Lingkungan, pilih Gambar terkelola.

   c. Untuk sistem operasi, pilih Amazon Linux 2.

   d. Untuk Runtime, pilih Standar.

   e. Untuk Gambar, pilih **`aws/codebuild/amazonlinux2-x86_64-standard:3.0`**.

   f. Untuk versi Gambar dan jenis Lingkungan, gunakan nilai default.

   g. Pilih Aktifkan bendera ini jika Anda ingin membuat gambar Docker atau ingin build Anda mendapatkan hak istimewa yang lebih tinggi.

   h. Batalkan pilihan CloudWatch log. Anda mungkin perlu memperluas Advanced.

   i. Pilih Lanjutkan ke CodePipeline.

   j. Pilih Berikutnya.

   > **Note**
   >
   > Wizard membuat peran CodeBuild layanan untuk proyek build Anda, yang disebut codebuild- ***`build-project-name`*** -service-role. Perhatikan nama peran ini, saat Anda menambahkan izin Amazon ECR ke sana nanti.

8. Pada Langkah 5: Tambahkan halaman tahap penerapan, untuk penyedia Deployment, pilih Amazon ECS.

   a. Untuk nama Cluster, pilih klaster Amazon ECS tempat layanan Anda berjalan. Untuk tutorial ini, cluster adalah default.

   b. Untuk nama Layanan, pilih layanan yang akan diperbarui dan pilih Berikutnya. Untuk tutorial ini, nama layanannya adalah hello-world.

9. Pada halaman Langkah 6: Tinjau, tinjau konfigurasi pipeline Anda dan pilih Buat pipeline untuk membuat pipeline.

> **Note**
>
> Sekarang pipa telah dibuat, ia mencoba untuk menjalankan melalui tahapan pipa yang berbeda. Namun, CodeBuild peran default yang dibuat oleh wizard tidak memiliki izin untuk menjalankan semua perintah yang terdapat dalam `buildspec.yml` file, sehingga tahap build gagal. Bagian selanjutnya menambahkan izin untuk tahap build.

Langkah 3: Tambahkan Izin Amazon ECR ke Peran CodeBuild

CodePipeline Wizard membuat peran IAM untuk proyek CodeBuild build, yang disebut codebuild-***`build-project-name`*** -service-role. Untuk tutorial ini, namanya adalah codebuild-hello-world-service-role. Karena `buildspec.yml` file melakukan panggilan ke operasi Amazon ECR API, peran tersebut harus memiliki kebijakan yang memungkinkan izin untuk melakukan panggilan ECR Amazon ini. Prosedur berikut membantu Anda melampirkan izin yang tepat ke peran tersebut.

Untuk menambahkan izin Amazon ECR ke peran CodeBuild

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi sebelah kiri, pilih Peran.
3. Di kotak pencarian, ketik codebuild- dan pilih peran yang dibuat oleh CodePipeline wizard. Untuk tutorial ini, nama peran adalah codebuild-hello-world-service-role.
4. Pada halaman Ringkasan, pilih Lampirkan kebijakan.
5. Pilih kotak di sebelah kiri EC2 ContainerRegistryPowerUser kebijakan Amazon, dan pilih Lampirkan kebijakan.

Langkah 4: Uji Pipa Anda

Pipeline Anda harus memiliki segalanya untuk menjalankan penerapan AWS berkelanjutan end-to-end asli. Sekarang, uji fungsinya dengan mendorong perubahan kode ke repositori sumber Anda.

Untuk menguji pipa Anda

1. Buat perubahan kode ke repositori sumber yang dikonfigurasi, komit, dan dorong perubahan.

2. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
3. Pilih pipeline Anda dari daftar.
4. Perhatikan kemajuan pipa melalui tahapannya. Pipeline Anda harus selesai dan layanan Amazon ECS Anda menjalankan image Docker yang dibuat dari perubahan kode Anda.

Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR

Dalam tutorial ini, Anda mengonfigurasi pipeline AWS CodePipeline yang menyebarkan aplikasi kontainer menggunakan blue/green penerapan yang mendukung gambar Docker. Dalam penerapan biru/hijau, Anda dapat meluncurkan versi baru aplikasi Anda bersama versi lama dan menguji versi baru sebelum Anda mengubah rute lalu lintas. Anda juga dapat memantau proses penerapan dan memutar kembali dengan cepat jika ada masalah.

Important

Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Note

Tutorial ini untuk Amazon ECS untuk CodeDeploy tindakan penyebaran biru/hijau untuk. CodePipeline Untuk tutorial yang menggunakan tindakan penerapan standar Amazon ECS CodePipeline, lihat. Tutorial: Penerapan Standar Amazon ECS dengan CodePipeline

Pipeline yang telah selesai mendeteksi perubahan pada gambar Anda, yang disimpan dalam repositori gambar seperti Amazon ECR, dan digunakan CodeDeploy untuk merutekan dan menyebarkan lalu lintas ke cluster Amazon ECS dan penyeimbang beban. CodeDeploy menggunakan pendengar untuk mengalihkan lalu lintas ke port wadah yang diperbarui yang ditentukan dalam file. AppSpec Untuk informasi tentang cara penyeimbang beban, pendengar

produksi, grup target, dan aplikasi Amazon ECS Anda digunakan dalam blue/green penerapan, lihat Tutorial: Menerapkan Layanan Amazon ECS.

Pipeline juga dikonfigurasi untuk menggunakan lokasi sumber, seperti CodeCommit, tempat definisi tugas Amazon ECS Anda disimpan. Dalam tutorial ini, Anda mengonfigurasi masing-masing sumber AWS daya ini dan kemudian membuat pipeline Anda dengan tahapan yang berisi tindakan untuk setiap sumber daya.

Pipeline pengiriman berkelanjutan Anda akan secara otomatis membuat dan menerapkan gambar kontainer setiap kali kode sumber diubah atau gambar dasar baru diunggah ke Amazon ECR.

Aliran ini menggunakan artefak berikut:

- File image Docker yang menentukan nama container dan URI repositori repositori image Amazon ECR Anda.
- Definisi tugas Amazon ECS yang mencantumkan nama gambar Docker, nama kontainer, nama layanan Amazon ECS, dan konfigurasi penyeimbang beban.
- CodeDeploy AppSpec File yang menentukan nama file definisi tugas Amazon ECS, nama kontainer aplikasi yang diperbarui, dan port kontainer tempat CodeDeploy mengalihkan lalu lintas produksi. Ini juga dapat menentukan konfigurasi jaringan opsional dan fungsi Lambda yang dapat Anda jalankan selama kait peristiwa siklus hidup penerapan.

> **Note**
>
> Saat Anda melakukan perubahan ke repositori gambar Amazon ECR, tindakan sumber pipeline akan membuat `imageDetail.json` file untuk komit tersebut. Untuk informasi tentang `imageDetail.json` file, lihatFile ImageDetail.json untuk tindakan penerapan Amazon ECS blue/green.

Saat Anda membuat atau mengedit pipeline dan memperbarui atau menentukan artefak sumber untuk tahap penerapan, pastikan untuk menunjuk ke artefak sumber dengan nama dan versi terbaru yang ingin Anda gunakan. Setelah menyiapkan pipeline, saat membuat perubahan pada definisi gambar atau tugas, Anda mungkin perlu memperbarui file artefak sumber di repositori, lalu mengedit tahap penerapan di pipeline.

Topik

- Prasyarat

- Langkah 1: Buat gambar dan dorong ke repositori Amazon ECR
- Langkah 2: Buat definisi tugas dan file AppSpec sumber dan dorong ke CodeCommit repositori
- Langkah 3: Buat Application Load Balancer dan grup target
- Langkah 4: Buat kluster dan layanan Amazon ECS Anda
- Langkah 5: Buat grup CodeDeploy aplikasi dan penyebaran Anda (platform komputasi ECS)
- Langkah 6: Buat pipeline Anda
- Langkah 7: Buat perubahan pada pipeline Anda dan verifikasi penerapan

Prasyarat

Anda harus sudah membuat sumber daya berikut:

- Sebuah CodeCommit repositori. Anda dapat menggunakan AWS CodeCommit repositori yang Anda buat. Tutorial: Buat pipeline sederhana (CodeCommit repositori)
- Luncurkan instance Amazon EC2 Linux dan instal Docker untuk membuat gambar seperti yang ditunjukkan dalam tutorial ini. Jika Anda sudah memiliki gambar yang ingin Anda gunakan, Anda dapat melewati prasyarat ini.

Langkah 1: Buat gambar dan dorong ke repositori Amazon ECR

Di bagian ini, Anda menggunakan Docker untuk membuat gambar dan kemudian menggunakan AWS CLI untuk membuat repositori Amazon ECR dan mendorong gambar ke repositori.

> **Note**
>
> Jika Anda sudah memiliki gambar yang ingin Anda gunakan, Anda dapat melewati langkah ini.

Untuk membuat gambar

1. Masuk ke instance Linux Anda di mana Anda telah menginstal Docker.

   Tarik gambar ke bawah untuk`nginx`. Perintah ini memberikan `nginx:latest` gambar:

   ```
   docker pull nginx
   ```

2. Jalankan docker images. Anda akan melihat gambar dalam daftar.

```
docker images
```

Untuk membuat repositori Amazon ECR dan mendorong gambar Anda

1. Buat repositori Amazon ECR untuk menyimpan gambar Anda. Buat catatan `repositoryUri` di output.

```
aws ecr create-repository --repository-name nginx
```

Output:

```
{
    "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. Tandai gambar dengan `repositoryUri` nilai dari langkah sebelumnya.

```
docker tag nginx:latest aws_account_id.dkr.ecr.us-east-1.amazonaws.com/nginx:latest
```

3. Jalankan aws ecr get-login-password perintah, seperti yang ditunjukkan dalam contoh ini untuk `us-west-2` Region dan ID akun 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. Dorong gambar ke Amazon ECR menggunakan `repositoryUri` dari langkah sebelumnya.

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

Langkah 2: Buat definisi tugas dan file AppSpec sumber dan dorong ke CodeCommit repositori

Di bagian ini, Anda membuat file JSON definisi tugas dan mendaftarkannya ke Amazon ECS. Anda kemudian membuat AppSpec file untuk CodeDeploy dan menggunakan klien Git Anda untuk mendorong file ke CodeCommit repositori Anda.

Untuk membuat definisi tugas untuk gambar Anda

1. Buat file bernama `taskdef.json` dengan isi berikut ini. Untuk`image`, masukkan nama gambar Anda, seperti nginx. Nilai ini diperbarui saat pipeline Anda berjalan.

> **Note**
>
> Pastikan bahwa peran eksekusi yang ditentukan dalam definisi tugas berisi`AmazonECSTaskExecutionRolePolicy`. Untuk informasi selengkapnya, lihat Peran IAM Eksekusi Tugas Amazon ECS di Panduan Pengembang Amazon ECS.

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/ecsTaskExecutionRole",
    "containerDefinitions": [
        {
            "name": "sample-website",
            "image": "nginx",
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "256",
```

```
    "memory": "512",
    "family": "ecs-demo"
}
```

2. Daftarkan definisi tugas Anda dengan `taskdef.json` file.

```
aws ecs register-task-definition --cli-input-json file://taskdef.json
```

3. Setelah definisi tugas terdaftar, edit file Anda untuk menghapus nama gambar dan sertakan teks `<IMAGE1_NAME>` placeholder di bidang gambar.

```
{
    "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"
}
```

Untuk membuat AppSpec file

- AppSpec File ini digunakan untuk CodeDeploy penerapan. File, yang mencakup bidang opsional, menggunakan format ini:

```
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"
```

Untuk informasi selengkapnya tentang AppSpec file, termasuk contoh, lihat Referensi CodeDeploy AppSpec File.

Buat file bernama `appspec.yaml` dengan isi berikut ini. Untuk`TaskDefinition`, jangan mengubah teks <TASK_DEFINITION> placeholder. Nilai ini diperbarui saat pipeline Anda berjalan.

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: <TASK_DEFINITION>
        LoadBalancerInfo:
          ContainerName: "sample-website"
          ContainerPort: 80
```

Untuk mendorong file ke CodeCommit repositori Anda

1. Dorong atau unggah file ke CodeCommit repositori Anda. File-file ini adalah artefak sumber yang dibuat oleh wizard Create pipeline untuk tindakan penerapan Anda. CodePipeline File Anda akan terlihat seperti ini di direktori lokal Anda:

```
/tmp
  |my-demo-repo
    |-- appspec.yaml
    |-- taskdef.json
```

2. Pilih metode yang ingin Anda gunakan untuk mengunggah file Anda:

   a. Untuk menggunakan baris perintah git Anda dari repositori kloning di komputer lokal Anda:

      i. Ubah direktori ke repositori lokal Anda:

```
(For Linux, macOS, or Unix) cd /tmp/my-demo-repo
(For Windows) cd c:\temp\my-demo-repo
```

      ii. Jalankan perintah berikut untuk mementaskan semua file Anda sekaligus:

```
git add -A
```

      iii. Jalankan perintah berikut untuk mengkomit file dengan pesan komit:

```
git commit -m "Added task definition files"
```

      iv. Jalankan perintah berikut untuk mendorong file dari repo lokal Anda ke CodeCommit repositori Anda:

```
git push
```

   b. Untuk menggunakan CodeCommit konsol untuk mengunggah file Anda:

      i. Buka CodeCommit konsol, dan pilih repositori Anda dari daftar Repositori.

      ii. Pilih Tambahkan file, lalu pilih Unggah file.

      iii. Pilih Pilih file, lalu telusuri file Anda. Lakukan perubahan dengan memasukkan nama pengguna dan alamat email Anda. Pilih Perubahan commit.

      iv. Ulangi langkah ini untuk setiap file yang ingin Anda unggah.

Langkah 3: Buat Application Load Balancer dan grup target

Di bagian ini, Anda membuat Application Load Balancer Amazon EC2. Anda menggunakan nama subnet dan nilai grup target yang Anda buat dengan penyeimbang beban nanti, saat Anda membuat layanan Amazon ECS. Anda dapat membuat Application Load Balancer atau Network Load Balancer. Penyeimbang beban harus menggunakan VPC dengan dua subnet publik di Availability Zone yang berbeda. Dalam langkah-langkah ini, Anda mengonfirmasi VPC default Anda, membuat penyeimbang beban, dan kemudian membuat dua grup target untuk penyeimbang beban Anda. Untuk informasi selengkapnya, lihat Grup Target untuk Penyeimbang Beban Jaringan Anda.

Untuk memverifikasi VPC default dan subnet publik

1. Masuk ke Konsol Manajemen AWS dan buka konsol VPC Amazon di. https://console.aws.amazon.com/vpc/
2. Verifikasi VPC default yang akan digunakan. Di panel navigasi, pilih Your VPCs. Perhatikan VPC mana yang menunjukkan Ya di kolom VPC Default. Ini adalah VPC default. Ini berisi subnet default untuk Anda pilih.
3. Pilih Subnet. Pilih dua subnet yang menunjukkan Ya di kolom subnet Default.

   > **Note**
   >
   > Catat subnet IDs Anda. Anda membutuhkannya nanti dalam tutorial ini.

4. Pilih subnet, lalu pilih tab Deskripsi. Verifikasi bahwa subnet yang ingin Anda gunakan berada di Availability Zone yang berbeda.
5. Pilih subnet, lalu pilih tab Route Table. Untuk memverifikasi bahwa setiap subnet yang ingin Anda gunakan adalah subnet publik, konfirmasikan bahwa baris gateway disertakan dalam tabel rute.

Untuk membuat Application Load Balancer Amazon EC2

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon EC2 di. https://console.aws.amazon.com/ec2/
2. Di panel navigasi, pilih Load Balancers.
3. Pilih Buat Penyeimbang Beban.
4. Pilih Application Load Balancer, lalu pilih Create.

5. Di Nama, masukkan nama penyeimbang beban Anda.
6. Dalam Skema, pilih yang menghadap ke internet.
7. Dalam jenis alamat IP, pilih ipv4.
8. Konfigurasikan dua port pendengar untuk penyeimbang beban Anda:
    a. Di bawah Protokol Load Balancer, pilih HTTP. Di bawah Port Load Balancer, masukkan. **80**
    b. Pilih Tambahkan pendengar.
    c. Di bawah Protokol Load Balancer untuk pendengar kedua, pilih HTTP. Di bawah Port Load Balancer, masukkan. **8080**
9. Di bawah Availability Zones, di VPC, pilih VPC default. Selanjutnya, pilih dua subnet default yang ingin Anda gunakan.
10. Pilih Selanjutnya: Konfigurasikan Pengaturan Keamanan.
11. Pilih Selanjutnya: Konfigurasikan Grup Keamanan.
12. Pilih Pilih grup keamanan yang ada, dan buat catatan ID grup keamanan.
13. Pilih Selanjutnya: Konfigurasi Perutean.
14. Di Grup target, pilih Grup target baru dan konfigurasikan grup target pertama Anda:
    a. Di Nama, masukkan nama grup target (misalnya,**target-group-1**).
    b. Pada tipe Target, pilih IP.
    c. Dalam Protokol pilih HTTP. Di Pelabuhan, masukkan**80**.
    d. Pilih Selanjutnya: Daftarkan Target.
15. Pilih Berikutnya: Tinjau, lalu pilih Buat.

Untuk membuat grup target kedua untuk penyeimbang beban Anda

1. Setelah penyeimbang beban Anda disediakan, buka konsol Amazon EC2. Di panel navigasi, pilih Target Groups.
2. PilihBuat grup target.
3. Di Nama, masukkan nama grup target (misalnya,**target-group-2**).
4. Pada tipe Target, pilih IP.
5. Dalam Protokol pilih HTTP. Di Pelabuhan, masukkan**8080**.
6. Di VPC, pilih VPC default.

7. Pilih Buat.

> **Note**
>
> Anda harus memiliki dua grup target yang dibuat untuk penyeimbang beban Anda agar penerapan Anda berjalan. Anda hanya perlu membuat catatan ARN dari kelompok sasaran pertama Anda. ARN ini digunakan dalam file `create-service` JSON pada langkah berikutnya.

Untuk memperbarui penyeimbang beban Anda untuk memasukkan grup target kedua Anda

1. Buka konsol Amazon EC2. Di panel navigasi, pilih Load Balancers.
2. Pilih penyeimbang beban Anda, lalu pilih tab Listeners. Pilih listener dengan port 8080, lalu pilih Edit.
3. Pilih ikon pensil di sebelah Teruskan ke. Pilih grup target kedua Anda, lalu pilih tanda centang. Pilih Perbarui untuk menyimpan pembaruan.

Langkah 4: Buat kluster dan layanan Amazon ECS Anda

Di bagian ini, Anda membuat kluster dan layanan Amazon ECS tempat CodeDeploy merutekan lalu lintas selama penerapan (ke kluster Amazon ECS, bukan instans EC2). Untuk membuat layanan Amazon ECS, Anda harus menggunakan nama subnet, grup keamanan, dan nilai grup target yang Anda buat dengan penyeimbang beban untuk membuat layanan Anda.

> **Note**
>
> Bila Anda menggunakan langkah-langkah ini untuk membuat klaster Amazon ECS, Anda menggunakan template cluster Networking only, yang menyediakan container AWS Fargate. AWS Fargate adalah teknologi yang mengelola infrastruktur instans kontainer Anda untuk Anda. Anda tidak perlu memilih atau membuat instans Amazon EC2 secara manual untuk cluster Amazon ECS Anda.

Untuk membuat cluster Amazon ECS

1. Buka konsol klasik Amazon ECS di https://console.aws.amazon.com/ecs/.

2. Di panel navigasi, pilih Kluster.
3. Pilih Buat kluster.
4. Pilih template cluster Networking only yang menggunakan AWS Fargate, lalu pilih Next step.
5. Masukkan nama cluster pada halaman Configure cluster. Anda dapat menambahkan tag opsional untuk sumber daya Anda. Pilih Buat.

Untuk membuat layanan Amazon ECS

Gunakan AWS CLI untuk membuat layanan Anda di Amazon ECS.

1. Buat file JSON dan beri nama. `create-service.json` Tempel berikut ini ke dalam file JSON.

   Untuk `taskDefinition` bidang ini, saat Anda mendaftarkan definisi tugas di Amazon ECS, Anda memberinya keluarga. Ini mirip dengan nama untuk beberapa versi definisi tugas, ditentukan dengan nomor revisi. Dalam contoh ini, gunakan "`ecs-demo:1`" untuk keluarga dan nomor revisi dalam file Anda. Gunakan nama subnet, grup keamanan, dan nilai grup target yang Anda buat dengan penyeimbang beban Anda. Langkah 3: Buat Application Load Balancer dan grup target

   > **Note**
   >
   > Anda perlu memasukkan ARN grup target Anda dalam file ini. Buka konsol Amazon EC2 dan dari panel navigasi, di bawah LOAD BALANCING, pilih Grup Target. Pilih kelompok target pertama Anda. Salin ARN Anda dari tab Deskripsi.

```
{
    "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. Jalankan create-service perintah, tentukan file JSON:

> **Important**
>
> Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

Contoh ini menciptakan layanan bernamamy`-service`.

> **Note**
>
> Perintah contoh ini membuat layanan bernama my-service. Jika Anda sudah memiliki layanan dengan nama ini, perintah mengembalikan kesalahan.

```
aws ecs create-service --service-name my-service --cli-input-json file://create-
service.json
```

Output mengembalikan kolom deskripsi untuk layanan Anda.

3. Jalankan describe-services perintah untuk memverifikasi bahwa layanan Anda telah dibuat.

```
aws ecs describe-services --cluster cluster-name --services service-name
```

Langkah 5: Buat grup CodeDeploy aplikasi dan penyebaran Anda (platform komputasi ECS)

Saat Anda membuat grup CodeDeploy aplikasi dan penerapan untuk platform komputasi Amazon ECS, aplikasi akan digunakan selama penerapan untuk mereferensikan grup penerapan, grup target, pendengar, dan perilaku pengalihan rute lalu lintas yang benar.

Untuk membuat CodeDeploy aplikasi

1. Buka CodeDeploy konsol dan pilih Buat aplikasi.
2. Di Nama aplikasi, masukkan nama yang ingin Anda gunakan.
3. Di platform Compute, pilih Amazon ECS.
4. Pilih Create application (Buat aplikasi).

Untuk membuat CodeDeploy grup penyebaran

1. Pada tab grup Deployment halaman aplikasi Anda, pilih Buat grup penerapan.
2. Dalam nama grup Deployment, masukkan nama yang menjelaskan grup penyebaran.
3. Dalam peran Layanan, pilih peran layanan yang memberikan CodeDeploy akses ke Amazon ECS. Untuk membuat peran layanan baru, ikuti langkah-langkah berikut:
    a. Buka konsol IAM di https://console.aws.amazon.com/iam/).
    b. Dari dasbor konsol, pilih Peran.
    c. Pilih Buat peran.
    d. Di bawah Pilih jenis entitas tepercaya, pilih Layanan AWS. Di bawah Pilih kasus penggunaan, pilih CodeDeploy. Di bawah Pilih kasus penggunaan Anda, pilih CodeDeploy - ECS. Pilih Berikutnya: Izin. Kebijakan yang `AWSCodeDeployRoleForECS` dikelola sudah melekat pada peran tersebut.
    e. Pilih Berikutnya: Tag, dan Berikutnya: Tinjau.
    f. Masukkan nama untuk peran (misalnya,**`CodeDeployECSRole`**), lalu pilih Buat peran.
4. Dalam konfigurasi Lingkungan, pilih nama cluster Amazon ECS dan nama layanan.
5. Dari Load balancer, pilih nama penyeimbang beban yang melayani lalu lintas ke layanan Amazon ECS Anda.

6. Dari port pendengar Produksi, pilih port dan protokol untuk pendengar yang menyajikan jalur produksi ke layanan Amazon ECS Anda. Dari port Test listener, pilih port dan protokol untuk test listener.
7. Dari nama grup target 1 dan nama Grup target 2, pilih grup target yang digunakan untuk merutekan lalu lintas selama penyebaran Anda. Pastikan bahwa ini adalah kelompok target yang Anda buat untuk penyeimbang beban Anda.
8. Pilih Rute lalu lintas segera untuk menentukan berapa lama setelah penerapan berhasil untuk mengalihkan lalu lintas ke tugas Amazon ECS Anda yang diperbarui.
9. Pilih Buat grup penyebaran.

Langkah 6: Buat pipeline Anda

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- CodeCommit Tindakan di mana artefak sumber adalah definisi tugas dan AppSpec file.
- Tahap sumber dengan aksi sumber Amazon ECR di mana artefak sumber adalah file gambar.
- Tahap penerapan dengan tindakan penerapan Amazon ECS di mana penerapan berjalan dengan grup CodeDeploy aplikasi dan penerapan.

Untuk membuat pipeline dua tahap dengan wizard

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyImagePipeline`**.
5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
6. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.
7. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.

8. Pada Langkah 3: Tambahkan tahap sumber, di penyedia Sumber, pilih AWS CodeCommit. Dalam nama Repositori, pilih nama CodeCommit repositori yang Anda buat. Langkah 1: Buat CodeCommit repositori Di Nama cabang, pilih nama cabang yang berisi pembaruan kode terbaru Anda.

   Pilih Berikutnya.

9. Pada Langkah 4: Tambahkan tahap build, pilih Lewati tahap build, lalu terima pesan peringatan dengan memilih Lewati lagi. Pilih Berikutnya.

10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

11. Pada Langkah 6: Tambahkan tahap penerapan:

    a. Di penyedia Deploy, pilih Amazon ECS (Biru/Hijau). Di Nama aplikasi, masukkan atau pilih nama aplikasi dari daftar, seperti`codedeployapp`. Di grup Deployment, masukkan atau pilih nama grup penyebaran dari daftar, seperti. `codedeploydeplgroup`

    > **Note**
    >
    > Nama “Deploy” adalah nama yang diberikan secara default ke tahap yang dibuat di Langkah 4: Deploy step, sama seperti “Source” adalah nama yang diberikan untuk tahap pertama pipeline.

    b. Di bawah definisi tugas Amazon ECS, pilih SourceArtifact. Di lapangan, masukkan**`taskdef.json`**.

    c. Di bawah AWS CodeDeploy AppSpec file, pilih SourceArtifact. Di lapangan, masukkan**`appspec.yaml`**.

    > **Note**
    >
    > Pada titik ini, jangan mengisi informasi apa pun di bawah Gambar definisi tugas perbarui secara dinamis.

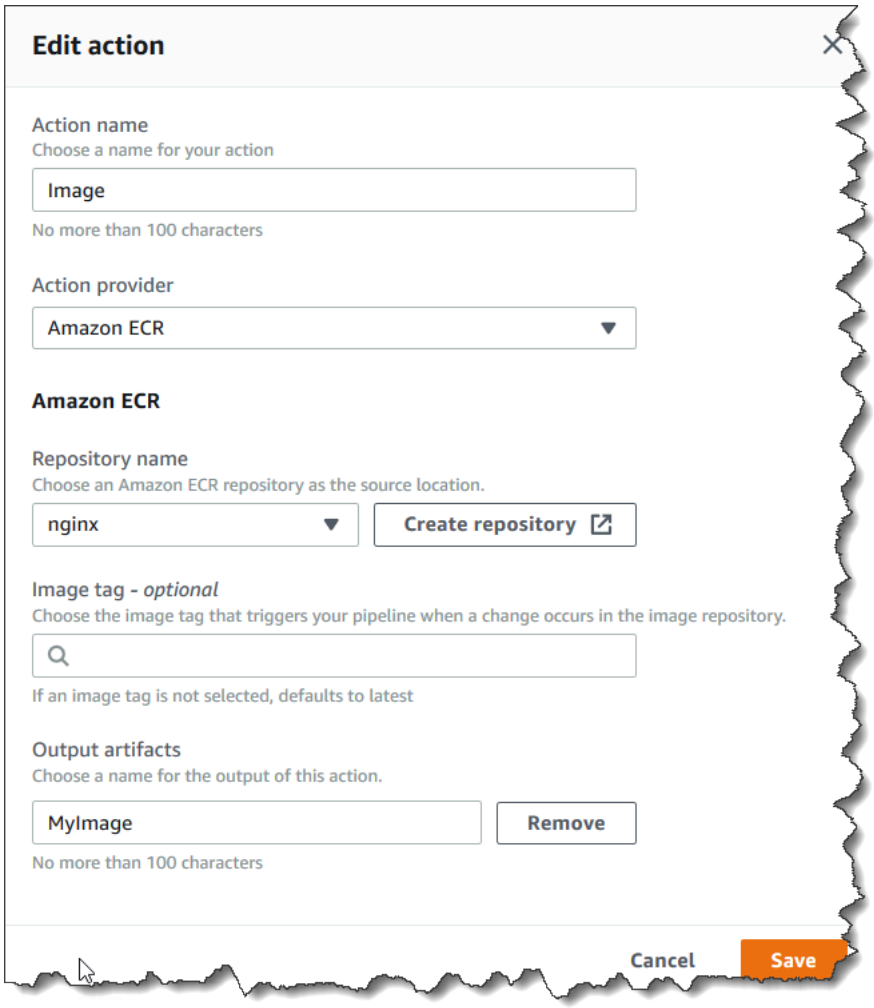
    d. Pilih Berikutnya.

12. Pada Langkah 7: Tinjau, tinjau informasinya, lalu pilih Buat pipeline.

Untuk menambahkan tindakan sumber ECR Amazon ke pipeline Anda

Lihat pipeline Anda dan tambahkan tindakan sumber Amazon ECR ke pipeline Anda.

1. Pilih pipa Anda. Di kiri atas, pilih Edit.
2. Pada tahap sumber, pilih Edit tahap.
3. Tambahkan aksi paralel dengan memilih + Tambahkan tindakan di sebelah tindakan CodeCommit sumber Anda.
4. Di Nama tindakan, masukkan nama (misalnya,**`Image`**).
5. Di penyedia Action, pilih Amazon ECR.



6. Dalam nama Repositori, pilih nama repositori Amazon ECR Anda.
7. Di tag Gambar, tentukan nama dan versi gambar, jika berbeda dari yang terbaru.
8. Di artefak Output, pilih artefak keluaran default (misalnya,`MyImage`) yang berisi nama gambar dan informasi URI repositori yang Anda inginkan untuk digunakan tahap selanjutnya.

9. Pilih Simpan di layar tindakan. Pilih Selesai di layar panggung. Pilih Simpan di pipa. Pesan menunjukkan aturan Amazon CloudWatch Events yang akan dibuat untuk tindakan sumber Amazon ECR.

Untuk menghubungkan artefak sumber Anda ke tindakan penerapan

1. Pilih Edit pada tahap Deploy Anda dan pilih ikon untuk mengedit tindakan Amazon ECS (Biru/Hijau).
2. Gulir ke bagian bawah panel. Di artefak Input, pilih Tambah. Tambahkan artefak sumber dari repositori Amazon ECR baru Anda (misalnya,). `MyImage`
3. Dalam Definisi Tugas SourceArtifact, pilih, lalu verifikasi **`taskdef.json`** dimasukkan.
4. Di AWS CodeDeploy AppSpec File, pilih SourceArtifact, lalu verifikasi **`appspec.yaml`** dimasukkan.
5. Dalam Perbarui gambar definisi tugas secara dinamis, di Artifact Input dengan URI Gambar, MyImagepilih, lalu masukkan teks placeholder yang digunakan dalam file:. `taskdef.json` **`IMAGE1_NAME`** Pilih Simpan.
6. Di AWS CodePipeline panel, pilih Simpan perubahan pipeline, lalu pilih Simpan perubahan. Lihat pipeline Anda yang diperbarui.

   Setelah contoh pipeline ini dibuat, konfigurasi tindakan untuk entri konsol muncul di struktur pipeline sebagai berikut:

   ```
   "configuration": {
     "AppSpecTemplateArtifact": "SourceArtifact",
     "AppSpecTemplatePath": "appspec.yaml",
     "TaskDefinitionTemplateArtifact": "SourceArtifact",
     "TaskDefinitionTemplatePath": "taskdef.json",
     "ApplicationName": "codedeployapp",
     "DeploymentGroupName": "codedeploydeplgroup",
     "Image1ArtifactName": "MyImage",
     "Image1ContainerName": "IMAGE1_NAME"
   },
   ```

7. Untuk mengirimkan perubahan dan memulai pembuatan pipeline, pilih Rilis perubahan, lalu pilih Rilis.
8. Pilih tindakan penerapan untuk melihatnya CodeDeploy dan melihat kemajuan pergeseran lalu lintas.

> **Note**
>
> Anda mungkin melihat langkah penerapan yang menunjukkan waktu tunggu opsional. Secara default, CodeDeploy menunggu satu jam setelah penerapan berhasil sebelum menghentikan set tugas asli. Anda dapat menggunakan waktu ini untuk memutar kembali atau menghentikan tugas, tetapi penerapan Anda akan selesai saat set tugas dihentikan.

Langkah 7: Buat perubahan pada pipeline Anda dan verifikasi penerapan

Buat perubahan pada gambar Anda dan kemudian dorong perubahan ke repositori Amazon ECR Anda. Ini memicu pipeline Anda untuk berjalan. Verifikasi bahwa perubahan sumber gambar Anda diterapkan.

Tutorial: Buat pipeline yang menerapkan keterampilan Amazon Alexa

Dalam tutorial ini, Anda mengonfigurasi pipeline yang terus-menerus memberikan keterampilan Alexa Anda menggunakan Alexa Skills Kit sebagai penyedia penerapan di tahap penerapan Anda. Pipeline yang telah selesai mendeteksi perubahan pada keahlian Anda saat Anda membuat perubahan pada file sumber di repositori sumber Anda. Pipeline kemudian menggunakan Alexa Skills Kit untuk menyebarkan ke tahap pengembangan keterampilan Alexa.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

> **Note**
>
> Fitur ini tidak tersedia di Wilayah Asia Pasifik (Hong Kong) atau Eropa (Milan). Untuk menggunakan tindakan penerapan lain yang tersedia di Wilayah tersebut, lihatMenyebarkan integrasi tindakan.

Untuk membuat skill kustom Anda sebagai fungsi Lambda, lihat Menghosting Keterampilan Kustom sebagai Fungsi Lambda AWS. Anda juga dapat membuat pipeline yang menggunakan file sumber Lambda dan CodeBuild proyek untuk menyebarkan perubahan ke Lambda untuk keahlian Anda.

Prasyarat

Anda harus sudah memiliki yang berikut:

- Sebuah CodeCommit repositori. Anda dapat menggunakan AWS CodeCommit repositori yang Anda buat. Tutorial: Buat pipeline sederhana (CodeCommit repositori)
- Akun pengembang Amazon. Ini adalah akun yang memiliki keterampilan Alexa Anda. Anda dapat membuat akun secara gratis di Alexa Skills Kit.
- Keterampilan Alexa. Anda dapat membuat contoh keterampilan menggunakan tutorial Get Custom Skill Sample Code.
- Instal ASK CLI dan konfigurasikan menggunakan kredensional `ask init` Anda AWS . Lihat Menginstal dan menginisialisasi ASK CLI.

Langkah 1: Buat profil keamanan LWA layanan pengembang Alexa

Di bagian ini, Anda membuat profil keamanan untuk digunakan dengan Login with Amazon (LWA). Jika Anda sudah memiliki profil, Anda dapat melewati langkah ini.

- Gunakan langkah-langkah generate-lwa-tokensuntuk membuat Profil Keamanan.
- Setelah Anda membuat profil, catat ID Klien dan Rahasia Klien.
- Pastikan Anda memasukkan Pengembalian yang Diizinkan URLs seperti yang disediakan dalam instruksi. URLs Izinkan perintah ASK CLI untuk mengarahkan ulang permintaan token penyegaran.

Langkah 2: Buat file sumber keterampilan Alexa dan dorong ke repositori Anda CodeCommit

Di bagian ini, Anda membuat dan mendorong file sumber keterampilan Alexa Anda ke repositori yang digunakan pipeline untuk tahap sumber Anda. Untuk keterampilan yang telah Anda buat di konsol pengembang Amazon, Anda memproduksi dan mendorong yang berikut ini:

- File `skill.json`.
- Sebuah `interactionModel/custom` folder.

> **Note**
>
> Struktur direktori ini sesuai dengan persyaratan format paket keterampilan Alexa Skills Kit, sebagaimana diuraikan dalam format paket Keterampilan. Jika struktur direktori Anda tidak menggunakan format paket keterampilan yang benar, perubahan tidak berhasil diterapkan ke konsol Alexa Skills Kit.

Untuk membuat file sumber untuk keahlian Anda

1. Ambil ID keahlian Anda dari konsol pengembang Alexa Skills Kit. Gunakan perintah ini:

   ```
   ask api list-skills
   ```

   Temukan keahlian Anda berdasarkan nama dan kemudian salin ID terkait di `skillId` bidang.

2. Hasilkan `skill.json` file yang berisi detail keahlian Anda. Gunakan perintah ini:

   ```
   ask api get-skill -s skill-ID > skill.json
   ```

3. (Opsional) Buat `interactionModel/custom` folder.

   Gunakan perintah ini untuk menghasilkan file model interaksi di dalam folder. Untuk lokal, tutorial ini menggunakan en-US sebagai lokal dalam nama file.

   ```
   ask api get-model --skill-id skill-ID --locale locale >
       ./interactionModel/custom/locale.json
   ```

Untuk mendorong file ke CodeCommit repositori Anda

1. Dorong atau unggah file ke CodeCommit repositori Anda. File-file ini adalah artefak sumber yang dibuat oleh wizard Create Pipeline untuk tindakan penerapan Anda. AWS CodePipeline File Anda akan terlihat seperti ini di direktori lokal Anda:

```
skill.json
/interactionModel
  /custom
    |en-US.json
```

2. Pilih metode yang ingin Anda gunakan untuk mengunggah file Anda:

   a. Untuk menggunakan baris perintah Git dari repositori kloning di komputer lokal Anda:

      i. Jalankan perintah berikut untuk mementaskan semua file Anda sekaligus:

      ```
      git add -A
      ```

      ii. Jalankan perintah berikut untuk mengkomit file dengan pesan komit:

      ```
      git commit -m "Added Alexa skill files"
      ```

      iii. Jalankan perintah berikut untuk mendorong file dari repo lokal Anda ke CodeCommit repositori Anda:

      ```
      git push
      ```

   b. Untuk menggunakan CodeCommit konsol untuk mengunggah file Anda:

      i. Buka CodeCommit konsol, dan pilih repositori Anda dari daftar Repositori.

      ii. Pilih Tambahkan file, lalu pilih Unggah file.

      iii. Pilih Pilih file, lalu telusuri file Anda. Lakukan perubahan dengan memasukkan nama pengguna dan alamat email Anda. Pilih Perubahan commit.

      iv. Ulangi langkah ini untuk setiap file yang ingin Anda unggah.

Langkah 3: Gunakan perintah ASK CLI untuk membuat token penyegaran

CodePipeline menggunakan token penyegaran berdasarkan ID klien dan rahasia di akun pengembang Amazon Anda untuk mengotorisasi tindakan yang dilakukannya atas nama Anda. Di bagian ini, Anda menggunakan ASK CLI untuk membuat token. Anda menggunakan kredensi ini saat menggunakan wizard Create Pipeline.

Untuk membuat token penyegaran dengan kredensi akun pengembang Amazon Anda

1. Gunakan perintah berikut ini.

   ```
   ask util generate-lwa-tokens
   ```

2. Saat diminta, masukkan ID klien dan rahasia Anda seperti yang ditunjukkan dalam contoh ini:

   ```
   ? Please type in the client ID:
   amzn1.application-client.example112233445566
   ? Please type in the client secret:
   example112233445566
   ```

3. Halaman browser masuk ditampilkan. Masuk dengan kredensi akun pengembang Amazon Anda.
4. Kembali ke layar baris perintah. Token akses dan token penyegaran dihasilkan dalam output. Salin token penyegaran yang dikembalikan dalam output.

Langkah 4: Buat pipeline Anda

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan CodeCommit aksi di mana artefak sumber adalah file keterampilan Alexa yang mendukung keahlian Anda.
- Tahap penyebaran dengan tindakan penerapan Alexa Skills Kit.

Untuk membuat alur dengan wizard

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pilih AWS Wilayah tempat Anda ingin membuat proyek dan sumber dayanya. Runtime skill Alexa hanya tersedia di Wilayah berikut:

- Asia Pasifik (Tokyo)
- Eropa (Irlandia)
- AS Timur (Virginia Utara)
- AS Barat (Oregon)

3. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
4. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
5. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyAlexaPipeline`**.
6. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
7. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.
8. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.
9. Pada Langkah 3: Tambahkan tahap sumber, di penyedia Sumber, pilih AWS CodeCommit. Dalam nama Repositori, pilih nama CodeCommit repositori yang Anda buat. Langkah 1: Buat CodeCommit repositori Di Nama cabang, pilih nama cabang yang berisi pembaruan kode terbaru Anda.

   Setelah Anda memilih nama repositori dan cabang, pesan menunjukkan aturan Amazon CloudWatch Events yang akan dibuat untuk pipeline ini.

   Pilih Berikutnya.
10. Pada Langkah 4: Tambahkan tahap build, pilih Lewati tahap build, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.
11. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.
12. Pada Langkah 6: Tambahkan tahap penerapan:

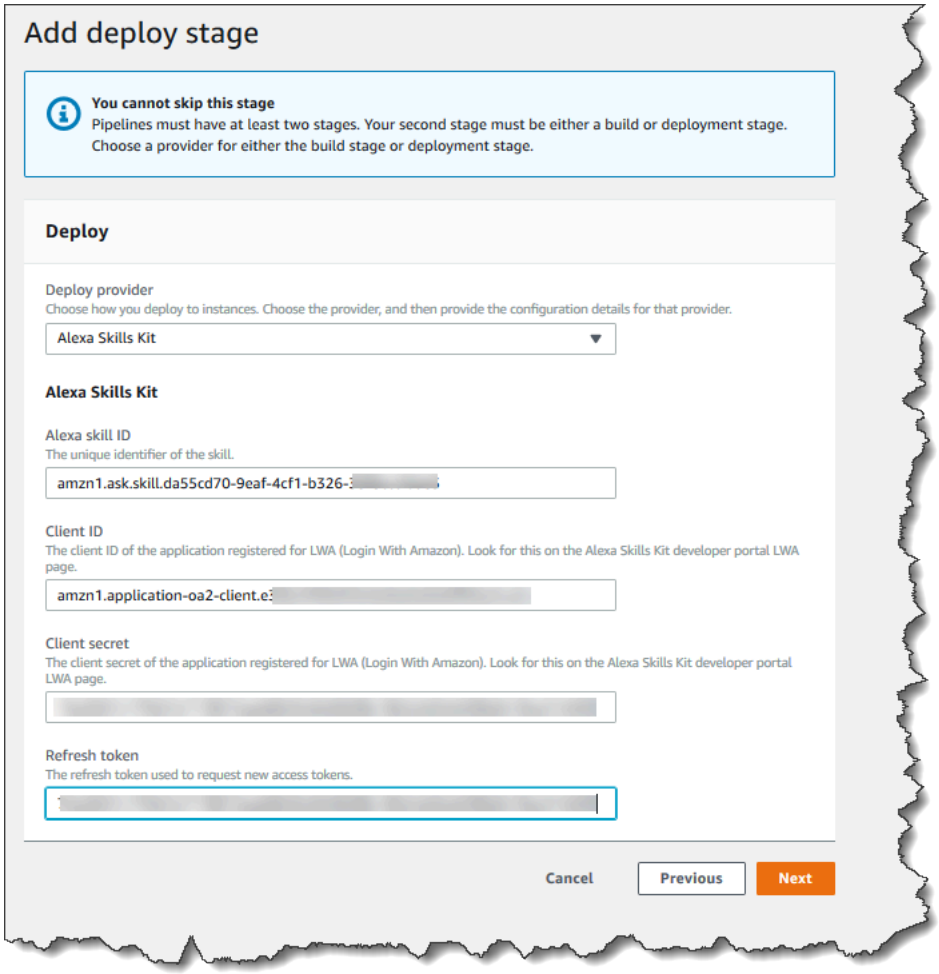
    a. Di penyedia Deploy, pilih Alexa Skills Kit.

b. Di ID keterampilan Alexa, masukkan ID keterampilan yang ditetapkan untuk keahlian Anda di konsol pengembang Alexa Skills Kit.

c. Di Client ID, masukkan ID aplikasi yang Anda daftarkan.

d. Dalam rahasia Klien, masukkan rahasia yang Anda pilih saat mendaftar.

e. Di Refresh token, masukkan token yang Anda buat di langkah 3.



f. Pilih Berikutnya.

13. Pada Langkah 7: Tinjau, tinjau informasinya, lalu pilih Buat pipeline.

Langkah 5: Buat perubahan pada file sumber apa pun dan verifikasi penyebaran

Buat perubahan pada keahlian Anda dan kemudian dorong perubahan ke repositori Anda. Ini memicu pipeline Anda untuk berjalan. Verifikasi bahwa keahlian Anda diperbarui di konsol pengembang Alexa Skills Kit.

Tutorial: Membuat pipeline yang menggunakan Amazon S3 sebagai penyedia penerapan

Dalam tutorial ini, Anda mengonfigurasi pipeline yang terus-menerus mengirimkan file menggunakan Amazon S3 sebagai penyedia tindakan penerapan di tahap penerapan. Pipeline yang telah selesai mendeteksi perubahan saat Anda membuat perubahan pada file sumber di repositori sumber Anda. Pipeline kemudian menggunakan Amazon S3 untuk menyebarkan file ke bucket Anda. Setiap kali Anda memodifikasi atau menambahkan file situs web Anda di lokasi sumber Anda, penyebaran membuat situs web dengan file terbaru Anda.

Important

Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Note

Bahkan jika Anda menghapus file dari repositori sumber, tindakan penerapan S3 tidak menghapus objek S3 yang sesuai dengan file yang dihapus.

Tutorial ini menyediakan dua opsi:

- Buat pipeline yang menyebarkan situs web statis ke bucket publik S3 Anda. Contoh ini membuat pipeline dengan aksi AWS CodeCommit sumber dan tindakan penerapan Amazon S3. Lihat Opsi 1: Menyebarkan file situs web statis ke Amazon S3.
- Buat pipeline yang mengkompilasi TypeScript kode sampel ke dalam JavaScript dan menyebarkan artefak CodeBuild keluaran ke bucket S3 Anda untuk arsip. Contoh ini membuat pipeline dengan aksi sumber Amazon S3, tindakan CodeBuild build, dan tindakan penerapan Amazon S3. Lihat Opsi 2: Menerapkan file arsip bawaan ke Amazon S3 dari bucket sumber S3.

> **Important**
>
> Banyak tindakan yang Anda tambahkan ke pipeline dalam prosedur ini melibatkan AWS sumber daya yang perlu Anda buat sebelum membuat pipeline. AWS sumber daya untuk tindakan sumber Anda harus selalu dibuat di AWS Wilayah yang sama tempat Anda membuat pipeline. Misalnya, jika Anda membuat pipeline di Wilayah AS Timur (Ohio), CodeCommit repositori Anda harus berada di Wilayah AS Timur (Ohio).
> Anda dapat menambahkan tindakan lintas wilayah saat membuat pipeline. AWS sumber daya untuk tindakan lintas wilayah harus berada di AWS Wilayah yang sama di mana Anda berencana untuk menjalankan tindakan. Untuk informasi selengkapnya, lihat Menambahkan tindakan Lintas wilayah di CodePipeline.

Opsi 1: Menyebarkan file situs web statis ke Amazon S3

Dalam contoh ini, Anda mengunduh contoh file template situs web statis, mengunggah file ke AWS CodeCommit repositori, membuat bucket, dan mengonfigurasinya untuk hosting. Selanjutnya, Anda menggunakan AWS CodePipeline konsol untuk membuat pipeline dan menentukan konfigurasi penerapan Amazon S3.

Prasyarat

Anda harus sudah memiliki yang berikut:

- Sebuah CodeCommit repositori. Anda dapat menggunakan AWS CodeCommit repositori yang Anda buat. Tutorial: Buat pipeline sederhana (CodeCommit repositori)
- File sumber untuk situs web statis Anda. Gunakan tautan ini untuk mengunduh contoh situs web statis. Unduhan sample-website.zip menghasilkan file-file berikut:
  - Sebuah `index.html` file
  - Sebuah `main.css` file
  - Sebuah `graphic.jpg` file
- Bucket S3 yang dikonfigurasi untuk hosting situs web. Lihat Hosting situs web statis di Amazon S3. Pastikan Anda membuat bucket di Wilayah yang sama dengan pipeline.

> **Note**
>
> Untuk meng-host situs web, bucket Anda harus memiliki akses baca publik, yang memberi semua orang akses baca. Dengan pengecualian hosting situs web, Anda harus menjaga pengaturan akses default yang memblokir akses publik ke bucket S3.

Langkah 1: Dorong file sumber ke CodeCommit repositori Anda

Di bagian ini, dorong file sumber Anda ke repositori yang digunakan pipeline untuk tahap sumber Anda.

Untuk mendorong file ke CodeCommit repositori Anda

1. Ekstrak file sampel yang diunduh. Jangan mengunggah file ZIP ke repositori Anda.
2. Dorong atau unggah file ke CodeCommit repositori Anda. File-file ini adalah artefak sumber yang dibuat oleh wizard Create Pipeline untuk tindakan penerapan Anda. CodePipeline File Anda akan terlihat seperti ini di direktori lokal Anda:

   ```
   index.html
   main.css
   graphic.jpg
   ```

3. Anda dapat menggunakan Git atau CodeCommit konsol untuk mengunggah file Anda:

   a. Untuk menggunakan baris perintah Git dari repositori kloning di komputer lokal Anda:

      i. Jalankan perintah berikut untuk mementaskan semua file Anda sekaligus:

      ```
      git add -A
      ```

      ii. Jalankan perintah berikut untuk mengkomit file dengan pesan komit:

      ```
      git commit -m "Added static website files"
      ```

      iii. Jalankan perintah berikut untuk mendorong file dari repo lokal Anda ke CodeCommit repositori Anda:

      ```
      git push
      ```

b. Untuk menggunakan CodeCommit konsol untuk mengunggah file Anda:

i. Buka CodeCommit konsol, dan pilih repositori Anda dari daftar Repositori.

ii. Pilih Tambahkan file, lalu pilih Unggah file.

iii. Pilih Pilih file, lalu telusuri file Anda. Lakukan perubahan dengan memasukkan nama pengguna dan alamat email Anda. Pilih Perubahan commit.

iv. Ulangi langkah ini untuk setiap file yang ingin Anda unggah.

Langkah 2: Buat alur Anda

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan CodeCommit tindakan di mana artefak sumber adalah file untuk situs web Anda.
- Tahap penerapan dengan tindakan penerapan Amazon S3.

Untuk membuat alur dengan wizard

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Alur, pilih Buat alur.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyS3DeployPipeline`**.
5. Dalam tipe Pipeline, pilih V2. Untuk informasi selengkapnya, lihat Jenis pipa. Pilih Berikutnya.
6. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.
7. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.
8. Pada Langkah 3: Tambahkan tahap sumber, di penyedia Sumber, pilih AWS CodeCommit. Dalam nama Repositori, pilih nama CodeCommit repositori yang Anda buat. Langkah 1: Buat CodeCommit repositori Di Nama cabang, pilih nama cabang yang berisi pembaruan kode terbaru Anda. Kecuali Anda membuat cabang yang berbeda sendiri, hanya `main` tersedia.

Setelah Anda memilih nama repositori dan cabang, aturan Amazon CloudWatch Events yang akan dibuat untuk pipeline ini akan ditampilkan.

Pilih Berikutnya.

9. Pada Langkah 4: Tambahkan tahap build, pilih Lewati tahap build, lalu terima pesan peringatan dengan memilih Lewati lagi.

   Pilih Berikutnya.

10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

11. Pada Langkah 6: Tambahkan tahap penerapan:

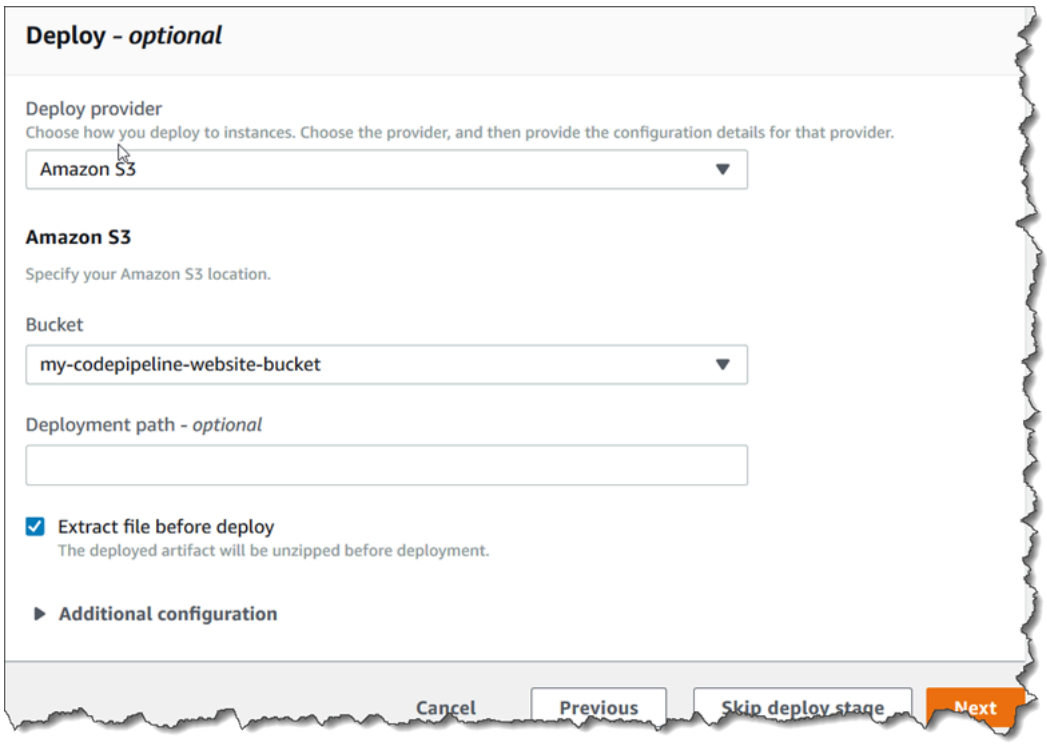
    a. Di penyedia Deploy, pilih Amazon S3.

    b. Di Bucket, masukkan nama ember publik Anda.

    c. Pilih Ekstrak file sebelum menerapkan.

    > **Note**
    >
    > Penerapan gagal jika Anda tidak memilih Ekstrak file sebelum menerapkan. Ini karena AWS CodeCommit tindakan dalam pipeline Anda meritsleting artefak sumber dan file Anda adalah file ZIP.

    Saat Ekstrak file sebelum penerapan dipilih, jalur Deployment ditampilkan. Masukkan nama jalur yang ingin Anda gunakan. Ini menciptakan struktur folder di Amazon S3 tempat file diekstraksi. Untuk tutorial ini, biarkan bidang ini kosong.



d. (Opsional) Di ACL Kalengan, Anda dapat menerapkan serangkaian hibah yang telah ditentukan sebelumnya, yang dikenal sebagai ACL kalengan, ke artefak yang diunggah.

e. (Opsional) Dalam kontrol Cache, masukkan parameter caching. Anda dapat mengatur ini untuk mengontrol perilaku caching untuk permintaan/tanggapan. Untuk nilai yang valid, lihat bidang `Cache-Control` header untuk operasi HTTP.

f. Pilih Berikutnya.

12. Pada Langkah 7: Tinjau, tinjau informasinya, lalu pilih Buat pipeline.

13. Setelah pipeline berhasil berjalan, buka konsol Amazon S3 dan verifikasi bahwa file Anda muncul di bucket publik seperti yang ditunjukkan:

```
index.html
main.css
graphic.jpg
```

14. Akses titik akhir Anda untuk menguji situs web. Titik akhir Anda mengikuti format ini:`http://bucket-name.s3-website-region.amazonaws.com/`.

    Contoh titik akhir:`http://my-bucket.s3-website-us-west-2.amazonaws.com/`.

    Halaman web sampel muncul.

Langkah 3: Buat perubahan pada file sumber apa pun dan verifikasi penyebaran

Buat perubahan pada file sumber Anda dan kemudian dorong perubahan ke repositori Anda. Ini memicu pipeline Anda untuk berjalan. Verifikasi bahwa situs web Anda diperbarui.

Opsi 2: Menerapkan file arsip bawaan ke Amazon S3 dari bucket sumber S3

Dalam opsi ini, perintah build di tahap build Anda mengompilasi TypeScript kode ke dalam JavaScript kode dan menerapkan output ke bucket target S3 Anda di bawah folder stempel waktu terpisah. Pertama, Anda membuat TypeScript kode dan file buildspec.yml. Setelah Anda menggabungkan file sumber dalam file ZIP, Anda mengunggah file ZIP sumber ke bucket sumber S3 Anda, dan menggunakan CodeBuild panggung untuk menyebarkan file ZIP aplikasi bawaan ke bucket target S3 Anda. Kode yang dikompilasi dipertahankan sebagai arsip di bucket target Anda.

Prasyarat

Anda harus sudah memiliki yang berikut:

- Bucket sumber S3. Anda dapat menggunakan ember yang Anda buat[Tutorial: Buat pipeline sederhana (ember S3)](#).
- Ember target S3. Lihat [Hosting situs web statis di Amazon S3](#). Pastikan Anda membuat bucket Wilayah AWS sama dengan pipeline yang ingin Anda buat.

> **Note**
>
> Contoh ini menunjukkan penerapan file ke bucket pribadi. Jangan aktifkan bucket target Anda untuk hosting situs web atau lampirkan kebijakan apa pun yang membuat bucket publik.

Langkah 1: Buat dan unggah file sumber ke bucket sumber S3 Anda

Di bagian ini, Anda membuat dan mengunggah file sumber ke bucket yang digunakan pipeline untuk tahap sumber Anda. Bagian ini memberikan instruksi untuk membuat file sumber berikut:

- `buildspec.yml`File, yang digunakan untuk CodeBuild membangun proyek.
- Sebuah `index.ts` file.

Untuk membuat file buildspec.yml.

- Buat file bernama `buildspec.yml` dengan isi berikut ini. Perintah build ini menginstal TypeScript dan menggunakan TypeScript compiler untuk menulis ulang kode `index.ts` ke JavaScript kode.

```
version: 0.2

phases:
  install:
    commands:
      - npm install -g typescript
  build:
    commands:
      - tsc index.ts
artifacts:
  files:
    - index.js
```

Untuk membuat file index.ts

- Buat file bernama `index.ts` dengan isi berikut ini.

```
interface Greeting {
    message: string;
}

class HelloGreeting implements Greeting {
    message = "Hello!";
}

function greet(greeting: Greeting) {
    console.log(greeting.message);
}

let greeting = new HelloGreeting();

greet(greeting);
```

Untuk mengunggah file ke bucket sumber S3

1. File Anda akan terlihat seperti ini di direktori lokal Anda:

   ```
   buildspec.yml
   index.ts
   ```

   Zip file dan beri nama file`source.zip`.
2. Di konsol Amazon S3, untuk bucket sumber Anda, pilih Unggah. Pilih Tambahkan file, lalu telusuri file ZIP yang Anda buat.
3. Pilih Unggah. File-file ini adalah artefak sumber yang dibuat oleh wizard Create Pipeline untuk tindakan penerapan Anda. CodePipeline File Anda akan terlihat seperti ini di bucket Anda:

   ```
   source.zip
   ```

Langkah 2: Buat alur Anda

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan aksi Amazon S3 di mana artefak sumber adalah file untuk aplikasi yang dapat Anda unduh.
- Tahap penerapan dengan tindakan penerapan Amazon S3.

Untuk membuat alur dengan wizard

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Alur, pilih Buat alur.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyS3DeployPipeline`**.
5. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.
6. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.

7. Pada Langkah 3: Tambahkan tahap sumber, di penyedia Sumber, pilih Amazon S3. Di Bucket, pilih nama bucket sumber Anda. Dalam kunci objek S3, masukkan nama file ZIP sumber Anda. Pastikan Anda menyertakan ekstensi file.zip.

   Pilih Berikutnya.

8. Pada Langkah 4: Tambahkan tahap build:

   a. Di Penyedia pembangunan, pilih CodeBuild.

   b. Pilih Buat proyek build. Pada halaman Create project:

   c. Di Nama proyek, masukkan nama untuk proyek pembangunan ini.

   d. Di Lingkungan, pilih Gambar terkelola. Untuk Sistem operasi, pilih Ubuntu.

   e. Untuk Waktu aktif, pilih Standar. Untuk versi Runtime, aws/codebuild/standard pilih:1.0.

   f. Dalam versi Gambar, pilih Selalu gunakan gambar terbaru untuk versi runtime ini.

   g. Untuk peran Layanan, pilih peran CodeBuild layanan Anda, atau buat peran layanan.

   h. Untuk spesifikasi Build, pilih Gunakan file buildspec.

   i. Pilih Lanjutkan ke CodePipeline. Pesan ditampilkan jika proyek berhasil dibuat.

   j. Pilih Berikutnya.

9. Pada Langkah 5: Tambahkan tahap penerapan:

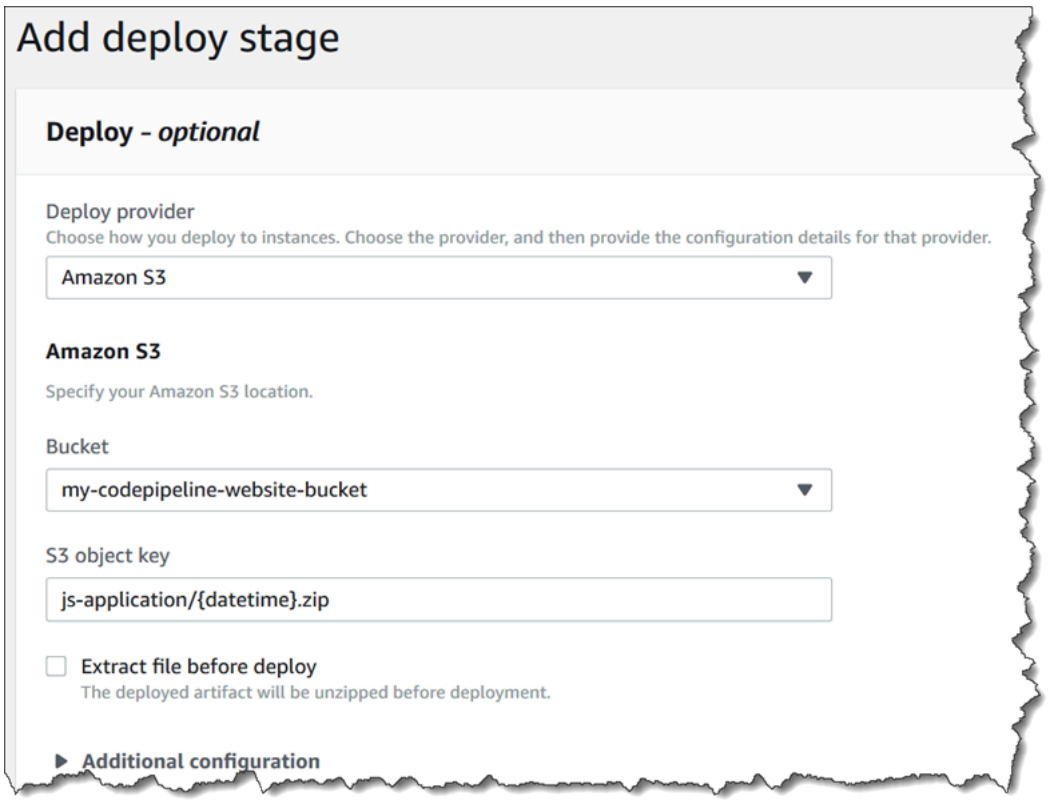
   a. Di penyedia Deploy, pilih Amazon S3.

   b. Di Bucket, masukkan nama bucket target S3 Anda.

   c. Pastikan file Extract sebelum deploy dihapus.

      Saat Ekstrak file sebelum penerapan dihapus, kunci objek S3 ditampilkan. Masukkan nama jalur yang ingin Anda gunakan:`js-application/{datetime}.zip`.

      Ini membuat `js-application` folder di Amazon S3 tempat file diekstraksi. Di folder ini, `{datetime}` variabel membuat stempel waktu pada setiap file keluaran saat pipeline Anda berjalan.



d. (Opsional) Di ACL Kalengan, Anda dapat menerapkan serangkaian hibah yang telah ditentukan sebelumnya, yang dikenal sebagai ACL kalengan, ke artefak yang diunggah.

e. (Opsional) Dalam kontrol Cache, masukkan parameter caching. Anda dapat mengatur ini untuk mengontrol perilaku caching untuk permintaan/tanggapan. Untuk nilai yang valid, lihat bidang `Cache-Control`header untuk operasi HTTP.

f. Pilih Berikutnya.

10. Pada Langkah 6: Tinjau, tinjau informasinya, lalu pilih Buat pipeline.

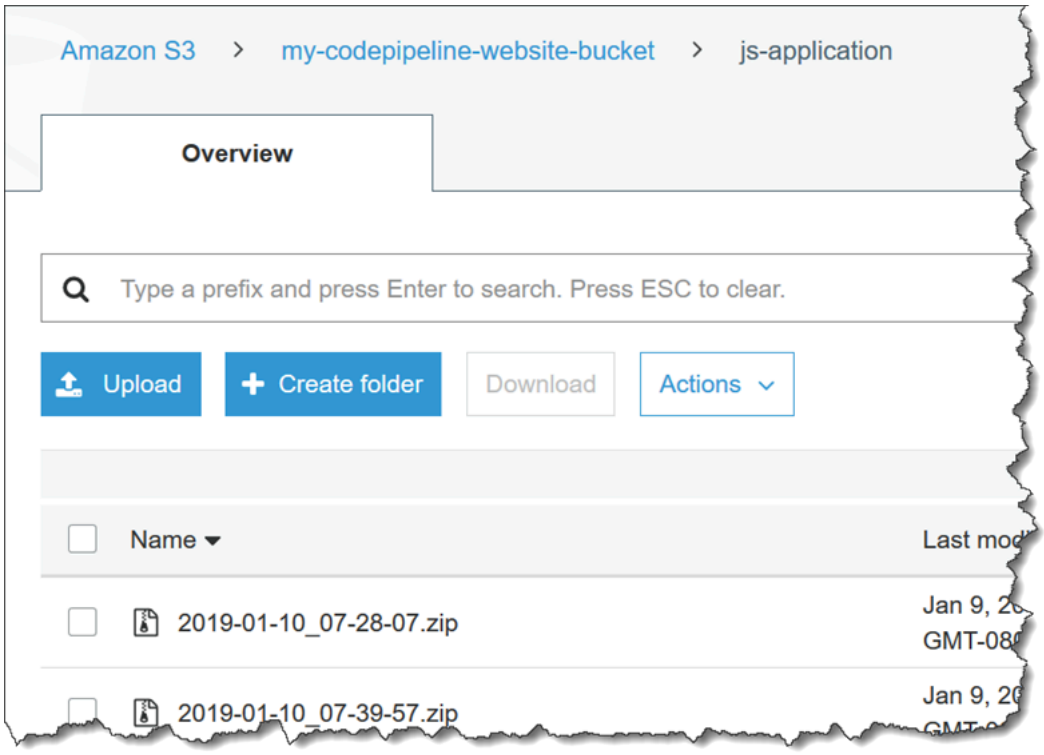
11. Setelah pipeline berhasil berjalan, lihat bucket Anda di konsol Amazon S3. Verifikasi bahwa file ZIP yang Anda gunakan ditampilkan di bucket target Anda di bawah `js-application` folder. JavaScript File yang terkandung dalam file ZIP harus`index.js`. `index.js`File berisi output berikut:

```
var HelloGreeting = /** @class */ (function () {
    function HelloGreeting() {
        this.message = "Hello!";
    }
    return HelloGreeting;
}());
function greet(greeting) {
```

```
    console.log(greeting.message);
}
var greeting = new HelloGreeting();
greet(greeting);
```

Langkah 3: Buat perubahan pada file sumber apa pun dan verifikasi penyebaran

Buat perubahan pada file sumber Anda dan kemudian unggah ke keranjang sumber Anda. Ini memicu pipeline Anda untuk berjalan. Lihat bucket target Anda dan verifikasi bahwa file keluaran yang digunakan tersedia di `js-application` folder seperti yang ditunjukkan:



Tutorial: Buat pipeline yang menerbitkan aplikasi tanpa server Anda ke AWS Serverless Application Repository

Anda dapat menggunakan AWS CodePipeline untuk terus mengirimkan aplikasi AWS SAM tanpa server Anda ke. AWS Serverless Application Repository

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Tutorial ini menunjukkan cara membuat dan mengkonfigurasi pipeline untuk membangun aplikasi tanpa server Anda yang di-host GitHub dan mempublikasikannya secara otomatis. AWS Serverless Application Repository Pipeline digunakan GitHub sebagai penyedia sumber dan CodeBuild sebagai penyedia build. Untuk memublikasikan aplikasi tanpa server ke AWS Serverless Application Repository, Anda menerapkan aplikasi (dari AWS Serverless Application Repository) dan mengaitkan fungsi Lambda yang dibuat oleh aplikasi tersebut sebagai penyedia tindakan Invoke di pipeline Anda. Kemudian Anda dapat terus mengirimkan pembaruan aplikasi ke AWS Serverless Application Repository, tanpa menulis kode apa pun.

> **Important**
>
> Banyak tindakan yang Anda tambahkan ke pipeline dalam prosedur ini melibatkan AWS sumber daya yang perlu Anda buat sebelum membuat pipeline. AWS sumber daya untuk tindakan sumber Anda harus selalu dibuat di AWS Wilayah yang sama tempat Anda membuat pipeline. Misalnya, jika Anda membuat pipeline di Wilayah AS Timur (Ohio), CodeCommit repositori Anda harus berada di Wilayah AS Timur (Ohio).
> Anda dapat menambahkan tindakan lintas wilayah saat membuat pipeline. AWS sumber daya untuk tindakan lintas wilayah harus berada di AWS Wilayah yang sama di mana Anda berencana untuk menjalankan tindakan. Untuk informasi selengkapnya, lihat Menambahkan tindakan Lintas wilayah di CodePipeline.

Sebelum kamu memulai

Dalam tutorial ini, kita asumsikan sebagai berikut.

- Anda sudah familiar dengan AWS Serverless Application Model (AWS SAM) dan AWS Serverless Application Repository.

- Anda memiliki aplikasi tanpa server yang dihosting di mana GitHub Anda telah mempublikasikan ke AWS Serverless Application Repository menggunakan CLI AWS SAM . Untuk mempublikasikan contoh aplikasi ke AWS Serverless Application Repository, lihat Mulai Cepat: Menerbitkan Aplikasi di Panduan AWS Serverless Application Repository Pengembang. Untuk mempublikasikan aplikasi Anda sendiri ke AWS Serverless Application Repository, lihat Menerbitkan Aplikasi Menggunakan AWS SAM CLI di Panduan AWS Serverless Application Model Pengembang.

Langkah 1: Buat file buildspec.yml.

Buat `buildspec.yml` file dengan konten berikut, dan tambahkan ke repositori aplikasi tanpa server Anda. GitHub Ganti *`template.yml`* dengan AWS SAM template aplikasi Anda dan *`bucketname`* dengan bucket S3 tempat aplikasi paket Anda disimpan.

```
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
```

Langkah 2: Buat dan konfigurasikan pipeline Anda

Ikuti langkah-langkah ini untuk membuat pipeline Anda di Wilayah AWS tempat Anda ingin mempublikasikan aplikasi tanpa server Anda.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Jika perlu, beralihlah ke Wilayah AWS tempat Anda ingin mempublikasikan aplikasi tanpa server Anda.
3. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
4. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.

5. Pilih Buat pipeline. Pada Langkah 2: Pilih halaman pengaturan pipeline, dalam nama Pipeline, masukkan nama untuk pipeline Anda.
6. Dalam tipe Pipeline, pilih V2. Untuk informasi selengkapnya, lihat Jenis pipa. Pilih Berikutnya.
7. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.
8. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.
9. Pada Langkah 3: Tambahkan halaman tahap sumber, di penyedia Sumber, pilih GitHub.
10. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.
11. Di Repositori, pilih repositori GitHub sumber Anda.
12. Di Cabang, pilih GitHub cabang Anda.
13. Biarkan default yang tersisa untuk tindakan sumber. Pilih Berikutnya.
14. Pada Langkah 4: Tambahkan halaman tahap build, tambahkan tahap build:
    a. Di Penyedia pembangunan, pilih AWS CodeBuild. Untuk Wilayah, gunakan Wilayah pipa.
    b. Pilih Buat proyek.
    c. Di Nama proyek, masukkan nama untuk proyek pembangunan ini.
    d. Di Citra lingkungan, pilih Citra terkelola. Untuk Sistem operasi, pilih Ubuntu.
    e. Untuk versi Runtime dan Runtime, pilih runtime dan versi yang diperlukan untuk aplikasi tanpa server Anda.
    f. Untuk Peran layanan, pilih Peran layanan baru.
    g. Untuk spesifikasi Build, pilih Gunakan file buildspec.
    h. Pilih Lanjutkan ke CodePipeline. Ini membuka CodePipeline konsol dan membuat CodeBuild proyek yang menggunakan `buildspec.yml` di repositori Anda untuk konfigurasi. Proyek build menggunakan peran layanan untuk mengelola Layanan AWS izin. Langkah ini mungkin memakan waktu beberapa menit.
    i. Pilih Berikutnya.
15. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.
16. Pada Langkah 6: Tambahkan halaman tahap penerapan, pilih Lewati tahap penerapan, lalu terima pesan peringatan dengan memilih Lewati lagi. Pilih Berikutnya.

17. Pada Langkah 7: Tinjau, pilih Buat pipeline. Anda harus melihat diagram yang menunjukkan tahapan.
18. Berikan izin peran CodeBuild layanan untuk mengakses bucket S3 tempat aplikasi paket Anda disimpan.

    a. Pada tahap Build pipeline baru Anda, pilih CodeBuild.

    b. Pilih tab Build details.

    c. Di Lingkungan, pilih peran CodeBuild layanan untuk membuka konsol IAM.

    d. Perluas pilihan untuk`CodeBuildBasePolicy`, dan pilih Edit kebijakan.

    e. PilihJSON.

    f. Tambahkan pernyataan kebijakan baru dengan konten berikut. Pernyataan ini memungkinkan CodeBuild untuk memasukkan objek ke dalam bucket S3 tempat aplikasi paket Anda disimpan. Ganti *`bucketname`* dengan nama bucket S3 Anda.

```
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::bucketname/*"
            ],
            "Action": [
                "s3:PutObject"
            ]
        }
```

    g. Pilih Tinjau kebijakan.

    h. Pilih Simpan perubahan.

Langkah 3: Menyebarkan aplikasi publikasi

Ikuti langkah-langkah ini untuk menyebarkan aplikasi yang berisi fungsi Lambda yang melakukan publikasi ke. AWS Serverless Application Repository Aplikasi ini adalah aws-serverless-codepipeline-serverlessrepo-publish.

> **Note**
>
> Anda harus menerapkan aplikasi yang Wilayah AWS sama dengan pipeline Anda.

1. Buka halaman aplikasi, dan pilih Deploy.
2. PilihSaya mengakui bahwa aplikasi ini menciptakan IAM role kustom.
3. Pilih Deploy.
4. Pilih View CloudFormation Stack untuk membuka CloudFormation konsol.
5. Perluas bagian Sumber Daya. Anda lihat ServerlessRepoPublish, yang merupakan tipe AWS::Lambda::Function. Catat ID fisik sumber daya ini untuk langkah selanjutnya. Anda menggunakan ID fisik ini saat membuat tindakan publikasi baru di CodePipeline.

Langkah 4: Buat tindakan publikasi

Ikuti langkah-langkah ini untuk membuat tindakan publikasi di pipeline Anda.

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Di bagian navigasi kiri, pilih pipeline yang ingin Anda edit.
3. Pilih Edit.
4. Setelah tahap terakhir dari pipeline Anda saat ini, pilih + Tambah tahap. Di Nama panggung masukkan nama, seperti**`Publish`**, dan pilih Tambah tahap.
5. Di tahap baru, pilih + Tambahkan grup tindakan.
6. Masukkan nama tindakan. Dari penyedia Action, di Invoke, pilih AWS Lambda.
7. Dari artefak Input, pilih BuildArtifact.
8. Dari nama Fungsi, pilih ID fisik dari fungsi Lambda yang Anda catat di langkah sebelumnya.
9. Pilih Simpan untuk tindakan.
10. Pilih Selesai untuk panggung.
11. Di kanan atas, pilih Simpan.
12. Untuk memverifikasi pipeline Anda, buat perubahan pada aplikasi Anda di GitHub. Misalnya, ubah deskripsi aplikasi di `Metadata` bagian file AWS SAM template Anda. Lakukan perubahan dan dorong ke GitHub cabang Anda. Ini memicu pipeline Anda untuk berjalan. Ketika pipeline selesai, periksa apakah aplikasi Anda telah diperbarui dengan perubahan Anda di file AWS Serverless Application Repository.

Tutorial: Penerapan fungsi Lambda dengan CodePipeline

Tutorial ini membantu Anda membuat tindakan penerapan yang menyebarkan kode Anda ke fungsi CodePipeline yang telah Anda konfigurasikan di Lambda. Anda akan membuat contoh fungsi Lambda di mana Anda akan membuat alias dan versi, menambahkan fungsi Lambda zip ke lokasi sumber Anda, dan menjalankan tindakan Lambda di pipeline Anda.

> **Note**
>
> Sebagai bagian dari pembuatan pipa di konsol, ember artefak S3 akan digunakan oleh CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

> **Note**
>
> Tindakan `Lambda` penerapan hanya tersedia untuk saluran pipa tipe V2.

Prasyarat

Ada beberapa sumber daya yang harus Anda miliki sebelum Anda dapat menggunakan tutorial ini untuk membuat pipeline CD Anda. Berikut adalah hal-hal yang Anda butuhkan untuk memulai:

> **Note**
>
> Semua sumber daya ini harus dibuat dalam AWS Wilayah yang sama.

- Repositori kontrol sumber, seperti GitHub, atau bucket S3 sumber (tutorial ini menggunakan S3) tempat Anda akan menyimpan `.zip` file yang Anda buat untuk fungsi Lambda Anda.
- Anda harus menggunakan peran CodePipeline layanan yang ada yang telah diperbarui dengan izin untuk tindakan ini. Untuk memperbarui peran layanan Anda, lihatIzin kebijakan peran layanan untuk tindakan penerapan Lambda.

Setelah Anda memenuhi prasyarat ini, Anda dapat melanjutkan dengan tutorial dan membuat pipeline CD Anda.

Langkah 1: Buat fungsi Lambda sampel Anda

Pada langkah ini, Anda membuat fungsi Lambda tempat Anda akan menyebarkan.

Untuk membuat fungsi Lambda Anda

1. Akses konsol Lambda dan ikuti langkah-langkah dalam tutorial berikut untuk membuat contoh fungsi Lambda: tautan.
2. Dari navigasi atas, pilih Buat, dan pilih Mulai dari awal dari bagian atas halaman.
3. Di Nama, masukkan **`MyLambdaFunction`**.
4. Publikasikan versi baru. Ini akan menjadi versi yang akan ditunjukkan oleh alias.
   a. Pilih fungsi Anda.
   b. Pilih dropdown Tindakan.
   c. PilihPublikasikan versi baru.
   d. (Opsional) Tambahkan ke deskripsi di Deskripsi.
   e. Pilih Terbitkan.
5. Buat alias untuk fungsi Anda, seperti`aliasV1`.
6. Pastikan alias menunjuk ke versi yang baru saja Anda buat (seperti 1).

> **Note**
>
> Jika Anda memilih $LATEST, Anda tidak dapat menggunakan fitur perpindahan lalu lintas karena Lambda tidak mendukung $LATEST untuk alias yang menunjuk ke lebih dari 1 versi.

Langkah 2: Unggah file fungsi ke repositori Anda

Unduh fungsinya dan simpan sebagai file zip. Unggah file zip ke bucket S3 Anda menggunakan langkah-langkah berikut.

Untuk menambahkan **`.zip`** file ke repositori sumber Anda

1. Buka ember S3 Anda.
2. Pilih Unggah.
3. Unggah file zip yang berisi `sample_lambda_source.zip` file Anda ke bucket sumber Anda.

   Catat jalannya.

   ```
   object key
   ```

Langkah 3: Membuat pipeline Anda

Gunakan CodePipeline wizard untuk membuat tahapan pipeline dan menghubungkan repositori sumber Anda.

Untuk membuat alur Anda

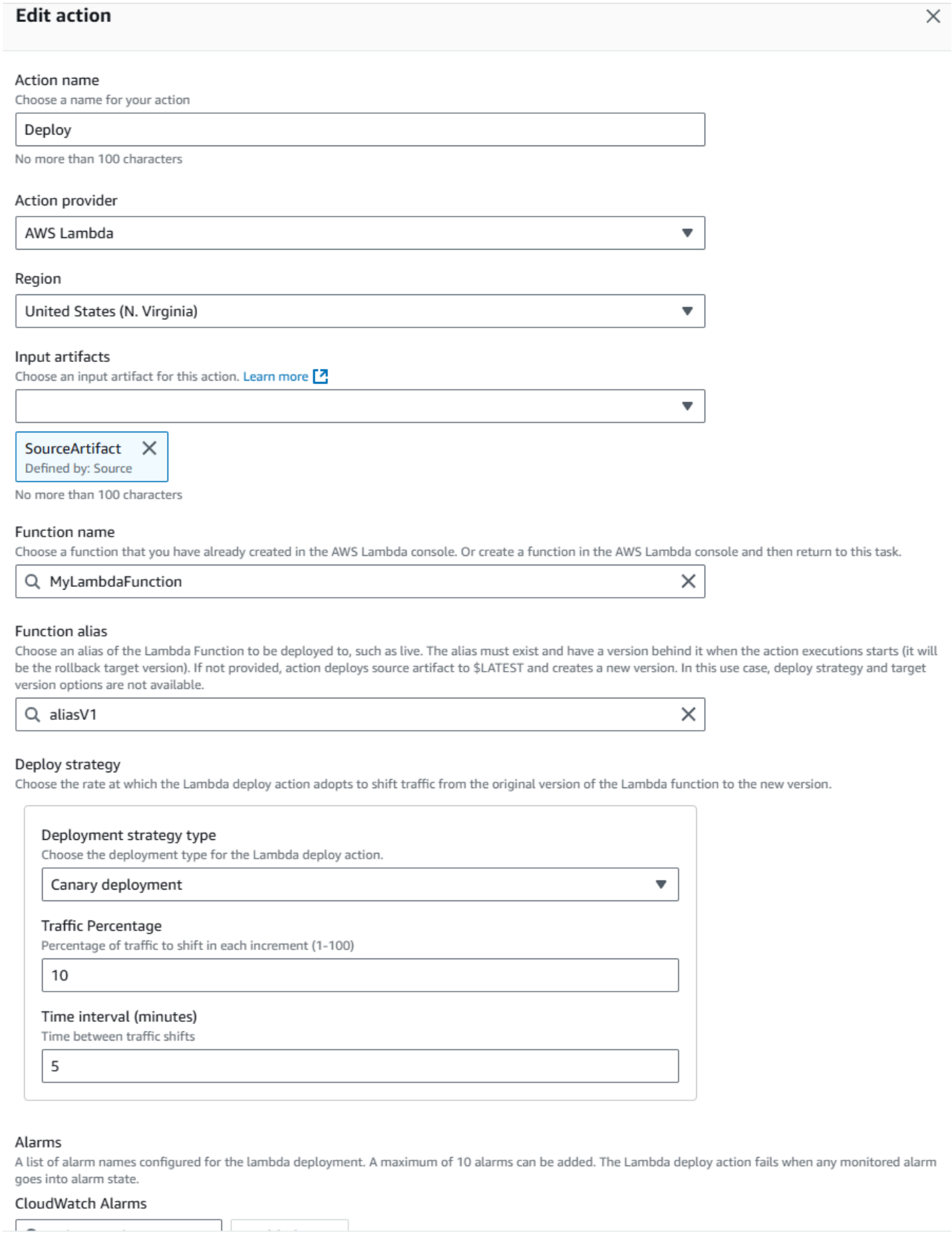
1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyPipeline`**.
5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
6. Di peran Layanan, pilih Gunakan peran layanan yang ada, lalu pilih peran CodePipeline layanan yang telah diperbarui dengan izin yang diperlukan untuk tindakan ini. Untuk mengonfigurasi peran CodePipeline layanan Anda untuk tindakan ini, lihatIzin kebijakan peran layanan untuk tindakan penerapan Lambda.
7. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.
8. Pada Langkah 3: Tambahkan halaman tahap sumber, tambahkan tahap sumber:

   a. Di penyedia Sumber, pilih Amazon S3.

   b. Dalam kunci Object, tambahkan nama file.zip Anda, termasuk ekstensi file, sepertisample_lambda_source.zip.

Pilih Berikutnya.

9. Pada Langkah 4: Tambahkan halaman tahap build, pilih Lewati.
10. Pada Langkah 5: Tambahkan halaman tahap pengujian, pilih Lewati.
11. Pada Langkah 6: Tambahkan halaman tahap penerapan, pilih Lambda.

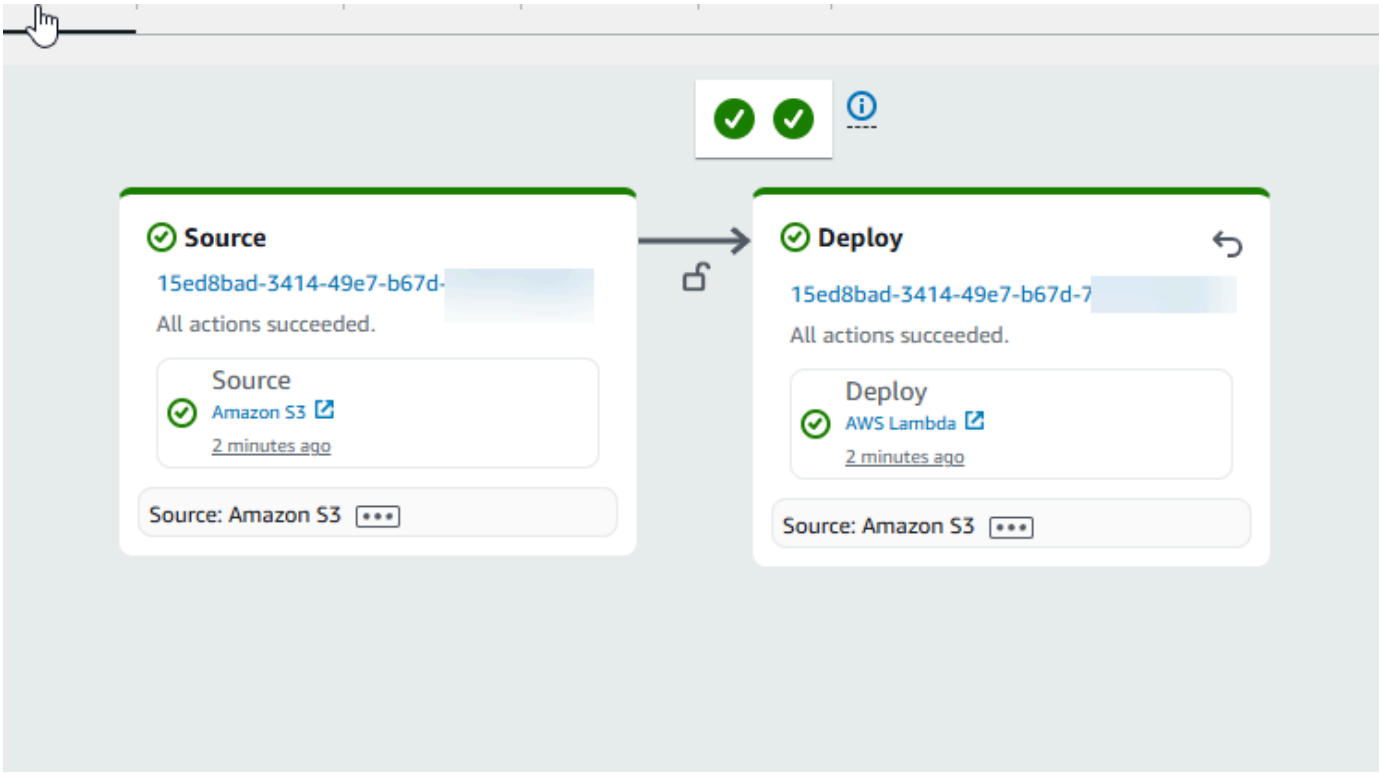

Edit action
Action name
Choose a name for your action
Deploy
No more than 100 characters
Action provider
AWS Lambda
Region
United States (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
Defined by: Source
No more than 100 characters
Function name
Choose 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. Tambahkan nama fungsi dan alias Anda.

b. Pilih strategi penerapan Anda.

c. Pilih Berikutnya.

12. Pada halaman Langkah 7: Tinjau, tinjau konfigurasi pipeline Anda dan pilih Buat pipeline untuk membuat pipeline.



Langkah 4: Uji Pipa Anda

Pipeline Anda harus memiliki segalanya untuk menjalankan penerapan AWS berkelanjutan end-to-end asli. Sekarang, uji fungsinya dengan mendorong perubahan kode ke repositori sumber Anda.

Untuk menguji pipa Anda

1. Buat perubahan kode ke repositori sumber yang dikonfigurasi, komit, dan dorong perubahan.
2. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
3. Pilih pipeline Anda dari daftar.
4. Perhatikan kemajuan pipa melalui tahapannya. Pipeline Anda harus selesai dan tindakan Anda diterapkan ke fungsi Lambda Anda.

Pelajari selengkapnya

Tindakan penyebaran Lambda memungkinkan dua metode penerapan. Salah satu metode adalah pergeseran lalu lintas sendiri tanpa artefak input dari aksi sumber. Metode lainnya adalah memperbarui kode fungsi menggunakan artefak input dari tindakan sumber, kemudian menerbitkan versi baru berdasarkan kode yang diperbarui. Untuk metode kedua, jika alias disediakan, CodePipeline akan melakukan pergeseran lalu lintas juga. Tutorial tindakan penyebaran Lambda ini menunjukkan memperbarui fungsi Anda menggunakan artefak sumber.

Untuk mempelajari lebih lanjut tentang tindakan, lihat halaman referensi tindakan diAWS Lambda menyebarkan referensi tindakan.

Tutorial: Menggunakan variabel dengan tindakan panggilan Lambda

Tindakan pemanggilan Lambda dapat menggunakan variabel dari tindakan lain sebagai bagian dari inputnya dan mengembalikan variabel baru bersama dengan outputnya. Untuk informasi tentang variabel untuk tindakan CodePipeline, lihatReferensi variabel.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Di akhir tutorial ini, Anda akan memiliki:

- Lambda memanggil tindakan yang:
  - Mengkonsumsi `CommitId` variabel dari aksi CodeCommit sumber
  - Menghasilkan tiga variabel baru:`dateTime`,`testRunId`, dan `region`
- Tindakan persetujuan manual yang menggunakan variabel baru dari tindakan pemanggilan Lambda Anda untuk menyediakan URL pengujian dan ID uji coba
- Pipeline diperbarui dengan tindakan baru

Topik

- Prasyarat
- Langkah 1: Membuat fungsi Lambda
- Langkah 2: Tambahkan tindakan pemanggilan Lambda dan tindakan persetujuan manual ke pipeline Anda

Prasyarat

Sebelum memulai, Anda harus memiliki hal-hal berikut:

- Anda dapat membuat atau menggunakan pipeline dengan CodeCommit sumbernyaTutorial: Buat pipeline sederhana (CodeCommit repositori).
- Edit pipeline Anda yang ada sehingga tindakan CodeCommit sumber memiliki namespace. Tetapkan namespace `SourceVariables` untuk tindakan.

Langkah 1: Membuat fungsi Lambda

Gunakan langkah-langkah berikut untuk membuat fungsi Lambda dan peran eksekusi Lambda. Anda menambahkan tindakan Lambda ke pipeline setelah Anda membuat fungsi Lambda.

Untuk membuat fungsi Lambda dan peran eksekusi

1. Masuk ke Konsol Manajemen AWS dan buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pilih Buat fungsi. Biarkan Penulis dari awal dipilih.
3. Dalam nama Fungsi, masukkan nama fungsi Anda, seperti**`myInvokeFunction`**. Di Runtime, biarkan opsi default dipilih.
4. Perluas Pilih atau buat peran eksekusi. Pilih Buat peran baru dengan izin Lambda dasar.
5. Pilih Buat fungsi.
6. Untuk menggunakan variabel dari tindakan lain, itu harus diteruskan ke konfigurasi tindakan `UserParameters` pemanggilan Lambda. Anda akan mengonfigurasi tindakan di pipeline kami nanti di tutorial, tetapi Anda akan menambahkan kode dengan asumsi variabel akan diteruskan.

   Untuk menghasilkan variabel baru, atur properti yang dipanggil `outputVariables` pada input `keputJobSuccessResult`. Perhatikan bahwa Anda tidak dapat menghasilkan variabel sebagai bagian dari `aputJobFailureResult`.

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

Dalam fungsi baru Anda, pada tab Kode, tempel kode contoh berikut di bawah`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. Izinkan fungsi untuk menyimpan secara otomatis.
8. Salin Nama Sumber Daya Amazon (ARN) yang terdapat di bidang Fungsi ARN di bagian atas layar.
9. Sebagai langkah terakhir, buka konsol AWS Identity and Access Management (IAM) di https://console.aws.amazon.com/iam/. Ubah peran eksekusi Lambda untuk menambahkan kebijakan

berikut:. AWSCodePipelineCustomActionAccess Untuk langkah-langkah membuat peran eksekusi Lambda atau mengubah kebijakan peran, lihat. Langkah 2: Buat fungsi Lambda

Langkah 2: Tambahkan tindakan pemanggilan Lambda dan tindakan persetujuan manual ke pipeline Anda

Pada langkah ini, Anda menambahkan tindakan pemanggilan Lambda ke pipeline Anda. Anda menambahkan tindakan sebagai bagian dari tahap bernama Test. Jenis tindakan adalah tindakan pemanggilan. Anda kemudian menambahkan tindakan persetujuan manual setelah tindakan pemanggilan.

Untuk menambahkan tindakan Lambda dan tindakan persetujuan manual ke pipeline

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

   Nama-nama semua saluran pipa yang terkait dengan AWS akun Anda ditampilkan. Pilih pipeline tempat Anda ingin menambahkan tindakan.
2. Tambahkan tindakan pengujian Lambda ke pipeline Anda.

   a. Untuk mengedit pipeline Anda, pilih Edit. Tambahkan tahapan setelah aksi sumber di pipeline yang ada. Masukkan nama untuk panggung, seperti**`Test`**.

   b. Di tahap baru, pilih Tambahkan grup tindakan untuk menambahkan tindakan. Dalam nama Tindakan, masukkan nama tindakan pemanggilan, seperti**`Test_Commit`**.

   c. Di penyedia Action, pilih AWS Lambda.

   d. Di artefak Input, pilih nama artefak keluaran aksi sumber Anda, seperti. `SourceArtifact`

   e. Di FunctionName, tambahkan ARN dari fungsi Lambda yang Anda buat.

   f. Di Namespace Variable, tambahkan nama namespace, seperti. **`TestVariables`**

   g. Dalam artefak Output, tambahkan nama artefak keluaran, seperti. **`LambdaArtifact`**

   h. Pilih Selesai.
3. Tambahkan tindakan persetujuan manual ke pipeline Anda.

   a. Dengan pipeline Anda masih dalam mode pengeditan, tambahkan tahapan setelah tindakan pemanggilan. Masukkan nama untuk panggung, seperti**`Approval`**.

   b. Di tahap baru, pilih ikon untuk menambahkan tindakan. Di Nama tindakan, masukkan nama tindakan persetujuan, seperti**`Change_Approval`**.

c. Di penyedia Tindakan, pilih Persetujuan manual.

d. Di URL untuk ditinjau, buat URL dengan menambahkan sintaks variabel untuk `region` variabel dan variabel. `CommitId` Pastikan Anda menggunakan ruang nama yang ditetapkan untuk tindakan yang menyediakan variabel output.

   Untuk contoh ini, URL dengan sintaks variabel untuk CodeCommit tindakan memiliki `SourceVariables` namespace default. Variabel keluaran wilayah Lambda memiliki namespace. `TestVariables` URL terlihat seperti berikut ini.

```
https://#{TestVariables.region}.console.aws.amazon.com/codesuite/codecommit/
repositories/MyDemoRepo/commit/#{SourceVariables.CommitId}
```

   Di Komentar, buat teks pesan persetujuan dengan menambahkan sintaks variabel untuk variabel. `testRunId` Untuk contoh ini, URL dengan sintaks variabel untuk variabel keluaran `testRunId` Lambda memiliki `TestVariables` namespace. Masukkan pesan berikut.

```
Make sure to review the code before approving this action. Test Run ID:
 #{TestVariables.testRunId}
```

4. Pilih Selesai untuk menutup layar edit untuk tindakan, lalu pilih Selesai untuk menutup layar edit untuk panggung. Untuk menyimpan pipa, pilih Selesai. Pipeline yang telah selesai sekarang berisi struktur dengan tahapan sumber, pengujian, persetujuan, dan penerapan.

   Pilih Rilis perubahan untuk menjalankan perubahan terbaru melalui struktur pipa.

5. Saat pipa mencapai tahap persetujuan manual, pilih Tinjau. Variabel yang diselesaikan muncul sebagai URL untuk ID komit. Penyetuju Anda dapat memilih URL untuk melihat komit.

6. Setelah pipeline berjalan dengan sukses, Anda juga dapat melihat nilai variabel pada halaman riwayat eksekusi tindakan.

Tutorial: Gunakan AWS Step Functions tindakan pemanggilan dalam pipeline

Anda dapat menggunakan AWS Step Functions untuk membuat dan mengkonfigurasi mesin negara. Tutorial ini menunjukkan cara menambahkan tindakan pemanggilan ke pipeline yang mengaktifkan eksekusi mesin status dari pipeline Anda.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Dalam tutorial ini, Anda melakukan tugas-tugas berikut:

- Buat mesin status standar di AWS Step Functions.
- Masukkan input mesin status JSON secara langsung. Anda juga dapat mengunggah file input mesin status ke bucket Amazon Simple Storage Service (Amazon S3).
- Perbarui pipeline Anda dengan menambahkan tindakan mesin status.

Topik

- Prasyarat: Buat atau pilih pipa sederhana
- Langkah 1: Buat mesin status sampel
- Langkah 2: Tambahkan tindakan pemanggilan Step Functions ke pipeline Anda

Prasyarat: Buat atau pilih pipa sederhana

Dalam tutorial ini, Anda menambahkan tindakan pemanggilan ke pipeline yang ada. Anda dapat menggunakan pipeline yang Anda buat di Tutorial: Buat pipeline sederhana (ember S3) atauTutorial: Buat pipeline sederhana (CodeCommit repositori).

Anda menggunakan pipeline yang ada dengan aksi sumber dan setidaknya struktur dua tahap, tetapi Anda tidak menggunakan artefak sumber untuk contoh ini.

> **Note**
>
> Anda mungkin perlu memperbarui peran layanan yang digunakan oleh pipeline Anda dengan izin tambahan yang diperlukan untuk menjalankan tindakan ini. Untuk melakukannya, buka konsol AWS Identity and Access Management (IAM), cari peran, lalu tambahkan

izin ke kebijakan peran. Untuk informasi selengkapnya, lihat Menambahkan izin ke peran CodePipeline layanan.

Langkah 1: Buat mesin status sampel

Di konsol Step Functions, buat mesin status menggunakan template `HelloWorld` sampel. Untuk petunjuk, lihat Membuat Mesin Status di Panduan AWS Step Functions Pengembang.

Langkah 2: Tambahkan tindakan pemanggilan Step Functions ke pipeline Anda

Tambahkan tindakan pemanggilan Step Functions ke pipeline Anda sebagai berikut:

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda edit. Ini membuka tampilan rinci dari pipa, termasuk keadaan masing-masing tindakan di setiap tahap pipa.
3. Pada halaman detail pipeline, pilih Edit.
4. Pada tahap kedua dari pipeline sederhana Anda, pilih Edit tahap. Pilih Hapus. Ini menghapus tahap kedua sekarang karena Anda tidak lagi membutuhkannya.
5. Di bagian bawah diagram, pilih + Tambahkan tahap.
6. Di Nama panggung, masukkan nama untuk panggung, seperti**`Invoke`**, lalu pilih Tambah tahap.
7. Pilih + Tambahkan grup tindakan.
8. Dalam nama Action, masukkan nama, seperti**`Invoke`**.
9. Di penyedia Action, pilih AWS Step Functions. Izinkan Wilayah ke default ke Wilayah alur.
10. Di artefak Input, pilih`SourceArtifact`.
11. Di mesin ARN negara, pilih Nama Sumber Daya Amazon (ARN) untuk mesin status yang Anda buat sebelumnya.
12. (Opsional) Dalam awalan nama Eksekusi, masukkan awalan yang akan ditambahkan ke ID eksekusi mesin negara.
13. Di tipe Input, pilih Literal.

14. Di Input, masukkan input JSON yang diharapkan oleh mesin status `HelloWorld` sampel.

    > **Note**
    >
    > Input untuk eksekusi mesin keadaan berbeda dari istilah yang digunakan CodePipeline untuk menggambarkan artefak input untuk tindakan.

    Untuk contoh ini, masukkan JSON berikut:

    ```
    {"IsHelloWorldExample": true}
    ```

15. Pilih Selesai.
16. Di panggung yang Anda edit, pilih Selesai. Di AWS CodePipeline panel, pilih Simpan, lalu pilih Simpan pada pesan peringatan.
17. Untuk mengirimkan perubahan dan memulai eksekusi pipeline, pilih Rilis perubahan, lalu pilih Rilis.
18. Pada pipeline yang telah selesai, pilih AWS Step Functions dalam tindakan pemanggilan Anda. Di AWS Step Functions konsol, lihat ID eksekusi mesin status Anda. ID menunjukkan nama mesin status Anda `HelloWorld` dan ID eksekusi mesin status dengan awalan`my-prefix`.

    ```
    arn:aws:states:us-west-2:account-ID:execution:HelloWorld:my-
    prefix-0d9a0900-3609-4ebc-925e-83d9618fcca1
    ```

Tutorial: Buat pipeline yang digunakan AWS AppConfig sebagai penyedia penyebaran

Dalam tutorial ini, Anda mengonfigurasi pipeline yang terus-menerus mengirimkan file konfigurasi menggunakan AWS AppConfig sebagai penyedia tindakan penerapan di tahap penerapan Anda.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk

pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Topik

- Prasyarat
- Langkah 1: Buat AWS AppConfig sumber daya Anda
- Langkah 2: Unggah file ke bucket sumber S3 Anda
- Langkah 3: Buat pipeline Anda
- Langkah 4: Buat perubahan pada file sumber apa pun dan verifikasi penyebaran

Prasyarat

Sebelum Anda mulai, Anda harus menyelesaikan yang berikut:

- Contoh ini menggunakan sumber S3 untuk pipeline Anda. Buat atau gunakan bucket Amazon S3 dengan versi diaktifkan. Ikuti instruksi Langkah 1: Buat bucket sumber S3 untuk aplikasi Anda untuk membuat ember S3.

Langkah 1: Buat AWS AppConfig sumber daya Anda

Di bagian ini, Anda membuat sumber daya berikut:

- Aplikasi dalam AWS AppConfig adalah unit kode logis yang menyediakan kemampuan untuk pelanggan Anda.
- Environment in AWS AppConfig adalah kelompok AppConfig target penyebaran logis, seperti aplikasi dalam lingkungan beta atau produksi.
- Profil konfigurasi adalah kumpulan pengaturan yang memengaruhi perilaku aplikasi Anda. Profil konfigurasi memungkinkan AWS AppConfig untuk mengakses konfigurasi Anda di lokasi yang disimpan.
- (Opsional) Strategi penerapan dalam AWS AppConfig mendefinisikan perilaku penerapan konfigurasi, seperti berapa persentase klien yang harus menerima konfigurasi baru yang diterapkan pada waktu tertentu selama penerapan.

Untuk membuat aplikasi, lingkungan, profil konfigurasi, dan strategi penyebaran

1. Masuk ke Konsol Manajemen AWS.
2. Gunakan langkah-langkah dalam topik berikut untuk membuat sumber daya Anda AWS AppConfig.
   - Buat aplikasi.
   - Buat lingkungan.
   - Buat profil AWS CodePipeline konfigurasi.
   - (Opsional) Pilih strategi penerapan yang telah ditentukan sebelumnya atau buat sendiri.

Langkah 2: Unggah file ke bucket sumber S3 Anda

Di bagian ini, buat file atau file konfigurasi Anda. Kemudian zip dan dorong file sumber Anda ke bucket yang digunakan pipeline untuk tahap sumber Anda.

Untuk membuat file konfigurasi

1. Buat `configuration.json` file untuk setiap konfigurasi di setiap Wilayah. Sertakan konten berikut:

   ```
   Hello World!
   ```

2. Gunakan langkah-langkah berikut untuk zip dan mengunggah file konfigurasi Anda.

Untuk zip dan mengunggah file sumber

1. Buat file.zip dengan file Anda dan beri nama file.zip. `configuration-files.zip` Sebagai contoh, file.zip Anda dapat menggunakan struktur berikut:

   ```
   .
   ### appconfig-configurations
       ### MyConfigurations
           ### us-east-1
           #   ### configuration.json
           ### us-west-2
               ### configuration.json
   ```

2. Di konsol Amazon S3 untuk bucket Anda, pilih Unggah, dan ikuti petunjuk untuk mengunggah file.zip Anda.

Langkah 3: Buat pipeline Anda

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan aksi Amazon S3 di mana artefak sumber adalah file untuk konfigurasi Anda.
- Tahap penyebaran dengan tindakan AppConfig penerapan.

Untuk membuat alur dengan wizard

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Pipelines, pilih Buat pipeline.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyAppConfigPipeline`**.
5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
6. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.
7. Biarkan pengaturan di bawah Pengaturan lanjutan pada defaultnya, lalu pilih Berikutnya.
8. Pada Langkah 3: Tambahkan tahap sumber, di penyedia Sumber, pilih Amazon S3. Di Bucket, pilih nama bucket sumber S3 Anda.

   Di tombol objek S3, masukkan nama file.zip Anda:. `configuration-files.zip`

   Pilih Berikutnya.
9. Pada Langkah 4: Tambahkan tahap build, pilih Lewati tahap build, lalu terima pesan peringatan dengan memilih Lewati lagi.

   Pilih Berikutnya.

10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

11. Pada Langkah 6: Tambahkan tahap penerapan:

    a. Di Penyedia Deploy, pilih AWS AppConfig.

    b. Di Aplikasi, pilih nama aplikasi yang Anda buat AWS AppConfig. Bidang menunjukkan ID untuk aplikasi Anda.

    c. Di Lingkungan, pilih nama lingkungan yang Anda buat AWS AppConfig. Bidang menunjukkan ID untuk lingkungan Anda.

    d. Di Profil konfigurasi, pilih nama profil konfigurasi yang Anda buat AWS AppConfig. Bidang menunjukkan ID untuk profil konfigurasi Anda.

    e. Dalam strategi Deployment, pilih nama strategi penyebaran Anda. Ini bisa berupa strategi penerapan yang Anda buat AppConfig atau yang telah Anda pilih dari strategi penerapan yang telah ditentukan sebelumnya. AppConfig Bidang menunjukkan ID untuk strategi penerapan Anda.

    f. Di jalur konfigurasi artefak input, masukkan jalur file. Pastikan jalur konfigurasi artefak masukan Anda cocok dengan struktur direktori dalam file bucket S3. zip Anda. Untuk contoh ini, masukkan path file berikut:`appconfig-configurations/MyConfigurations/us-west-2/configuration.json`.

    g. Pilih Berikutnya.

12. Pada Langkah 7: Tinjau, tinjau informasi, lalu pilih Buat pipeline.

Langkah 4: Buat perubahan pada file sumber apa pun dan verifikasi penyebaran

Buat perubahan pada file sumber Anda dan unggah perubahan ke bucket Anda. Ini memicu pipeline Anda untuk berjalan. Verifikasi bahwa konfigurasi Anda tersedia dengan melihat versinya.

Tutorial: Gunakan klon lengkap dengan sumber GitHub pipa

Anda dapat memilih opsi klon lengkap untuk tindakan GitHub sumber Anda di CodePipeline. Gunakan opsi ini untuk menjalankan CodeBuild perintah untuk metadata Git dalam tindakan pembuatan pipeline Anda.

> **Note**
>
> Opsi klon lengkap yang dijelaskan di sini mengacu pada menentukan apakah CodePipeline harus mengkloning metadata repositori, yang hanya dapat digunakan oleh perintah. CodeBuild Untuk menggunakan token akses GitHub pengguna untuk digunakan dengan CodeBuild proyek, ikuti langkah-langkah di sini untuk menginstal AWS Konektor untuk GitHub aplikasi dan kemudian biarkan bidang Instalasi aplikasi kosong. CodeConnections akan menggunakan token akses pengguna untuk koneksi.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Dalam tutorial ini, Anda akan membuat pipeline yang terhubung ke GitHub repositori Anda, menggunakan opsi klon lengkap untuk data sumber, dan menjalankan CodeBuild build yang mengkloning repositori Anda dan melakukan perintah Git untuk repositori.

> **Note**
>
> Fitur ini tidak tersedia di Asia Pasifik (Hong Kong), Afrika (Cape Town), Timur Tengah (Bahrain), Eropa (Zurich), atau AWS GovCloud (AS-Barat). Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline. Untuk pertimbangan dengan tindakan ini di Wilayah Eropa (Milan), lihat catatan diCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Topik

- Prasyarat
- Langkah 1: Buat file README

- Langkah 2: Buat pipeline Anda dan bangun proyek
- Langkah 3: Perbarui kebijakan peran CodeBuild layanan untuk menggunakan koneksi
- Langkah 4: Lihat perintah repositori dalam output build

Prasyarat

Sebelum Anda mulai, Anda harus melakukan hal berikut:

- Buat GitHub repositori dengan akun Anda GitHub .
- Siapkan GitHub kredensialmu. Ketika Anda menggunakan Konsol Manajemen AWS untuk mengatur koneksi, Anda diminta untuk masuk dengan GitHub kredensi Anda.

Langkah 1: Buat file README

Setelah Anda membuat GitHub repositori Anda, gunakan langkah-langkah ini untuk menambahkan file README.

1. Masuk ke repositori Anda dan pilih GitHub repositori Anda.
2. Untuk membuat file baru, pilih Tambah file > Buat file baru. Beri nama file`README.md`. file dan tambahkan teks berikut.

   ```
   This is a GitHub repository!
   ```

3. Pilih Perubahan commit.

   Pastikan file `README.md` berada di tingkat root repositori Anda..

Langkah 2: Buat pipeline Anda dan bangun proyek

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan koneksi ke GitHub repositori dan tindakan Anda.
- Tahap build dengan aksi AWS CodeBuild build.

Untuk membuat alur dengan wizard

1. Masuk ke CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Alur, pilih Buat alur.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyGitHubPipeline`**.
5. Dalam tipe Pipeline, pilih V1 untuk keperluan tutorial ini. Anda juga dapat memilih V2; Namun, perhatikan bahwa jenis pipa berbeda dalam karakteristik dan harga. Untuk informasi selengkapnya, lihat Jenis pipa.
6. Di Peran layanan, pilih Peran layanan baru.

   > **Note**
   >
   > Jika Anda memilih untuk menggunakan peran CodePipeline layanan yang ada, pastikan Anda telah menambahkan izin `codestar-connections:UseConnection` IAM ke kebijakan peran layanan Anda. Untuk petunjuk tentang peran CodePipeline layanan, lihat Menambahkan izin ke peran CodePipeline layanan.

7. Di bawah Pengaturan lanjutan, biarkan default. Di Penyimpanan artifact, pilih Lokasi default untuk menggunakan penyimpanan artifact default, seperti bucket artifact Amazon S3 yang ditetapkan sebagai default, untuk alur Anda di Wilayah yang Anda pilih untuk alur Anda.

   > **Note**
   >
   > Ini bukan bucket sumber untuk kode sumber Anda. Ini adalah penyimpanan artifact untuk alur Anda. Penyimpanan artifact terpisah, seperti bucket S3, diperlukan untuk setiap alur.

   Pilih Berikutnya.
8. Pada Langkah 3: Tambahkan halaman tahap sumber, tambahkan tahap sumber:

   a. Di penyedia Sumber, pilih GitHub (melalui GitHub Aplikasi).

   b. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.

      Anda menginstal satu aplikasi untuk semua koneksi Anda ke penyedia tertentu. Jika Anda telah menginstal AWS Connector for GitHub app, pilih dan lewati langkah ini.

> **Note**
>
> Jika Anda ingin membuat token akses pengguna, pastikan Anda telah menginstal AWS Connector for GitHub app lalu biarkan bidang Instalasi aplikasi kosong. CodeConnections akan menggunakan token akses pengguna untuk koneksi. Untuk informasi selengkapnya, lihat Mengakses penyedia sumber Anda di CodeBuild.

c. Dalam nama Repositori, pilih nama repositori Anda GitHub.

d. Dalam nama Branch, pilih cabang repositori yang ingin Anda gunakan.

e. Pastikan opsi Mulai pipeline pada perubahan kode sumber dipilih.

f. Di bawah Format artefak keluaran, pilih Klon penuh untuk mengaktifkan opsi klon Git untuk repositori sumber. Hanya tindakan yang disediakan oleh yang CodeBuild dapat menggunakan opsi klon Git. Anda akan menggunakan Langkah 3: Perbarui kebijakan peran CodeBuild layanan untuk menggunakan koneksi dalam tutorial ini untuk memperbarui izin untuk peran layanan CodeBuild proyek Anda untuk menggunakan opsi ini.

Pilih Berikutnya.

9. Pada Langkah 4: Tambahkan tahap build, tambahkan tahap build:

a. Di Penyedia pembangunan, pilih AWS CodeBuild. Izinkan Wilayah ke default ke Wilayah alur.

b. Pilih Buat proyek.

c. Di Nama proyek, masukkan nama untuk proyek pembangunan ini.

d. Di Citra lingkungan, pilih Citra terkelola. Untuk Sistem operasi, pilih Ubuntu.

e. Untuk Waktu aktif, pilih Standar. Untuk Gambar, pilih:5.0aws/codebuild/standard.

f. Untuk Peran layanan, pilih Peran layanan baru.

> **Note**
>
> Perhatikan nama peran CodeBuild layanan Anda. Anda akan membutuhkan nama peran untuk langkah terakhir dalam tutorial ini.

g. Pada Buildspec, untuk Spesifikasi membangun, pilih Sisipkan perintah membangun. Pilih Beralih ke editor, dan tempel yang berikut ini di bawah perintah Build.

> **Note**
>
> Di `env` bagian spesifikasi build, pastikan credential helper untuk perintah git diaktifkan seperti yang ditunjukkan dalam contoh ini.

```
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. Pilih Lanjutkan ke CodePipeline. Ini kembali ke CodePipeline konsol dan membuat CodeBuild proyek yang menggunakan perintah build Anda untuk konfigurasi. Proyek build menggunakan peran layanan untuk mengelola Layanan AWS izin. Langkah ini mungkin memakan waktu beberapa menit.

i. Pilih Berikutnya.

10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

11. Pada Langkah 6: Tambahkan halaman tahap penerapan, pilih Lewati tahap penerapan, lalu terima pesan peringatan dengan memilih Lewati lagi. Pilih Berikutnya.

12. Pada Langkah 7: Tinjau, pilih Buat pipeline.

Langkah 3: Perbarui kebijakan peran CodeBuild layanan untuk menggunakan koneksi

Proses pipeline awal akan gagal karena peran CodeBuild layanan harus diperbarui dengan izin untuk menggunakan koneksi. Tambahkan izin `codestar-connections:UseConnection` IAM ke kebijakan peran layanan Anda. Untuk petunjuk memperbarui kebijakan di konsol IAM, lihat[Tambahkan CodeBuild GitClone izin untuk koneksi ke Bitbucket,, GitHub Enterprise Server GitHub, atau .com GitLab](#).

Langkah 4: Lihat perintah repositori dalam output build

1. Jika peran layanan Anda berhasil diperbarui, pilih Coba lagi pada CodeBuild tahap yang gagal.
2. Setelah pipeline berjalan dengan sukses, pada tahap build yang berhasil, pilih Lihat detail.

   Pada halaman detail, pilih tab Log. Lihat output CodeBuild build. Perintah menampilkan nilai variabel yang dimasukkan.

   Perintah menampilkan isi `README.md` file, daftar file dalam direktori, mengkloning repositori, melihat log, dan mengarsipkan repositori sebagai file ZIP.

Tutorial: Gunakan klon lengkap dengan sumber CodeCommit pipa

Anda dapat memilih opsi klon lengkap untuk tindakan CodeCommit sumber Anda di CodePipeline. Gunakan opsi ini CodeBuild untuk mengizinkan akses metadata Git dalam tindakan pembuatan pipeline Anda.

Dalam tutorial ini, Anda membuat pipeline yang mengakses CodeCommit repositori Anda, menggunakan opsi klon lengkap untuk data sumber, dan menjalankan CodeBuild build yang mengkloning repositori Anda dan melakukan perintah Git untuk repositori.

> **Note**
>
> CodeBuild action adalah satu-satunya tindakan hilir yang mendukung penggunaan metadata Git yang tersedia dengan opsi klon Git. Selain itu, meskipun pipeline Anda dapat berisi tindakan lintas akun, CodeCommit tindakan dan CodeBuild tindakan harus berada di akun yang sama agar opsi klon lengkap berhasil.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Topik

- Prasyarat
- Langkah 1: Buat file README
- Langkah 2: Buat pipeline Anda dan bangun proyek
- Langkah 3: Perbarui kebijakan peran CodeBuild layanan untuk mengkloning repositori
- Langkah 4: Lihat perintah repositori dalam output build

Prasyarat

Sebelum memulai, Anda harus membuat CodeCommit repositori di AWS akun dan Wilayah yang sama dengan pipeline Anda.

Langkah 1: Buat file README

Gunakan langkah-langkah ini untuk menambahkan file README ke repositori sumber Anda. File README menyediakan file sumber contoh untuk tindakan CodeBuild hilir untuk dibaca.

Untuk menambahkan file README

1. Masuk ke repositori Anda dan pilih repositori Anda.
2. Untuk membuat file baru, pilih Tambah file > Buat file. Beri nama file`README.md`. file dan tambahkan teks berikut.

   ```
   This is a CodeCommit repository!
   ```
3. Pilih Perubahan commit.

   Pastikan file `README.md` berada di tingkat root repositori Anda..

Langkah 2: Buat pipeline Anda dan bangun proyek

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan aksi CodeCommit sumber.
- Tahap build dengan aksi AWS CodeBuild build.

Untuk membuat alur dengan wizard

1. Masuk ke CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Alur, pilih Buat alur.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyCodeCommitPipeline`**.

5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
6. Dalam peran Layanan, lakukan salah satu hal berikut:
   - Pilih Peran layanan yang ada.
   - Pilih peran CodePipeline layanan Anda yang ada. Peran ini harus memiliki izin `codecommit:GetRepository` IAM untuk kebijakan peran layanan Anda. Lihat Menambahkan izin ke peran CodePipeline layanan.
7. Di bawah Pengaturan lanjutan, biarkan default. Pilih Berikutnya.
8. Pada Langkah 3: Tambahkan halaman tahap sumber, lakukan hal berikut:
   a. Di penyedia Sumber, pilih CodeCommit.
   b. Dalam nama Repositori, pilih nama repositori Anda.
   c. Di Nama cabang, pilih nama cabang Anda.
   d. Pastikan opsi Mulai pipeline pada perubahan kode sumber dipilih.
   e. Di bawah Format artefak keluaran, pilih Klon penuh untuk mengaktifkan opsi klon Git untuk repositori sumber. Hanya tindakan yang disediakan oleh yang CodeBuild dapat menggunakan opsi klon Git.

   Pilih Berikutnya.
9. Pada Langkah 4: Tambahkan tahap build, lakukan hal berikut:
   a. Di Penyedia pembangunan, pilih AWS CodeBuild. Izinkan Wilayah ke default ke Wilayah alur.
   b. Pilih Buat proyek.
   c. Di Nama proyek, masukkan nama untuk proyek pembangunan ini.
   d. Di Citra lingkungan, pilih Citra terkelola. Untuk Sistem operasi, pilih Ubuntu.
   e. Untuk Waktu aktif, pilih Standar. Untuk Gambar, pilih:5.0aws/codebuild/standard.
   f. Untuk Peran layanan, pilih Peran layanan baru.

> **Note**
>
> Perhatikan nama peran CodeBuild layanan Anda. Anda akan membutuhkan nama peran untuk langkah terakhir dalam tutorial ini.

g. Pada Buildspec, untuk Spesifikasi membangun, pilih Sisipkan perintah membangun. Pilih Beralih ke editor, lalu di bawah perintah Build paste kode berikut.

```
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. Pilih Lanjutkan ke CodePipeline. Ini mengembalikan Anda ke CodePipeline konsol dan membuat CodeBuild proyek yang menggunakan perintah build untuk konfigurasi. Proyek build menggunakan peran layanan untuk mengelola Layanan AWS izin. Langkah ini mungkin memakan waktu beberapa menit.

i. Pilih Berikutnya.

10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

11. Pada Langkah 6: Tambahkan halaman tahap penerapan, pilih Lewati tahap penerapan, lalu terima pesan peringatan dengan memilih Lewati lagi. Pilih Berikutnya.
12. Pada Langkah 7: Tinjau, pilih Buat pipeline.

Langkah 3: Perbarui kebijakan peran CodeBuild layanan untuk mengkloning repositori

Proses pipeline awal akan gagal karena Anda perlu memperbarui peran CodeBuild layanan dengan izin untuk menarik dari repositori Anda.

Tambahkan izin `codecommit:GitPull` IAM ke kebijakan peran layanan Anda. Untuk petunjuk memperbarui kebijakan di konsol IAM, lihatTambahkan CodeBuild GitClone izin untuk tindakan CodeCommit sumber.

Langkah 4: Lihat perintah repositori dalam output build

Untuk melihat output build

1. Jika peran layanan Anda berhasil diperbarui, pilih Coba lagi pada CodeBuild tahap yang gagal.
2. Setelah pipeline berjalan dengan sukses, pada tahap build yang berhasil, pilih Lihat detail.

   Pada halaman detail, pilih tab Log. Lihat output CodeBuild build. Perintah menampilkan nilai variabel yang dimasukkan.

Perintah menampilkan isi `README.md` file, daftar file dalam direktori, mengkloning repositori, melihat log, dan menjalankan. `git describe --all`

Tutorial: Buat pipeline dengan AWS CloudFormation StackSets tindakan penerapan

Dalam tutorial ini, Anda menggunakan AWS CodePipeline konsol untuk membuat pipeline dengan tindakan penerapan untuk membuat kumpulan tumpukan dan membuat instance tumpukan. Saat pipeline berjalan, template membuat kumpulan tumpukan dan juga membuat dan memperbarui instance tempat set tumpukan digunakan.

> **Important**
>
> Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Ada dua cara untuk mengelola izin untuk kumpulan tumpukan: peran IAM yang dikelola sendiri dan AWS-dikelola. Tutorial ini memberikan contoh dengan izin yang dikelola sendiri.

Untuk menggunakan Stacksets secara efektif CodePipeline, Anda harus memiliki pemahaman yang jelas tentang konsep di balik CloudFormation StackSets dan cara kerjanya. Lihat StackSets konsep di Panduan AWS CloudFormation Pengguna.

Topik

- Prasyarat
- Langkah 1: Unggah AWS CloudFormation template sampel dan file parameter
- Langkah 2: Buat alur Anda
- Langkah 3: Lihat penyebaran awal
- Langkah 4: Tambahkan CloudFormationStackInstances tindakan
- Langkah 5: Lihat sumber daya set tumpukan untuk penerapan Anda
- Langkah 6: Buat pembaruan ke set tumpukan Anda

Prasyarat

Untuk operasi set tumpukan, Anda menggunakan dua akun berbeda: akun administrasi dan akun target. Anda membuat set tumpukan di akun administrator. Anda membuat tumpukan individual milik tumpukan yang ditetapkan di akun target.

Untuk membuat peran administrator dengan akun administrator Anda

- Ikuti petunjuk di Siapkan izin dasar untuk operasi set tumpukan. Peran Anda harus diberi nama **`AWSCloudFormationStackSetAdministrationRole`**.

Untuk membuat peran layanan di akun target

- Buat peran layanan di akun target yang mempercayai akun administrator. Ikuti petunjuk di Siapkan izin dasar untuk operasi set tumpukan. Peran Anda harus diberi nama **`AWSCloudFormationStackSetExecutionRole`**.

Langkah 1: Unggah AWS CloudFormation template sampel dan file parameter

Buat bucket sumber untuk template set tumpukan dan file parameter Anda. Unduh file AWS CloudFormation templat sampel, atur file parameter, lalu zip file sebelum diunggah ke bucket sumber S3 Anda.

> **Note**
>
> Pastikan untuk ZIP file sumber sebelum Anda mengunggah ke bucket sumber S3 Anda, meskipun satu-satunya file sumber adalah template.

Untuk membuat bucket sumber S3

1. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon S3 di. https://console.aws.amazon.com/s3/
2. Pilih Buat bucket.
3. Dalam nama Bucket, masukkan nama untuk bucket Anda.

Di Wilayah, pilih Wilayah tempat Anda ingin membuat pipeline. Pilih Buat bucket.

4. Setelah ember dibuat, spanduk sukses ditampilkan. Pilih Buka detail ember.
5. Pada tab Properties, pilih Versioning. Pilih Aktifkan pembuatan versi, lalu pilih Simpan.

Untuk membuat file AWS CloudFormation template

1. Unduh contoh file template berikut untuk menghasilkan CloudTrail konfigurasi untuk set tumpukan:https://s3.amazonaws.com/cloudformation-stackset-sample-templates-us-east-1/EnableAWSCloudtrail.yml.
2. Simpan file sebagai `template.yml`.

Untuk membuat file parameters.txt

1. Buat file dengan parameter untuk penyebaran Anda. Parameter adalah nilai yang ingin Anda perbarui di tumpukan Anda saat runtime. File contoh berikut memperbarui parameter template untuk kumpulan tumpukan Anda untuk mengaktifkan validasi logging dan peristiwa global.

```
[
  {
    "ParameterKey": "EnableLogFileValidation",
    "ParameterValue": "true"
  },
  {
    "ParameterKey": "IncludeGlobalEvents",
    "ParameterValue": "true"
  }
]
```

2. Simpan file sebagai `parameters.txt`.

Untuk membuat file accounts.txt

1. Buat file dengan akun tempat Anda ingin membuat instance, seperti yang ditunjukkan pada file contoh berikut.

```
[
    "111111222222","333333444444"
```

```
]
```

2. Simpan file sebagai `accounts.txt`.

Untuk membuat dan mengunggah file sumber

1. Gabungkan file menjadi satu file ZIP. File Anda akan terlihat seperti ini di file ZIP Anda.

```
template.yml
parameters.txt
accounts.txt
```

2. Unggah file ZIP ke bucket S3 Anda. File ini adalah artefak sumber yang dibuat oleh wizard Create Pipeline untuk tindakan penerapan Anda. CodePipeline

Langkah 2: Buat alur Anda

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan aksi sumber S3 di mana artefak sumber adalah file template Anda dan file sumber pendukung apa pun.
- Tahap penerapan dengan aksi penyebaran set CloudFormation tumpukan yang membuat kumpulan tumpukan.
- Tahap penerapan dengan aksi penerapan instance CloudFormation tumpukan yang membuat tumpukan dan instance di dalam akun target.

Untuk membuat pipeline dengan CloudFormationStackSet tindakan

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Alur, pilih Buat alur.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyStackSetsPipeline`**.

5. Dalam tipe Pipeline, pilih V1 untuk keperluan tutorial ini. Anda juga dapat memilih V2; Namun, perhatikan bahwa jenis pipa berbeda dalam karakteristik dan harga. Untuk informasi selengkapnya, lihat Jenis pipa.
6. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan di IAM.
7. Di toko Artifact, tinggalkan default.

> **Note**
>
> Ini bukan bucket sumber untuk kode sumber Anda. Ini adalah penyimpanan artifact untuk alur Anda. Penyimpanan artifact terpisah, seperti bucket S3, diperlukan untuk setiap alur. Saat membuat atau mengedit pipeline, Anda harus memiliki bucket artefak di Wilayah pipeline dan satu bucket artefak per AWS Wilayah tempat Anda menjalankan tindakan. Untuk informasi selengkapnya, lihat Artefak input dan output dan CodePipeline referensi struktur pipa.

   Pilih Berikutnya.
8. Pada Langkah 3: Tambahkan halaman tahap sumber, di penyedia Sumber, pilih Amazon S3.
9. Di Bucket, masukkan bucket sumber S3 yang Anda buat untuk tutorial ini, seperti`BucketName`. Dalam kunci objek S3, masukkan path file dan nama file untuk file ZIP Anda, seperti`MyFiles.zip`.
10. Pilih Berikutnya.
11. Pada Langkah 4: Tambahkan tahap build, pilih Lewati tahap build, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.
12. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.
13. Pada Langkah 6: Tambahkan tahap penerapan:

    a. Di penyedia Deploy, pilih AWS CloudFormation Stack Set.

    b. Di Stack set name, masukkan nama untuk stack set. Ini adalah nama kumpulan tumpukan yang dibuat template.

> **Note**
>
> Catat nama set tumpukan Anda. Anda akan menggunakannya saat menambahkan tindakan StackSets penerapan kedua ke pipeline Anda.

c. Di jalur Template, masukkan nama artefak dan jalur file tempat Anda mengunggah file template Anda. Misalnya, masukkan yang berikut ini menggunakan nama `SourceArtifact` artefak sumber default.

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

d. Di target Deployment, masukkan nama artefak dan jalur file tempat Anda mengunggah file akun. Misalnya, masukkan yang berikut ini menggunakan nama `SourceArtifact` artefak sumber default.

```
SourceArtifact::accounts.txt
```

e. Di target Deployment Wilayah AWS, masukkan satu Wilayah untuk penerapan instance tumpukan awal Anda, seperti. `us-east-1`

f. Perluas opsi Deployment. Di Parameter, masukkan nama artefak dan jalur file tempat Anda mengunggah file parameter Anda. Misalnya, masukkan yang berikut ini menggunakan nama `SourceArtifact` artefak sumber default.

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

Untuk memasukkan parameter sebagai input literal daripada jalur file, masukkan yang berikut ini:

```
ParameterKey=EnableLogFileValidation,ParameterValue=true
ParameterKey=IncludeGlobalEvents,ParameterValue=true
```

g. Di Capabilities, pilih CAPABILITY_IAM dan CAPABILITY_NAMED_IAM.

h. Dalam model Izin, pilih SELF_MANAGED.

i. Dalam persentase toleransi kegagalan, masukkan`20`.

j. Dalam persentase bersamaan Max, masukkan`25`.

k. Pilih Berikutnya.

l. Pada Langkah 7: Tinjau, pilih Buat pipeline. Tampilan pipa Anda.

m. Biarkan pipeline Anda berjalan.

Langkah 3: Lihat penyebaran awal

Lihat sumber daya dan status untuk penerapan awal Anda. Setelah memverifikasi penerapan berhasil membuat kumpulan tumpukan Anda, Anda dapat menambahkan tindakan kedua ke tahap Deploy Anda.

Untuk melihat sumber daya

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Di bawah Pipelines, pilih pipeline Anda dan pilih View. Diagram menunjukkan sumber pipa dan tahapan penerapan Anda.
3. Pilih CloudFormation tindakan pada CloudFormationStackSettindakan di pipeline Anda. Templat, sumber daya, dan peristiwa untuk kumpulan tumpukan Anda ditampilkan di CloudFormation konsol.
4. Di panel navigasi kiri, pilih StackSets. Dalam daftar, pilih set tumpukan baru.
5. Pilih tab Stack instance. Verifikasi bahwa satu instance tumpukan untuk setiap akun yang Anda berikan telah dibuat di Wilayah us-east-1. Verifikasi bahwa status untuk setiap instance stack adalah`CURRENT`.

Langkah 4: Tambahkan CloudFormationStackInstances tindakan

Buat tindakan berikutnya di pipeline Anda yang memungkinkan Anda CloudFormation StackSets membuat instance tumpukan yang tersisa.

Untuk membuat tindakan selanjutnya di pipeline Anda

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.

   Di bawah Pipelines, pilih pipeline Anda dan pilih View. Diagram menunjukkan sumber pipa dan tahapan penerapan Anda.
2. Pilih untuk mengedit pipeline. Pipeline ditampilkan dalam mode Edit.
3. Pada tahap Deploy, pilih Edit.
4. Di bawah tindakan penyebaran AWS CloudFormation Stack Set, pilih Tambahkan grup tindakan.

5. Pada halaman Edit tindakan, tambahkan detail tindakan:

   a. Di Nama tindakan, masukkan nama untuk tindakan tersebut.

   b. Di penyedia Action, pilih AWS CloudFormation Stack Instances.

   c. Di bawah Input artefak, pilih SourceArtifact.

   d. Di Stack set name, masukkan nama untuk stack set. Ini adalah nama kumpulan tumpukan yang Anda berikan dalam tindakan pertama.

   e. Di target Deployment, masukkan nama artefak dan jalur file tempat Anda mengunggah file akun. Misalnya, masukkan yang berikut ini menggunakan nama `SourceArtifact` artefak sumber default.

   ```
   SourceArtifact::accounts.txt
   ```

   f. Di target Deployment Wilayah AWS, masukkan Wilayah untuk penerapan instance tumpukan Anda yang tersisa, seperti `us-east-2` dan `eu-central-1` sebagai berikut:

   ```
   us-east2, eu-central-1
   ```

   g. Dalam persentase toleransi kegagalan, masukkan`20`.

   h. Dalam persentase bersamaan Max, masukkan`25`.

   i. Pilih Simpan.

   j. .Secara manual melepaskan perubahan. Pipeline Anda yang diperbarui ditampilkan dengan dua tindakan di tahap Deploy.

Langkah 5: Lihat sumber daya set tumpukan untuk penerapan Anda

Anda dapat melihat sumber daya dan status untuk penerapan set tumpukan Anda.

Untuk melihat sumber daya

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Di bawah Pipelines, pilih pipeline Anda, lalu pilih View. Diagram menunjukkan sumber pipa dan tahapan penerapan Anda.
3. Pilih CloudFormation tindakan pada **`AWS CloudFormation Stack Instances`**tindakan di pipeline Anda. Templat, sumber daya, dan peristiwa untuk kumpulan tumpukan Anda ditampilkan di CloudFormation konsol.

4. Di panel navigasi kiri, pilih StackSets. Dalam daftar, pilih set tumpukan Anda.
5. Pilih tab Stack instance. Verifikasi bahwa semua instance tumpukan yang tersisa untuk setiap akun yang Anda berikan telah dibuat atau diperbarui di Wilayah yang diharapkan. Verifikasi bahwa status untuk setiap instance stack adalah`CURRENT`.

Langkah 6: Buat pembaruan ke set tumpukan Anda

Buat pembaruan ke set tumpukan Anda dan terapkan pembaruan ke instance. Dalam contoh ini, Anda juga membuat perubahan pada target penerapan yang ingin Anda tetapkan untuk pembaruan. Contoh yang bukan bagian dari pembaruan pindah ke status usang.

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Di bawah Pipelines, pilih pipeline Anda dan kemudian pilih Edit. Pada tahap Deploy, pilih Edit.
3. Pilih untuk mengedit tindakan AWS CloudFormation Stack Set di pipeline Anda. Dalam Deskripsi, tulis deskripsi yang ada dengan deskripsi baru untuk kumpulan tumpukan.
4. Pilih untuk mengedit tindakan AWS CloudFormation Stack Instances di pipeline Anda. Di target Deployment Wilayah AWS, hapus `us-east-2` nilai yang dimasukkan saat tindakan dibuat.
5. Simpan perubahan. Pilih Rilis perubahan untuk menjalankan pipeline Anda.
6. Buka tindakan Anda di CloudFormation. Pilih tab StackSet info. Dalam StackSet deskripsi, verifikasi bahwa deskripsi baru ditampilkan.
7. Pilih tab Stack instance. Di bawah Status, verifikasi bahwa status untuk instance tumpukan di us-east-2 adalah. `OUTDATED`

Tutorial: Buat aturan pemeriksaan variabel untuk pipeline sebagai kondisi entri

Dalam tutorial ini, Anda mengonfigurasi pipeline yang terus-menerus mengirimkan file menggunakan GitHub sebagai penyedia tindakan sumber di tahap sumber Anda. Pipeline yang telah selesai mendeteksi perubahan saat Anda membuat perubahan pada file sumber di repositori sumber Anda. Pipeline berjalan dan kemudian memeriksa variabel output terhadap nama repositori sumber dan nama cabang yang disediakan dalam kondisi untuk masuk ke tahap build.

Important

Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Important

Banyak tindakan yang Anda tambahkan ke pipeline dalam prosedur ini melibatkan AWS sumber daya yang perlu Anda buat sebelum membuat pipeline. AWS sumber daya untuk tindakan sumber Anda harus selalu dibuat di AWS Wilayah yang sama tempat Anda membuat pipeline. Misalnya, jika Anda membuat pipeline di Wilayah AS Timur (Ohio), CodeCommit repositori Anda harus berada di Wilayah AS Timur (Ohio).
Anda dapat menambahkan tindakan lintas wilayah saat membuat pipeline. AWS sumber daya untuk tindakan lintas wilayah harus berada di AWS Wilayah yang sama di mana Anda berencana untuk menjalankan tindakan. Untuk informasi selengkapnya, lihat Menambahkan tindakan Lintas wilayah di CodePipeline.

Contoh ini menggunakan pipeline contoh dengan aksi sumber GitHub (Version2) dan aksi CodeBuild build di mana kondisi entri untuk tahap build akan memeriksa variabel.

Prasyarat

Sebelum Anda mulai, Anda harus melakukan hal berikut:

- Buat GitHub repositori dengan akun Anda GitHub .
- Siapkan GitHub kredensialmu. Ketika Anda menggunakan Konsol Manajemen AWS untuk mengatur koneksi, Anda diminta untuk masuk dengan GitHub kredensi Anda.
- Koneksi ke repositori Anda untuk disiapkan GitHub (melalui GitHub Aplikasi) sebagai tindakan sumber untuk pipeline Anda. Untuk membuat koneksi ke GitHub repositori Anda, lihat. GitHub koneksi

Langkah 1: Buat file sumber sampel dan tambahkan ke GitHub repositori Anda

Di bagian ini, Anda membuat dan menambahkan file sumber contoh Anda ke repositori yang digunakan pipeline untuk tahap sumber Anda. Untuk contoh ini, Anda menghasilkan dan menambahkan yang berikut:

- File `README.md`.

Setelah Anda membuat GitHub repositori Anda, gunakan langkah-langkah ini untuk menambahkan file README Anda.

1. Masuk ke repositori Anda dan pilih GitHub repositori Anda.
2. Untuk membuat file baru, pilih Tambah file, lalu pilih Buat file baru. Beri nama file `README.md` dan tambahkan teks berikut.

   ```
   This is a GitHub repository!
   ```
3. Pilih Perubahan commit. Untuk keperluan tutorial ini, tambahkan pesan komit yang berisi kata huruf besar “Update” seperti pada contoh berikut:

   ```
   Update to source files
   ```

   > **Note**
   >
   > Pemeriksaan aturan untuk string peka huruf besar/kecil.

   Pastikan file `README.md` berada di tingkat root repositori Anda..

Langkah 2: Buat alur Anda

Dalam bagian ini, Anda membuat alur dengan tindakan berikut:

- Tahap sumber dengan koneksi ke GitHub repositori dan tindakan Anda.
- Tahap CodeBuild build di mana stage memiliki kondisi On Entry yang dikonfigurasi untuk aturan pemeriksaan variabel.

Untuk membuat alur dengan wizard

1. Masuk ke CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pada halaman Selamat Datang, halaman Memulai, atau halaman Alur, pilih Buat alur.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih pengaturan pipeline, dalam nama Pipeline, masukkan**`MyVarCheckPipeline`**.
5. CodePipeline menyediakan pipa tipe V1 dan V2, yang berbeda dalam karakteristik dan harga. Jenis V2 adalah satu-satunya jenis yang dapat Anda pilih di konsol. Untuk informasi selengkapnya, lihat jenis pipa. Untuk informasi tentang harga CodePipeline, lihat Harga.
6. Di Peran layanan, pilih Peran layanan baru.

   > **Note**
   >
   > Jika Anda memilih untuk menggunakan peran CodePipeline layanan yang ada, pastikan Anda telah menambahkan izin `codeconnections:UseConnection` IAM ke kebijakan peran layanan Anda. Untuk petunjuk tentang peran CodePipeline layanan, lihat Menambahkan izin ke peran CodePipeline layanan.

7. Di bawah Pengaturan lanjutan, biarkan default.

   Pilih Berikutnya.
8. Pada Langkah 3: Tambahkan halaman tahap sumber, tambahkan tahap sumber:

   a. Di penyedia Sumber, pilih GitHub(melalui GitHub Aplikasi).

   b. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.

   c. Dalam nama Repositori, pilih nama repositori Anda GitHub .

   d. Dalam nama Branch, pilih cabang repositori yang ingin Anda gunakan.

   e. Pastikan opsi Tidak ada pemicu dipilih.

   Pilih Berikutnya.
9. Pada Langkah 4: Tambahkan tahap build, tambahkan tahap build:

a. Di Penyedia pembangunan, pilih AWS CodeBuild. Izinkan Wilayah ke default ke Wilayah alur.

b. Pilih Buat proyek.

c. Di Nama proyek, masukkan nama untuk proyek pembangunan ini.

d. Di Citra lingkungan, pilih Citra terkelola. Untuk Sistem operasi, pilih Ubuntu.

e. Untuk Waktu aktif, pilih Standar. Untuk Gambar, pilih:5.0aws/codebuild/standard.

f. Untuk Peran layanan, pilih Peran layanan baru.

> **Note**
>
> Perhatikan nama peran CodeBuild layanan Anda. Anda akan membutuhkan nama peran untuk langkah terakhir dalam tutorial ini.

g. Pada Buildspec, untuk Spesifikasi membangun, pilih Sisipkan perintah membangun. Pilih Beralih ke editor, dan tempel yang berikut ini di bawah perintah Build.

```
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. Pilih Lanjutkan ke CodePipeline. Ini kembali ke CodePipeline konsol dan membuat CodeBuild proyek yang menggunakan perintah build Anda untuk konfigurasi. Proyek build menggunakan peran layanan untuk mengelola Layanan AWS izin. Langkah ini mungkin memakan waktu beberapa menit.

i. Pilih Berikutnya.

10. Pada Langkah 5: Tambahkan tahap pengujian, pilih Lewati tahap pengujian, lalu terima pesan peringatan dengan memilih Lewati lagi.

    Pilih Berikutnya.

11. Pada Langkah 6: Tambahkan halaman tahap penerapan, pilih Lewati tahap penerapan, lalu terima pesan peringatan dengan memilih Lewati lagi. Pilih Berikutnya.

12. Pada Langkah 7: Tinjau, pilih Buat pipeline.

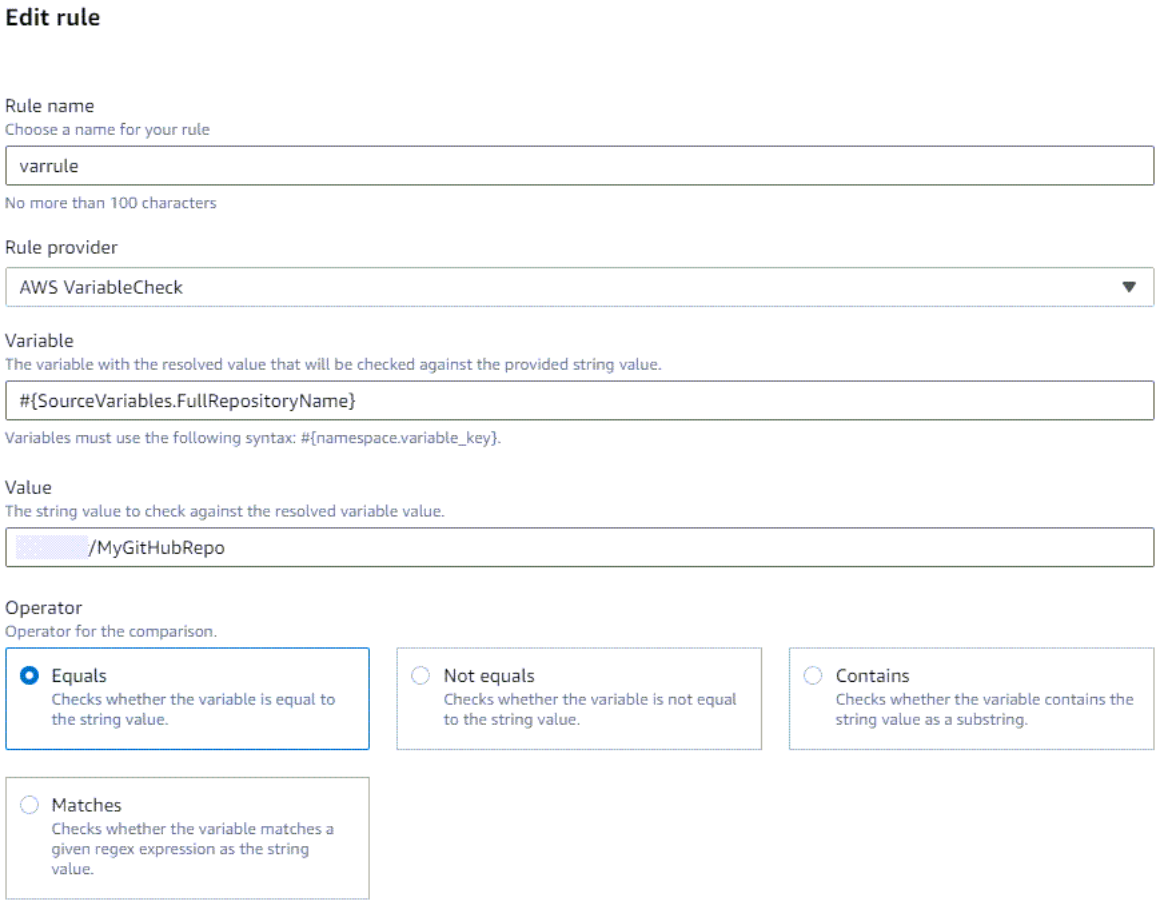
Langkah 2: Edit tahap build untuk menambahkan kondisi dan aturan

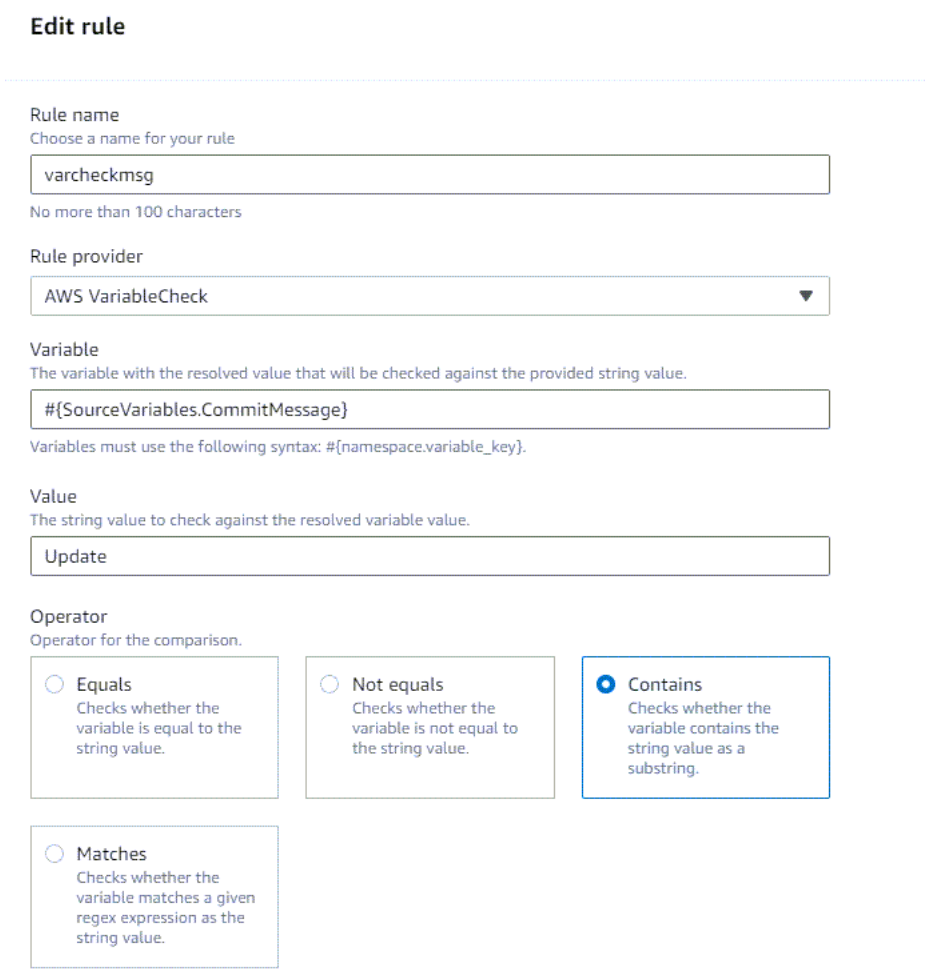
Pada langkah ini, Anda mengedit tahap untuk menambahkan kondisi On Entry untuk aturan pemeriksaan variabel.

1. Pilih pipeline Anda, lalu pilih Edit. Pilih untuk menambahkan aturan entri pada tahap build.

   Di Penyedia aturan, pilih VariableCheck.

2. Di Variable, masukkan variabel atau variabel yang ingin Anda periksa. Di Nilai, masukkan nilai string untuk memeriksa variabel yang diselesaikan. Di layar contoh berikut, aturan dibuat untuk pemeriksaan “sama dengan”, dan aturan lain dibuat untuk pemeriksaan “berisi”.





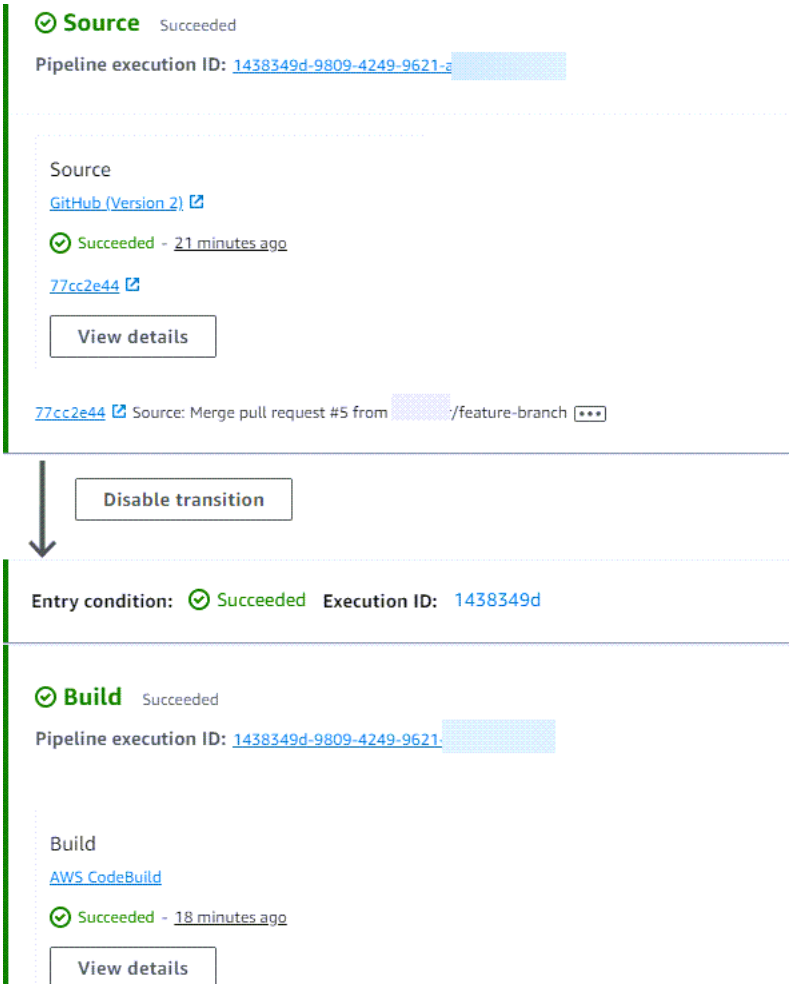
3. Pilih Simpan.

   Pilih Selesai.

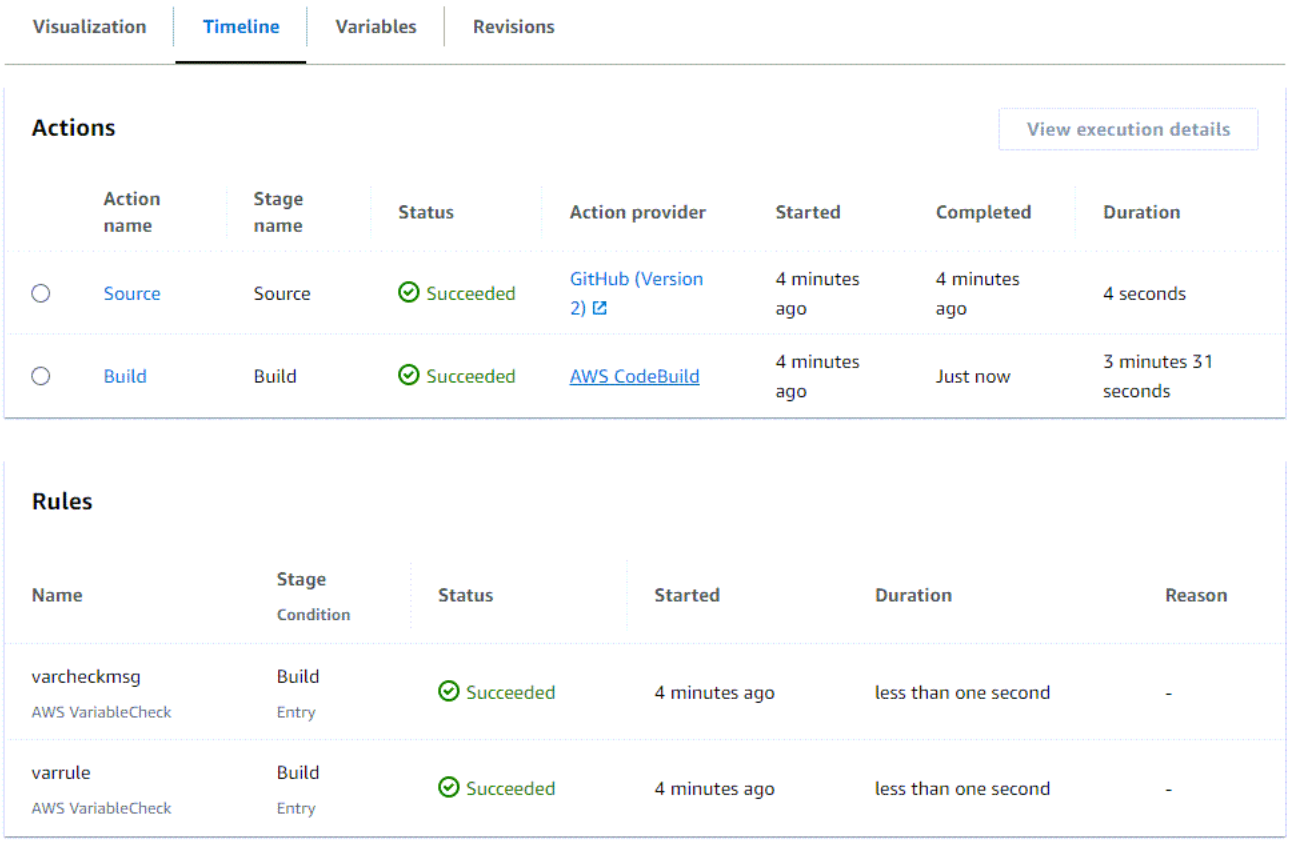
Langkah 3: Jalankan pipeline dan lihat variabel yang diselesaikan

Pada langkah ini, Anda melihat nilai dan hasil yang diselesaikan dari aturan pemeriksaan variabel.

1. Lihat proses yang diselesaikan setelah pemeriksaan aturan berhasil seperti yang ditunjukkan pada contoh berikut.



2. Lihat informasi variabel pada tab Timeline.


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 kasus penggunaan

Bagian berikut menjelaskan kasus penggunaan untuk CodePipeline.

Topik

- Gunakan kasus untuk CodePipeline

Gunakan kasus untuk CodePipeline

Anda dapat membuat saluran pipa yang terintegrasi dengan yang lain Layanan AWS. Ini bisa berupa Layanan AWS, seperti Amazon S3, atau produk pihak ketiga, seperti. GitHub Bagian ini memberikan contoh penggunaan CodePipeline untuk mengotomatiskan rilis kode Anda menggunakan integrasi produk yang berbeda. Untuk daftar lengkap integrasi dengan jenis tindakan yang CodePipeline diatur menurut, lihatCodePipeline referensi struktur pipa.

Topik

- Gunakan CodePipeline dengan Amazon S3,, dan AWS CodeCommit AWS CodeDeploy
- Gunakan CodePipeline dengan penyedia tindakan pihak ketiga (GitHub dan Jenkins)
- Gunakan CodePipeline untuk mengkompilasi, membangun, dan menguji kode dengan CodeBuild
- Gunakan CodePipeline dengan Amazon ECS untuk pengiriman berkelanjutan aplikasi berbasis kontainer ke cloud
- Gunakan CodePipeline dengan Elastic Beanstalk untuk pengiriman berkelanjutan aplikasi web ke cloud
- Gunakan CodePipeline dengan AWS Lambda untuk pengiriman berkelanjutan aplikasi berbasis Lambda dan tanpa server
- Gunakan CodePipeline dengan CloudFormation template untuk pengiriman berkelanjutan ke cloud

Gunakan CodePipeline dengan Amazon S3,, dan AWS CodeCommit AWS CodeDeploy

Saat Anda membuat pipeline, CodePipeline integrasikan dengan AWS produk dan layanan yang bertindak sebagai penyedia tindakan di setiap tahap pipeline Anda. Ketika Anda memilih tahapan di wizard, Anda harus memilih tahap sumber dan setidaknya tahap build atau deploy. Wizard membuat

tahapan untuk Anda dengan nama default yang tidak dapat diubah. Ini adalah nama panggung yang dibuat saat Anda menyiapkan pipeline tiga tahap penuh di wizard:

- Tahap aksi sumber dengan nama default “Sumber.”
- Tahap tindakan build dengan nama default “Build.”
- Tahap tindakan penerapan dengan nama default “Staging.”

Anda dapat menggunakan tutorial dalam panduan ini untuk membuat saluran pipa dan menentukan tahapan:

- Langkah-langkah dalam Tutorial: Buat pipeline sederhana (ember S3) membantu Anda menggunakan wizard untuk membuat pipeline dengan dua tahap default: “Sumber” dan “Pementasan”, di mana repositori Amazon S3 Anda adalah penyedia sumber. Tutorial ini membuat pipeline yang digunakan AWS CodeDeploy untuk menyebarkan aplikasi sampel dari bucket Amazon S3 ke instans Amazon EC2 yang menjalankan Amazon Linux.
- Langkah-langkah ini Tutorial: Buat pipeline sederhana (CodeCommit repositori) membantu Anda menggunakan wizard untuk membuat pipeline dengan tahap “Sumber” yang menggunakan AWS CodeCommit repositori Anda sebagai penyedia sumber. Tutorial ini membuat pipeline yang digunakan AWS CodeDeploy untuk menyebarkan aplikasi sampel dari AWS CodeCommit repositori ke instans Amazon EC2 yang menjalankan Amazon Linux.

Gunakan CodePipeline dengan penyedia tindakan pihak ketiga (GitHub dan Jenkins)

Anda dapat membuat jaringan pipa yang terintegrasi dengan produk pihak ketiga seperti GitHub dan Jenkins. Langkah-langkah dalam Tutorial: Buat pipeline empat tahap menunjukkan kepada Anda cara membuat pipeline yang:

- Mendapat kode sumber dari GitHub repositori,
- Menggunakan Jenkins untuk membangun dan menguji kode sumber,
- Menggunakan AWS CodeDeploy untuk menyebarkan kode sumber yang dibangun dan diuji ke instans Amazon EC2 yang menjalankan Amazon Linux atau Microsoft Windows Server.

Gunakan CodePipeline untuk mengkompilasi, membangun, dan menguji kode dengan CodeBuild

CodeBuild adalah layanan build terkelola di cloud yang memungkinkan Anda membangun dan menguji kode Anda tanpa server atau sistem. Gunakan CodePipeline dengan CodeBuild untuk mengotomatiskan revisi yang sedang berjalan melalui pipeline untuk pengiriman berkelanjutan dari build perangkat lunak setiap kali ada perubahan pada kode sumber. Untuk informasi selengkapnya, lihat Menggunakan CodePipeline dengan CodeBuild untuk menguji kode dan menjalankan build.

Gunakan CodePipeline dengan Amazon ECS untuk pengiriman berkelanjutan aplikasi berbasis kontainer ke cloud

Amazon ECS adalah layanan manajemen kontainer yang memungkinkan Anda menyebarkan aplikasi berbasis kontainer ke instans Amazon ECS di cloud. Gunakan CodePipeline dengan Amazon ECS untuk mengotomatiskan revisi yang sedang berjalan melalui pipeline untuk penerapan berkelanjutan aplikasi berbasis kontainer setiap kali ada perubahan pada repositori gambar sumber. Untuk informasi selengkapnya, lihat Tutorial: Penerapan Berkelanjutan dengan CodePipeline.

Gunakan CodePipeline dengan Elastic Beanstalk untuk pengiriman berkelanjutan aplikasi web ke cloud

Elastic Beanstalk adalah layanan komputasi yang memungkinkan Anda menyebarkan aplikasi dan layanan web ke server web. Gunakan CodePipeline dengan Elastic Beanstalk untuk penyebaran aplikasi web secara terus menerus ke lingkungan aplikasi Anda. Anda juga dapat menggunakan AWS CodeStar untuk membuat pipeline dengan aksi penyebaran Elastic Beanstalk.

Gunakan CodePipeline dengan AWS Lambda untuk pengiriman berkelanjutan aplikasi berbasis Lambda dan tanpa server

Anda dapat menggunakan AWS Lambda dengan CodePipeline untuk menjalankan AWS Lambda fungsi, seperti yang dijelaskan dalam Menyebarkan Aplikasi Tanpa Server. Anda juga dapat menggunakan AWS Lambda dan AWS CodeStar membuat pipeline untuk menyebarkan aplikasi tanpa server.

Gunakan CodePipeline dengan CloudFormation template untuk pengiriman berkelanjutan ke cloud

Anda dapat menggunakannya CloudFormation CodePipeline untuk pengiriman dan otomatisasi berkelanjutan. Untuk informasi selengkapnya, lihat Pengiriman Berkelanjutan dengan CodePipeline. CloudFormation juga digunakan untuk membuat template untuk pipeline yang dibuat di AWS CodeStar.

Gunakan CodePipeline dengan Amazon Virtual Private Cloud

AWS CodePipeline sekarang mendukung titik akhir Amazon Virtual Private Cloud (Amazon VPC) yang didukung oleh. AWS PrivateLink Ini berarti Anda dapat terhubung langsung CodePipeline melalui titik akhir pribadi di VPC Anda, menjaga semua lalu lintas di dalam VPC dan jaringan Anda. AWS

Amazon VPC adalah Layanan AWS yang dapat Anda gunakan untuk meluncurkan AWS sumber daya di jaringan virtual yang Anda tentukan. Dengan VPC, Anda memiliki kontrol atas pengaturan jaringan Anda, seperti:

- Rentang alamat IP
- Subnet
- Tabel rute
- Gateway jaringan

Endpoint VPC antarmuka didukung oleh AWS PrivateLink, sebuah AWS teknologi yang memfasilitasi komunikasi pribadi antara Layanan AWS menggunakan elastic network interface dengan alamat IP pribadi. Untuk menghubungkan VPC Anda CodePipeline, Anda menentukan titik akhir VPC antarmuka untuk. CodePipeline Jenis titik akhir ini memungkinkan Anda untuk menghubungkan Layanan AWS VPC Anda ke. Endpoint menyediakan konektivitas yang andal dan dapat diskalakan CodePipeline tanpa memerlukan gateway internet, instance terjemahan alamat jaringan (NAT), atau koneksi VPN. Untuk informasi tentang pengaturan VPC, lihat Panduan Pengguna VPC.

Ketersediaan

CodePipeline saat ini mendukung titik akhir VPC sebagai berikut: Wilayah AWS

- AS Timur (Ohio)
- AS Timur (Virginia Utara)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Canada (Central)

- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Milan) *
- Eropa (Paris)
- Eropa (Stockholm)
- Asia Pasifik (Hong Kong) *
- Asia Pasifik (Mumbai)
- Asia Pacific (Tokyo)
- Asia Pacific (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Amerika Selatan (Sao Paulo)
- AWS GovCloud (AS-Barat)

* Anda harus mengaktifkan Wilayah ini sebelum Anda dapat menggunakannya.

Buat titik akhir VPC untuk CodePipeline

Anda dapat menggunakan konsol VPC Amazon untuk membuat com.amazonaws.*region*.codepipeline titik akhir VPC. Di konsol, *region* adalah pengenal Wilayah untuk yang Wilayah AWS didukung oleh CodePipeline, seperti `us-east-2` untuk Wilayah Timur AS (Ohio). Untuk informasi selengkapnya, silakan lihat Membuat sebuah Titik Akhir Antarmuka dalam Panduan Pengguna Amazon VPC.

Titik akhir diisi sebelumnya dengan Wilayah yang Anda tentukan saat Anda masuk. AWS Jika Anda masuk ke Wilayah lain, titik akhir VPC diperbarui dengan Wilayah baru.

> **Note**
>
> Lainnya Layanan AWS yang menyediakan dukungan VPC dan terintegrasi dengan CodePipeline, seperti CodeCommit, mungkin tidak mendukung penggunaan titik akhir Amazon VPC untuk integrasi itu. Misalnya, lalu lintas antara CodePipeline dan CodeCommit tidak dapat dibatasi pada rentang subnet VPC.

Memecahkan masalah penyiapan VPC

Saat memecahkan masalah VPC, gunakan informasi yang muncul di pesan kesalahan konektivitas internet untuk membantu Anda mengidentifikasi, mendiagnosis, dan mengatasi masalah.

1. Pastikan gateway internet Anda terpasang ke VPC Anda.
2. Pastikan bahwa tabel rute untuk subnet publik Anda menunjuk ke gateway internet.
3. Pastikan jaringan Anda ACLs memungkinkan lalu lintas mengalir.
4. Pastikan grup keamanan Anda mengizinkan lalu lintas mengalir.
5. Pastikan bahwa tabel rute untuk subnet pribadi menunjuk ke gateway pribadi virtual.
6. Pastikan bahwa peran layanan yang digunakan oleh CodePipeline memiliki izin yang sesuai. Misalnya, jika CodePipeline tidak memiliki izin Amazon EC2 yang diperlukan untuk bekerja dengan VPC Amazon, Anda mungkin menerima kesalahan yang mengatakan, “Kesalahan EC2 yang tidak terduga:.” UnauthorizedOperation

Tentukan CI/CD saluran pipa dengan tahapan dan tindakan

Untuk menentukan proses rilis otomatis di AWS CodePipeline, Anda membuat pipeline, yang merupakan konstruksi alur kerja yang menjelaskan bagaimana perubahan perangkat lunak melalui proses rilis. Pipeline terdiri dari tahapan dan tindakan yang Anda konfigurasikan.

> **Note**
>
> Saat menambahkan tahapan Build, Deploy, Test, atau Invoke, selain opsi default yang disediakan CodePipeline, Anda dapat memilih tindakan kustom yang telah dibuat untuk digunakan dengan pipeline Anda. Tindakan kustom dapat digunakan untuk tugas-tugas seperti menjalankan proses build yang dikembangkan secara internal atau rangkaian pengujian. Pengidentifikasi versi disertakan untuk membantu Anda membedakan di antara berbagai versi tindakan kustom dalam daftar penyedia. Untuk informasi selengkapnya, lihat Buat dan tambahkan tindakan kustom di CodePipeline.

Sebelum Anda dapat membuat pipeline, Anda harus terlebih dahulu menyelesaikan langkah-langkahnyaMemulai dengan CodePipeline.

Untuk informasi selengkapnya tentang pipelineCodePipeline konsep , lihatCodePipeline tutorial,, dan, jika Anda ingin menggunakan AWS CLI untuk membuat pipeline,Buat pipeline, tahapan, dan tindakan. Untuk melihat daftar saluran pipa, lihatLihat saluran pipa dan detailnya di CodePipeline.

Topik

- Buat pipeline, tahapan, dan tindakan
- Edit pipa di CodePipeline
- Lihat saluran pipa dan detailnya di CodePipeline
- Hapus pipa di CodePipeline
- Buat pipeline CodePipeline yang menggunakan sumber daya dari AWS akun lain
- Migrasi jaringan pemungutan suara untuk menggunakan deteksi perubahan berbasis peristiwa
- Buat peran CodePipeline layanan
- Penandaan pada sumber daya
- Tandai pipa di CodePipeline

- Membuat aturan notifikasi

Buat pipeline, tahapan, dan tindakan

Anda dapat menggunakan AWS CodePipeline konsol atau AWS CLI untuk membuat pipeline. Pipa harus memiliki setidaknya dua tahap. Tahap pertama dari pipa harus menjadi tahap sumber. Pipeline harus memiliki setidaknya satu tahap lain yaitu tahap build atau deployment.

Important

Sebagai bagian dari pembuatan pipa, ember artefak S3 yang disediakan oleh pelanggan akan digunakan CodePipeline untuk artefak. (Ini berbeda dari bucket yang digunakan untuk aksi sumber S3.) Jika bucket artefak S3 berada di akun yang berbeda dari akun untuk pipeline Anda, pastikan bucket artefak S3 dimiliki oleh Akun AWS yang aman dan dapat diandalkan.

Anda dapat menambahkan tindakan ke pipeline yang Wilayah AWS berbeda dari pipeline Anda. Tindakan lintas wilayah adalah tindakan di mana an Layanan AWS adalah penyedia tindakan dan tipe tindakan atau tipe penyedia berada di AWS Wilayah yang berbeda dari pipeline Anda. Untuk informasi selengkapnya, lihat Menambahkan tindakan Lintas wilayah di CodePipeline.

Anda juga dapat membuat pipeline yang membangun dan menyebarkan aplikasi berbasis container dengan menggunakan Amazon ECS sebagai penyedia penyebaran. Sebelum membuat pipeline yang menyebarkan aplikasi berbasis container dengan Amazon ECS, Anda harus membuat file definisi gambar seperti yang dijelaskan dalam. Referensi file definisi gambar

CodePipeline menggunakan metode deteksi perubahan untuk memulai pipeline Anda saat perubahan kode sumber didorong. Metode deteksi ini didasarkan pada jenis sumber:

- CodePipeline menggunakan Amazon CloudWatch Events untuk mendeteksi perubahan di repositori CodeCommit sumber dan cabang atau bucket sumber S3 Anda.

Note

Saat Anda menggunakan konsol untuk membuat atau mengedit pipeline, sumber daya deteksi perubahan akan dibuat untuk Anda. Jika Anda menggunakan AWS CLI untuk

membuat pipeline, Anda harus membuat sumber daya tambahan sendiri. Untuk informasi selengkapnya, lihat CodeCommit tindakan sumber dan EventBridge.

Topik

- Buat pipeline khusus (konsol)
- Buat pipeline (CLI)
- Buat pipeline dari template statis

Buat pipeline khusus (konsol)

Untuk membuat pipeline khusus di konsol, Anda harus memberikan lokasi file sumber dan informasi tentang penyedia yang akan Anda gunakan untuk tindakan Anda.

Saat Anda menggunakan konsol untuk membuat pipeline, Anda harus menyertakan tahap sumber dan salah satu atau kedua hal berikut:

- Panggung membangun.
- Tahap penyebaran.

Saat Anda menggunakan wizard pipeline, CodePipeline buat nama tahapan (source, build, staging). Nama-nama ini tidak dapat diubah. Anda dapat menggunakan nama yang lebih spesifik (misalnya, BuildToGamma atau DeployToProd) ke tahapan yang Anda tambahkan nanti.

Langkah 1: Buat dan beri nama pipeline Anda

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Selamat Datang, pilih Buat pipeline.

   Jika ini adalah pertama kalinya Anda menggunakan CodePipeline, pilih Memulai.
3. Pada Langkah 1: Pilih halaman opsi pembuatan, di bawah Opsi pembuatan, pilih opsi Build custom pipeline. Pilih Berikutnya.
4. Pada Langkah 2: Pilih halaman pengaturan pipeline, dalam nama Pipeline, masukkan nama untuk pipeline Anda.

Dalam satu AWS akun, setiap pipeline yang Anda buat di AWS Wilayah harus memiliki nama yang unik. Nama dapat digunakan kembali untuk jaringan pipa di berbagai Wilayah.

> **Note**
>
> Setelah Anda membuat pipeline, Anda tidak dapat mengubah namanya. Untuk informasi tentang batasan lainnya, lihatKuota di AWS CodePipeline.

5. Dalam tipe Pipeline, pilih salah satu opsi berikut. Jenis pipa berbeda dalam karakteristik dan harga. Untuk informasi selengkapnya, lihat Jenis pipa.

   - Pipa tipe V1 memiliki struktur JSON yang berisi pipa standar, tahap, dan parameter tingkat aksi.
   - Pipeline tipe V2 memiliki struktur yang sama dengan tipe V1, bersama dengan dukungan parameter tambahan, seperti pemicu pada tag Git dan variabel tingkat pipa.

6. Dalam peran Layanan, lakukan salah satu hal berikut:

   - Pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan baru di IAM.
   - Pilih Peran layanan yang ada untuk menggunakan peran layanan yang sudah dibuat di IAM. Di ARN Peran, pilih ARN peran layanan Anda dari daftar.

   > **Note**
   >
   > Bergantung pada kapan peran layanan Anda dibuat, Anda mungkin perlu memperbarui izinnya untuk mendukung tambahan Layanan AWS. Untuk informasi, lihat Menambahkan izin ke peran CodePipeline layanan.

   Untuk informasi selengkapnya tentang peran layanan dan pernyataan kebijakannya, lihatKelola peran CodePipeline layanan.

7. (Opsional) Di bawah Variabel, pilih Tambahkan variabel untuk menambahkan variabel di tingkat pipeline.

Untuk informasi lebih lanjut tentang variabel di tingkat pipa, lihatReferensi variabel. Untuk tutorial dengan variabel tingkat pipa yang dilewatkan pada saat eksekusi pipeline, lihat. Tutorial: Gunakan variabel tingkat pipa

> **Note**
>
> Meskipun opsional untuk menambahkan variabel pada tingkat pipa, untuk pipeline yang ditentukan dengan variabel pada tingkat pipa di mana tidak ada nilai yang disediakan, eksekusi pipeline akan gagal.

8. (Opsional) Perluas pengaturan lanjutan.
9. Di toko Artifact, lakukan salah satu hal berikut:

   a. Pilih lokasi default untuk menggunakan penyimpanan artefak default, seperti bucket artefak S3 yang ditetapkan sebagai default, untuk pipeline di pipeline yang telah Wilayah AWS Anda pilih untuk pipeline Anda.

   b. Pilih Lokasi khusus jika Anda sudah memiliki toko artefak, seperti bucket artefak S3, di Wilayah yang sama dengan pipeline Anda. Di Bucket, pilih nama bucket.

> **Note**
>
> Ini bukan bucket sumber untuk kode sumber Anda. Ini adalah penyimpanan artifact untuk alur Anda. Penyimpanan artifact terpisah, seperti bucket S3, diperlukan untuk setiap alur. Saat membuat atau mengedit pipeline, Anda harus memiliki bucket artefak di Wilayah pipeline dan satu bucket artefak per AWS Wilayah tempat Anda menjalankan tindakan. Untuk informasi selengkapnya, lihat Artefak input dan output dan CodePipeline referensi struktur pipa.

10. Dalam kunci Enkripsi, lakukan salah satu hal berikut:

    a. Untuk menggunakan CodePipeline default AWS KMS key untuk mengenkripsi data di penyimpanan artefak pipeline (bucket S3), pilih Default AWS Managed Key.

    b. Untuk menggunakan kunci terkelola pelanggan Anda untuk mengenkripsi data di toko artefak pipeline (bucket S3), pilih Kunci yang Dikelola Pelanggan. Pilih ID kunci, kunci ARN, atau alias ARN.

11. Pilih Berikutnya.

Langkah 2: Buat tahap sumber

1. Pada Langkah 3: Tambahkan halaman tahap sumber, di Penyedia sumber, pilih jenis repositori tempat kode sumber Anda disimpan, tentukan opsi yang diperlukan. Bidang tambahan ditampilkan tergantung pada penyedia sumber yang dipilih sebagai berikut.

   - Untuk Bitbucket Cloud, GitHub (melalui GitHub Aplikasi), Server GitHub Perusahaan, GitLab .com, atau yang dikelola GitLab sendiri:

     1. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.
     2. Pilih repositori yang ingin Anda gunakan sebagai lokasi sumber untuk pipeline Anda.

        Pilih untuk menambahkan pemicu atau filter pada jenis pemicu untuk memulai pipeline Anda. Untuk informasi selengkapnya tentang bekerja dengan pemicu, lihatTambahkan pemicu dengan tipe acara permintaan push atau pull kode. Untuk informasi selengkapnya tentang pemfilteran dengan pola glob, lihat. Bekerja dengan pola glob dalam sintaks
     3. Dalam format artefak Output, pilih format untuk artefak Anda.
        - Untuk menyimpan artefak keluaran dari GitHub tindakan menggunakan metode default, pilih CodePipelinedefault. Tindakan mengakses file dari GitHub repositori dan menyimpan artefak dalam file ZIP di toko artefak pipa.
        - Untuk menyimpan file JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung, pilih klon Penuh. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.

          Jika Anda memilih opsi ini, Anda perlu memperbarui izin untuk peran layanan CodeBuild proyek Anda seperti yang ditunjukkan padaPemecahan masalah CodePipeline. Untuk tutorial yang menunjukkan cara menggunakan opsi klon Penuh, lihatTutorial: Gunakan klon lengkap dengan sumber GitHub pipa.

   - Untuk Amazon S3:

     1. Di lokasi Amazon S3, berikan nama bucket S3 dan path ke objek dalam bucket dengan versi diaktifkan. Format nama bucket dan path terlihat seperti ini:

        ```
        s3://bucketName/folderName/objectName
        ```

> **Note**
>
> Jika Amazon S3 adalah penyedia sumber untuk pipeline Anda, Anda dapat mem-zip file sumber atau file ke dalam satu .zip dan mengunggah.zip ke bucket sumber Anda. Anda juga dapat mengunggah satu file yang tidak di-zip; namun, tindakan hilir yang mengharapkan file.zip akan gagal.

2. Setelah memilih bucket sumber S3, CodePipeline buat aturan Amazon CloudWatch Events dan AWS CloudTrail jejak yang akan dibuat untuk pipeline ini. Terima default di bawah Ubah opsi deteksi. Hal ini memungkinkan Anda CodePipeline untuk menggunakan Amazon CloudWatch Events dan AWS CloudTrail mendeteksi perubahan pada pipeline baru Anda. Pilih Berikutnya.

- Untuk AWS CodeCommit:
  - Dalam nama Repositori, pilih nama CodeCommit repositori yang ingin Anda gunakan sebagai lokasi sumber untuk pipeline Anda. Di Nama cabang, dari daftar drop-down, pilih cabang yang ingin Anda gunakan.
  - Dalam format artefak Output, pilih format untuk artefak Anda.
    - Untuk menyimpan artefak keluaran dari CodeCommit tindakan menggunakan metode default, pilih CodePipelinedefault. Tindakan mengakses file dari CodeCommit repositori dan menyimpan artefak dalam file ZIP di toko artefak pipa.
    - Untuk menyimpan file JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung, pilih klon Penuh. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.

      Jika Anda memilih opsi ini, Anda perlu menambahkan `codecommit:GitPull` izin ke peran CodeBuild layanan Anda seperti yang ditunjukkan padaTambahkan CodeBuild GitClone izin untuk tindakan CodeCommit sumber. Anda juga perlu menambahkan `codecommit:GetRepository` izin ke peran CodePipeline layanan Anda seperti yang ditunjukkan padaMenambahkan izin ke peran CodePipeline layanan. Untuk tutorial yang menunjukkan cara menggunakan opsi klon Penuh, lihatTutorial: Gunakan klon lengkap dengan sumber GitHub pipa.
  - Setelah Anda memilih nama CodeCommit repositori dan cabang, pesan akan ditampilkan di Ubah opsi deteksi yang menunjukkan aturan Amazon CloudWatch Events yang akan dibuat untuk pipeline ini. Terima default di bawah Ubah opsi deteksi. Hal ini CodePipeline

memungkinkan Anda menggunakan Amazon CloudWatch Events untuk mendeteksi perubahan pada pipeline baru Anda.

- Untuk Amazon ECR:
  - Dalam nama Repositori, pilih nama repositori Amazon ECR Anda.
  - Di tag Gambar, tentukan nama dan versi gambar, jika berbeda dari TERBARU.
  - Di artefak Output, pilih artefak keluaran default MyApp, seperti, yang berisi nama gambar dan informasi URI repositori yang ingin digunakan tahap selanjutnya.

    Untuk tutorial tentang membuat pipeline untuk Amazon ECS dengan penerapan CodeDeploy biru-hijau yang menyertakan tahap sumber Amazon ECR, lihat. Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR

  Saat Anda menyertakan tahap sumber ECR Amazon dalam pipeline, tindakan sumber akan menghasilkan `imageDetail.json` file sebagai artefak keluaran saat Anda melakukan perubahan. Untuk informasi tentang `imageDetail.json` file, lihatFile ImageDetail.json untuk tindakan penerapan Amazon ECS blue/green.

> **Note**
>
> Objek dan jenis file harus kompatibel dengan sistem penyebaran yang Anda rencanakan untuk digunakan (misalnya, Elastic Beanstalk atau). CodeDeploy Jenis file yang didukung mungkin termasuk file.zip, .tar, dan .tgz. Untuk informasi selengkapnya tentang jenis wadah yang didukung untuk Elastic Beanstalk, lihat Menyesuaikan dan Mengonfigurasi Lingkungan Elastic Beanstalk dan Platform yang Didukung. Untuk informasi selengkapnya tentang menerapkan revisi dengan CodeDeploy, lihat Mengunggah Revisi Aplikasi Anda dan Mempersiapkan Revisi.

2. Untuk mengonfigurasi panggung untuk percobaan ulang otomatis, pilih Aktifkan coba ulang otomatis pada kegagalan panggung. Untuk informasi selengkapnya tentang coba ulang otomatis, lihatKonfigurasikan tahapan untuk coba lagi otomatis pada kegagalan.
3. Pilih Berikutnya.

Langkah 4: Buat tahap build

Langkah ini opsional jika Anda berencana untuk membuat tahap penerapan.

1. Pada Langkah 4: Tambahkan halaman tahap build, lakukan salah satu hal berikut, lalu pilih Berikutnya:

   - Pilih Lewati tahap build jika Anda berencana membuat tahap pengujian atau penerapan.
   - Untuk memilih tindakan Command untuk tahap build Anda, pilih Command.

     > **Note**
     >
     > Menjalankan tindakan Commands akan dikenakan biaya terpisah di AWS CodeBuild. Jika Anda berencana untuk menyisipkan perintah build sebagai bagian dari CodeBuild tindakan, lanjutkan dan pilih Penyedia build lainnya, lalu pilih CodeBuild.

     Di Command, masukkan perintah shell untuk tindakan Anda. Untuk informasi selengkapnya tentang tindakan Perintah, lihatReferensi tindakan perintah.
   - Untuk memilih penyedia build lain seperti CodeBuild, pilih Penyedia lain. Dari penyedia Build, pilih penyedia tindakan khusus untuk layanan build, dan berikan detail konfigurasi untuk penyedia tersebut. Untuk contoh cara menambahkan Jenkins sebagai penyedia build, lihatTutorial: Buat pipeline empat tahap.
   - Dari penyedia Build, pilih AWS CodeBuild.

     Di Wilayah, pilih AWS Wilayah tempat sumber daya ada. Bidang Region menentukan tempat AWS sumber daya dibuat untuk tipe tindakan dan jenis penyedia ini. Bidang ini hanya ditampilkan untuk tindakan di mana penyedia tindakan adalah Layanan AWS. Bidang Region default ke AWS Region yang sama dengan pipeline Anda.

     Dalam nama Project, pilih proyek build Anda. Jika Anda telah membuat proyek build di CodeBuild, pilihlah. Atau Anda dapat membuat proyek build CodeBuild dan kemudian kembali ke tugas ini. Ikuti petunjuk di Buat Pipeline yang Digunakan CodeBuild di Panduan CodeBuild Pengguna.

     Di bawah spesifikasi Build, file CodeBuild buildspec bersifat opsional, dan Anda dapat memasukkan perintah sebagai gantinya. Dalam Insert perintah build, masukkan perintah shell untuk tindakan Anda. Untuk informasi selengkapnya tentang pertimbangan penggunaan perintah build, lihatReferensi tindakan perintah. Pilih Gunakan file buildspec jika Anda ingin menjalankan perintah dalam fase lain atau jika Anda memiliki daftar perintah yang panjang.

Dalam variabel Lingkungan, untuk menambahkan variabel CodeBuild lingkungan ke tindakan build Anda, pilih Tambahkan variabel lingkungan. Setiap variabel terdiri dari tiga entri:

- Dalam Nama, masukkan nama atau kunci variabel lingkungan.
- Dalam Nilai, masukkan nilai variabel lingkungan. Jika Anda memilih Parameter untuk tipe variabel, pastikan nilai ini adalah nama parameter yang telah Anda simpan di AWS Systems Manager Parameter Store.

> **Note**
>
> Kami sangat tidak menyarankan penggunaan variabel lingkungan untuk menyimpan nilai sensitif, terutama AWS kredensil. Saat Anda menggunakan CodeBuild konsol atau AWS CLI, variabel lingkungan ditampilkan dalam teks biasa. Untuk nilai sensitif, kami sarankan Anda menggunakan tipe Parameter sebagai gantinya.

- (Opsional) Dalam Jenis, masukkan jenis variabel lingkungan. Nilai yang valid adalah Plaintext atau Parameter. Defaultnya adalah Plaintext.

(Opsional) Dalam tipe Build, pilih salah satu dari berikut ini:

- Untuk menjalankan setiap build dalam satu eksekusi aksi build, pilih Single build.
- Untuk menjalankan beberapa build dalam eksekusi aksi build yang sama, pilih Batch build.

(Opsional) Jika Anda memilih untuk menjalankan build batch, Anda dapat memilih Gabungkan semua artefak dari batch ke satu lokasi untuk menempatkan semua artefak build ke dalam artefak keluaran tunggal.

2. Untuk mengonfigurasi panggung untuk percobaan ulang otomatis, pilih Aktifkan coba ulang otomatis pada kegagalan panggung. Untuk informasi selengkapnya tentang coba ulang otomatis, lihatKonfigurasikan tahapan untuk coba lagi otomatis pada kegagalan.
3. Pilih Berikutnya.

Langkah 5: Buat tahap pengujian

Langkah ini bersifat opsional jika Anda berencana membuat tahap build atau deployment.

1. Pada Langkah 5: Tambahkan halaman tahap pengujian, lakukan salah satu hal berikut, lalu pilih Berikutnya:

   - Pilih Lewati tahap pengujian jika Anda berencana membuat tahap build atau deployment.

- Di Penyedia pengujian, pilih penyedia tindakan pengujian dan lengkapi bidang yang sesuai.

2. Pilih Berikutnya.

Langkah 6: Buat tahap penyebaran

Langkah ini opsional jika Anda telah membuat tahap build.

1. Pada Langkah 6: Tambahkan halaman tahap penerapan, lakukan salah satu hal berikut, lalu pilih Berikutnya:

   - Pilih Lewati tahap penerapan jika Anda membuat tahap build atau pengujian di langkah sebelumnya.

     > **Note**
     >
     > Opsi ini tidak muncul jika Anda telah melewati tahap pembuatan atau pengujian.

   - Di penyedia Deploy, pilih tindakan kustom yang telah Anda buat untuk penyedia penerapan.

     Di Wilayah, hanya untuk tindakan Lintas wilayah, pilih AWS Wilayah tempat sumber daya dibuat. Bidang Region menentukan tempat AWS sumber daya dibuat untuk tipe tindakan dan jenis penyedia ini. Bidang ini hanya menampilkan tindakan di mana penyedia tindakan adalah Layanan AWS. Bidang Region default ke AWS Region yang sama dengan pipeline Anda.

   - Di penyedia Deploy, bidang tersedia untuk penyedia default sebagai berikut:

     - CodeDeploy

       Dalam nama Aplikasi, masukkan atau pilih nama CodeDeploy aplikasi yang ada. Di grup Deployment, masukkan nama grup penyebaran untuk aplikasi. Pilih Berikutnya. Anda juga dapat membuat aplikasi, grup penyebaran, atau keduanya di CodeDeploy konsol.

     - AWS Elastic Beanstalk

       Pada nama Aplikasi, masukkan atau pilih nama aplikasi Elastic Beanstalk yang ada. Dalam nama Lingkungan, masukkan lingkungan untuk aplikasi. Pilih Berikutnya. Anda juga dapat membuat aplikasi, lingkungan, atau keduanya di konsol Elastic Beanstalk.

     - AWS OpsWorks Stacks

Di Stack, masukkan atau pilih nama tumpukan yang ingin Anda gunakan. Di Layer, pilih layer yang menjadi milik instance target Anda. Di App, pilih aplikasi yang ingin Anda perbarui dan terapkan. Jika Anda perlu membuat aplikasi, pilih Buat yang baru di AWS OpsWorks.

Untuk informasi tentang menambahkan aplikasi ke tumpukan dan lapisan AWS OpsWorks, lihat Menambahkan Aplikasi di Panduan AWS OpsWorks Pengguna.

Untuk end-to-end contoh cara menggunakan pipeline sederhana CodePipeline sebagai sumber kode yang Anda jalankan pada AWS OpsWorks lapisan, lihat Menggunakan CodePipeline dengan AWS OpsWorks Stacks.

- AWS CloudFormation

  Lakukan salah satu tindakan berikut:

  - Dalam mode Tindakan, pilih Buat atau perbarui tumpukan, masukkan nama tumpukan dan nama file templat, lalu pilih nama peran AWS CloudFormation untuk diasumsikan. Secara opsional, masukkan nama file konfigurasi dan pilih opsi kemampuan IAM.
  - Dalam mode Tindakan, pilih Buat atau ganti set perubahan, masukkan nama tumpukan dan ubah nama set, lalu pilih nama peran AWS CloudFormation untuk diasumsikan. Secara opsional, masukkan nama file konfigurasi dan pilih opsi kemampuan IAM.

  Untuk informasi tentang mengintegrasikan AWS CloudFormation kemampuan ke dalam pipeline CodePipeline, lihat Pengiriman Berkelanjutan dengan CodePipeline di Panduan AWS CloudFormation Pengguna.

- Amazon ECS

  Dalam nama Cluster, masukkan atau pilih nama cluster Amazon ECS yang ada. Di nama Layanan, masukkan atau pilih nama layanan yang berjalan di cluster. Anda juga dapat membuat cluster dan layanan. Dalam nama file Image, masukkan nama file definisi gambar yang menjelaskan wadah dan gambar layanan Anda.

  > **Note**
  >
  > Tindakan penyebaran Amazon ECS memerlukan `imagedefinitions.json` file sebagai masukan untuk tindakan penerapan. Nama file default untuk fille adalah imagedefinitions.json. Jika Anda memilih untuk menggunakan nama file yang berbeda, Anda harus menyediakannya saat membuat tahap penerapan pipeline.

Untuk informasi selengkapnya, lihat file imagedefinitions.json untuk tindakan penerapan standar Amazon ECS.

Pilih Berikutnya.

> **Note**
>
> Pastikan cluster Amazon ECS Anda dikonfigurasi dengan dua instans atau lebih. Cluster Amazon ECS harus berisi setidaknya dua instans sehingga satu dipertahankan sebagai instans utama dan yang lain digunakan untuk mengakomodasi penerapan baru.

Untuk tutorial tentang penerapan aplikasi berbasis container dengan pipeline Anda, lihat Tutorial: Continuous Deployment with. CodePipeline

- Amazon ECS (Biru/Hijau)

  Masukkan grup CodeDeploy aplikasi dan penyebaran, definisi tugas Amazon ECS, dan informasi AppSpec file, lalu pilih Berikutnya.

  > **Note**
  >
  > Tindakan Amazon ECS (Biru/Hijau) memerlukan file ImageDetail.json sebagai artefak masukan untuk tindakan penerapan. Karena tindakan sumber Amazon ECR membuat file ini, saluran pipa dengan tindakan sumber Amazon ECR tidak perlu menyediakan file. `imageDetail.json` Untuk informasi selengkapnya, lihat File ImageDetail.json untuk tindakan penerapan Amazon ECS blue/green.

  Untuk tutorial tentang membuat pipeline untuk penerapan biru-hijau ke cluster Amazon ECS dengan, lihat. CodeDeploy Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR

- AWS Service Catalog

Pilih Masukkan konfigurasi penerapan jika Anda ingin menggunakan bidang di konsol untuk menentukan konfigurasi Anda, atau pilih File konfigurasi jika Anda memiliki file konfigurasi terpisah. Masukkan informasi produk dan konfigurasi, lalu pilih Berikutnya.

Untuk tutorial tentang menerapkan perubahan produk ke Service Catalog dengan pipeline Anda, lihatTutorial: Buat pipeline yang di-deploy ke Service Catalog.

- Kit Keterampilan Alexa

  Di Alexa Skill ID, masukkan ID keterampilan untuk keterampilan Alexa Anda. Dalam ID Klien dan rahasia Klien, masukkan kredensil yang dihasilkan menggunakan profil keamanan Login with Amazon (LWA). Di token Refresh, masukkan token penyegaran yang Anda buat menggunakan perintah ASK CLI untuk mengambil token penyegaran. Pilih Berikutnya.

  Untuk tutorial tentang menerapkan keterampilan Alexa dengan pipeline Anda dan menghasilkan kredenal LWA, lihat. Tutorial: Buat pipeline yang menerapkan keterampilan Amazon Alexa

- Amazon S3

  Di Bucket, masukkan nama bucket S3 yang ingin Anda gunakan. Pilih Ekstrak file sebelum menerapkan jika artefak input ke tahap penerapan Anda adalah file ZIP. Jika Ekstrak file sebelum penerapan dipilih, Anda dapat secara opsional memasukkan nilai untuk jalur Deployment tempat file ZIP Anda akan dibuka ritsletingnya. Jika tidak dipilih, Anda diminta untuk memasukkan nilai dalam kunci objek S3.

  > **Note**
  >
  > Sebagian besar artefak keluaran tahap sumber dan build di-zip. Semua penyedia sumber pipeline kecuali Amazon S3 mem-zip file sumber Anda sebelum menyediakannya sebagai artefak input ke tindakan selanjutnya.

  (Opsional) Di ACL Kalengan, masukkan ACL kalengan untuk diterapkan ke objek yang disebarkan ke Amazon S3.

  > **Note**
  >
  > Menerapkan ACL kalengan menimpa ACL yang ada yang diterapkan ke objek.

(Opsional) Dalam kontrol Cache, tentukan parameter kontrol cache untuk permintaan mengunduh objek dari ember. Untuk daftar nilai yang valid, lihat bidang `Cache-Control`header untuk operasi HTTP. Untuk memasukkan beberapa nilai dalam kontrol Cache, gunakan koma di antara setiap nilai. Anda dapat menambahkan spasi setelah setiap koma (opsional), seperti yang ditunjukkan dalam contoh ini.

Contoh entri sebelumnya ditampilkan di CLI sebagai berikut:

```
"CacheControl": "public, max-age=0, no-transform"
```

Pilih Berikutnya.

Untuk tutorial tentang membuat pipeline dengan penyedia tindakan penerapan Amazon S3, lihat. Tutorial: Membuat pipeline yang menggunakan Amazon S3 sebagai penyedia penerapan

2. Untuk mengonfigurasi panggung untuk percobaan ulang otomatis, pilih Aktifkan coba ulang otomatis pada kegagalan panggung. Untuk informasi selengkapnya tentang coba ulang otomatis, lihatKonfigurasikan tahapan untuk coba lagi otomatis pada kegagalan.
3. Untuk mengkonfigurasi panggung untuk rollback otomatis, pilih Konfigurasikan rollback otomatis pada kegagalan panggung. Untuk informasi selengkapnya tentang rollback otomatis, lihat. Konfigurasikan panggung untuk rollback otomatis
4. Pilih Langkah selanjutnya.

Langkah 7: Tinjau pipa

- Pada Langkah 7: Tinjau halaman, tinjau konfigurasi pipeline Anda, lalu pilih Buat pipeline untuk membuat pipeline atau Sebelumnya untuk kembali dan mengedit pilihan Anda. Untuk keluar dari wizard tanpa membuat pipeline, pilih Batal.

Sekarang setelah Anda membuat pipeline, Anda dapat melihatnya di konsol. Pipeline mulai berjalan setelah Anda membuatnya. Untuk informasi selengkapnya, lihat Lihat saluran pipa dan detailnya

di CodePipeline. Untuk informasi selengkapnya tentang membuat perubahan pada pipeline Anda, lihatEdit pipa di CodePipeline.

Buat pipeline (CLI)

Untuk menggunakan AWS CLI untuk membuat pipeline, Anda membuat file JSON untuk menentukan struktur pipa, dan kemudian menjalankan create-pipeline perintah dengan `--cli-input-json` parameter.

> **Important**
>
> Anda tidak dapat menggunakan AWS CLI untuk membuat pipeline yang mencakup tindakan mitra. Anda harus menggunakan CodePipeline konsol sebagai gantinya.

Untuk informasi selengkapnya tentang struktur pipeline, lihat CodePipeline referensi struktur pipa dan buat-pipeline di Referensi API. CodePipeline

Untuk membuat file JSON, gunakan file JSON pipeline sampel, edit, lalu panggil file itu saat Anda menjalankan perintah. create-pipeline

Prasyarat:

Anda memerlukan ARN dari peran layanan yang Anda buat CodePipeline . Memulai dengan CodePipeline Anda menggunakan ARN peran CodePipeline layanan dalam file JSON pipeline saat Anda menjalankan perintah. create-pipeline Untuk informasi selengkapnya tentang pembuatan peran layanan, lihat Buat peran CodePipeline layanan. Berbeda dengan konsol, menjalankan create-pipeline perintah di AWS CLI tidak memiliki opsi untuk membuat peran CodePipeline layanan untuk Anda. Peran layanan harus sudah ada.

Anda memerlukan nama ember S3 tempat artefak untuk pipa disimpan. Bucket ini harus berada di Wilayah yang sama dengan pipa. Anda menggunakan nama bucket dalam file JSON pipeline saat Anda menjalankan create-pipeline perintah. Berbeda dengan konsol, menjalankan create-pipeline perintah di AWS CLI tidak membuat bucket S3 untuk menyimpan artefak. Bucket harus sudah ada.

> **Note**
>
> Anda juga dapat menggunakan get-pipeline perintah untuk mendapatkan salinan struktur JSON dari pipeline itu, dan kemudian memodifikasi struktur itu dalam editor teks biasa.

Untuk membuat file JSON

1. Di terminal (Linux, macOS, atau Unix) atau command prompt (Windows), buat file teks baru di direktori lokal.
2. (Opsional) Anda dapat menambahkan satu atau lebih variabel di tingkat pipa. Anda dapat mereferensikan nilai ini dalam konfigurasi CodePipeline tindakan. Anda dapat menambahkan nama dan nilai variabel saat membuat pipeline, dan Anda juga dapat memilih untuk menetapkan nilai saat memulai pipeline di konsol.

   > **Note**
   >
   > Meskipun opsional untuk menambahkan variabel pada tingkat pipa, untuk pipeline yang ditentukan dengan variabel pada tingkat pipa di mana tidak ada nilai yang disediakan, eksekusi pipeline akan gagal.

   Variabel pada tingkat pipa diselesaikan pada waktu proses pipa. Semua variabel tidak dapat diubah, yang berarti bahwa mereka tidak dapat diperbarui setelah nilai ditetapkan. Variabel pada tingkat pipeline dengan nilai yang diselesaikan akan ditampilkan dalam riwayat untuk setiap eksekusi.

   Anda menyediakan variabel pada tingkat pipeline menggunakan atribut variabel dalam struktur pipa. Dalam contoh berikut, variabel `Variable1` memiliki nilai`Value1`.

   ```
   "variables": [
        {
            "name": "Timeout",
            "defaultValue": "1000",
            "description": "description"
        }
    ]
   ```

   Tambahkan struktur ini ke JSON pipeline Anda, atau ke contoh JSON pada langkah berikut. Untuk informasi selengkapnya tentang variabel, termasuk informasi namespace, lihat. Referensi variabel
3. Buka file di editor teks biasa dan edit nilainya untuk mencerminkan struktur yang ingin Anda buat. Minimal, Anda harus mengubah nama pipa. Anda juga harus mempertimbangkan apakah Anda ingin berubah:

- Bucket S3 tempat artefak untuk pipa ini disimpan.
- Lokasi sumber untuk kode Anda.
- Penyedia penyebaran.
- Bagaimana Anda ingin kode Anda diterapkan.
- Tag untuk pipeline Anda.

Struktur pipa sampel dua tahap berikut menyoroti nilai yang harus Anda pertimbangkan untuk diubah untuk pipa Anda. Pipeline Anda kemungkinan berisi lebih dari dua tahap:

```
{
    "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"
    }]
}
```

Contoh ini menambahkan penandaan ke pipeline dengan menyertakan kunci `Project` tag dan `ProjectA` nilai pada pipeline. Untuk informasi selengkapnya tentang menandai sumber daya CodePipeline, lihatPenandaan pada sumber daya.

Pastikan `PollForSourceChanges` parameter dalam file JSON Anda diatur sebagai berikut:

```
                "PollForSourceChanges": "false",
```

CodePipeline menggunakan Amazon CloudWatch Events untuk mendeteksi perubahan di repositori CodeCommit sumber dan cabang atau bucket sumber S3 Anda. Langkah selanjutnya mencakup instruksi untuk membuat sumber daya ini secara manual untuk pipeline Anda. Menyetel bendera untuk `false` menonaktifkan pemeriksaan berkala, yang tidak diperlukan saat Anda menggunakan metode deteksi perubahan yang disarankan.

4. Untuk membuat tindakan build, test, atau deploy di Region yang berbeda dari pipeline, Anda harus menambahkan hal berikut ke struktur pipeline. Untuk petunjuk, lihat Menambahkan tindakan Lintas wilayah di CodePipeline.

   - Tambahkan `Region` parameter ke struktur pipeline tindakan Anda.
   - Gunakan `artifactStores` parameter untuk menentukan bucket artefak untuk setiap AWS Wilayah tempat Anda memiliki tindakan.

5. Ketika Anda puas dengan strukturnya, simpan file Anda dengan nama seperti**`pipeline.json`**.

Untuk membuat pipa

1. Jalankan create-pipeline perintah dan gunakan `--cli-input-json` parameter untuk menentukan file JSON yang Anda buat sebelumnya.

   Untuk membuat pipeline bernama *`MySecondPipeline`* dengan file JSON bernama pipeline.json yang menyertakan nama "*`MySecondPipeline`*" sebagai nilai untuk `name` di JSON, perintah Anda akan terlihat seperti berikut:

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

   > **Important**
   >
   > Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

   Perintah ini mengembalikan struktur seluruh pipeline yang Anda buat.

2. Untuk melihat pipeline, buka CodePipeline konsol dan pilih dari daftar pipeline, atau gunakan get-pipeline-state perintah. Untuk informasi selengkapnya, lihat Lihat saluran pipa dan detailnya di CodePipeline.

3. Jika Anda menggunakan CLI untuk membuat pipeline, Anda harus secara manual membuat sumber daya deteksi perubahan yang direkomendasikan untuk pipeline Anda:

   - Untuk pipeline dengan CodeCommit repositori, Anda harus membuat aturan CloudWatch Peristiwa secara manual, seperti yang dijelaskan dalam. Buat EventBridge aturan untuk CodeCommit sumber (CLI)

- Untuk pipeline dengan sumber Amazon S3, Anda harus membuat aturan dan AWS CloudTrail jejak CloudWatch Acara secara manual, seperti yang dijelaskan dalam. Menghubungkan ke tindakan sumber Amazon S3 yang menggunakan dan EventBridge AWS CloudTrail

Buat pipeline dari template statis

Anda dapat membuat pipeline di konsol yang menggunakan templat untuk mengonfigurasi pipeline dengan kode sumber dan properti yang Anda tentukan. Anda harus memberikan lokasi file sumber dan informasi tentang penyedia sumber yang akan Anda gunakan untuk tindakan Anda. Anda dapat menentukan tindakan sumber untuk Amazon ECR atau repositori pihak ketiga yang didukung oleh, seperti. CodeConnections GitHub

Template akan membuat tumpukan CloudFormation untuk pipeline Anda yang menyertakan sumber daya berikut:

- Pipa dibuat dengan tipe pipa V2. Dalam tipe Pipeline, pilih salah satu opsi berikut. Jenis pipa berbeda dalam karakteristik dan harga. Untuk informasi selengkapnya, lihat Jenis pipa.
- Peran layanan dibuat untuk pipeline Anda dan direferensikan dalam template.
- Penyimpanan artefak dibuat menggunakan penyimpanan artefak default, seperti bucket artefak S3 yang ditetapkan sebagai default, untuk pipeline yang telah Anda pilih untuk pipeline Wilayah AWS Anda.

Untuk melihat koleksi template starter open source yang digunakan untuk wizard pembuatan template statis, lihat repositori di. https://github.com/aws/codepipeline-starter-templates

Saat Anda menggunakan templat statis untuk membuat pipeline, struktur pipa dikonfigurasi di setiap templat sesuai dengan kebutuhan kasus penggunaan. Misalnya, template untuk penerapan CloudFormation digunakan sebagai contoh dalam prosedur ini. Template menghasilkan pipeline bernama DeployToCloudFormationServicedengan struktur berikut:

- Tahap build yang berisi aksi sumber dengan konfigurasi yang Anda tentukan di wizard.
- Tahap penerapan dengan tindakan penerapan dan tumpukan sumber daya terkait di. CloudFormation

Saat Anda menggunakan template statis untuk membuat pipeline, CodePipeline buat nama tahapan (source, build, staging). Nama-nama ini tidak dapat diubah. Anda dapat menggunakan nama yang

lebih spesifik (misalnya, BuildToGamma atau DeployToProd) ke tahapan yang Anda tambahkan nanti.

Langkah 1: Akses konsol

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Selamat Datang, pilih Buat pipeline.

   Jika ini adalah pertama kalinya Anda menggunakan CodePipeline, pilih Memulai.

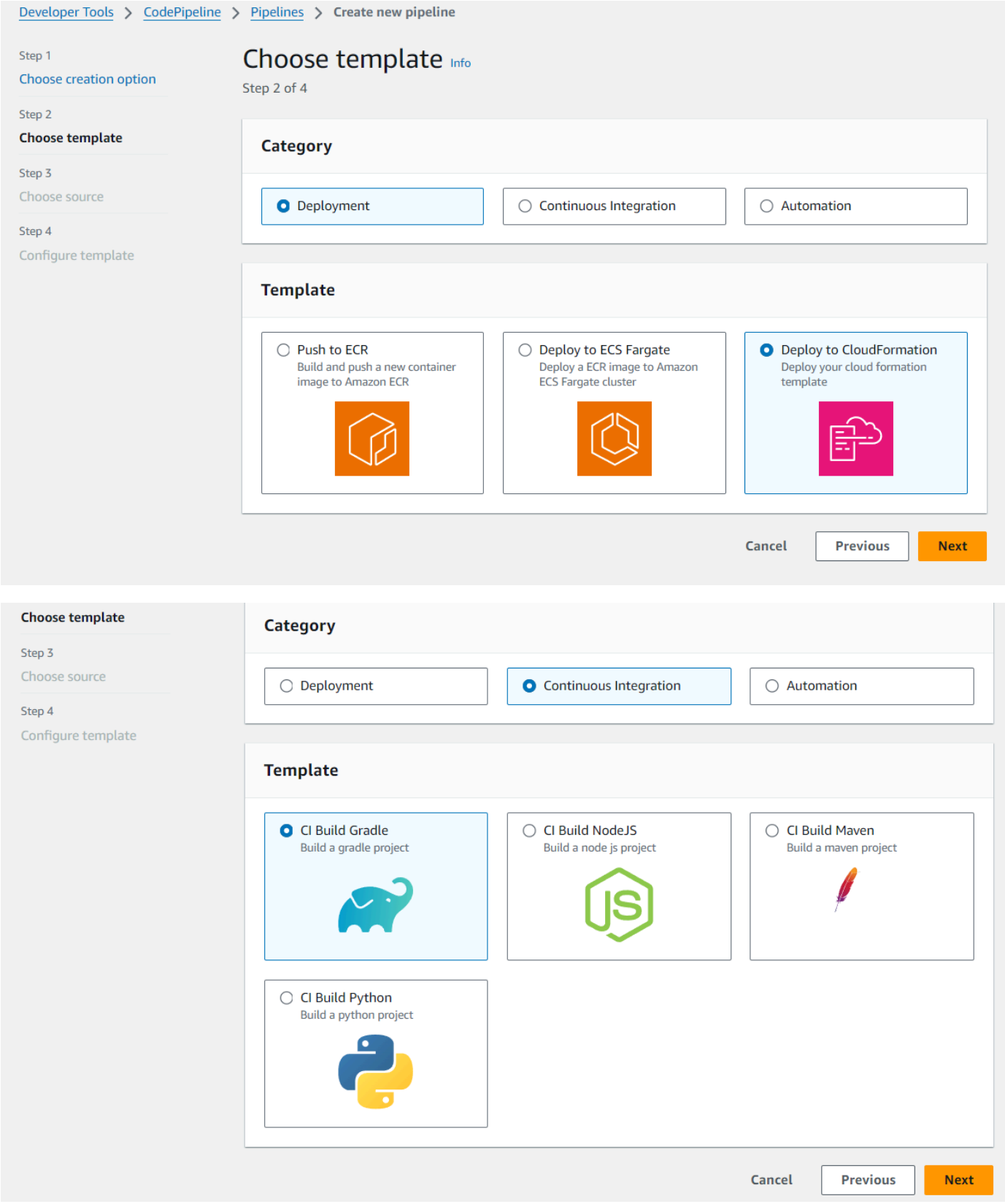
Langkah 2: Pilih Template

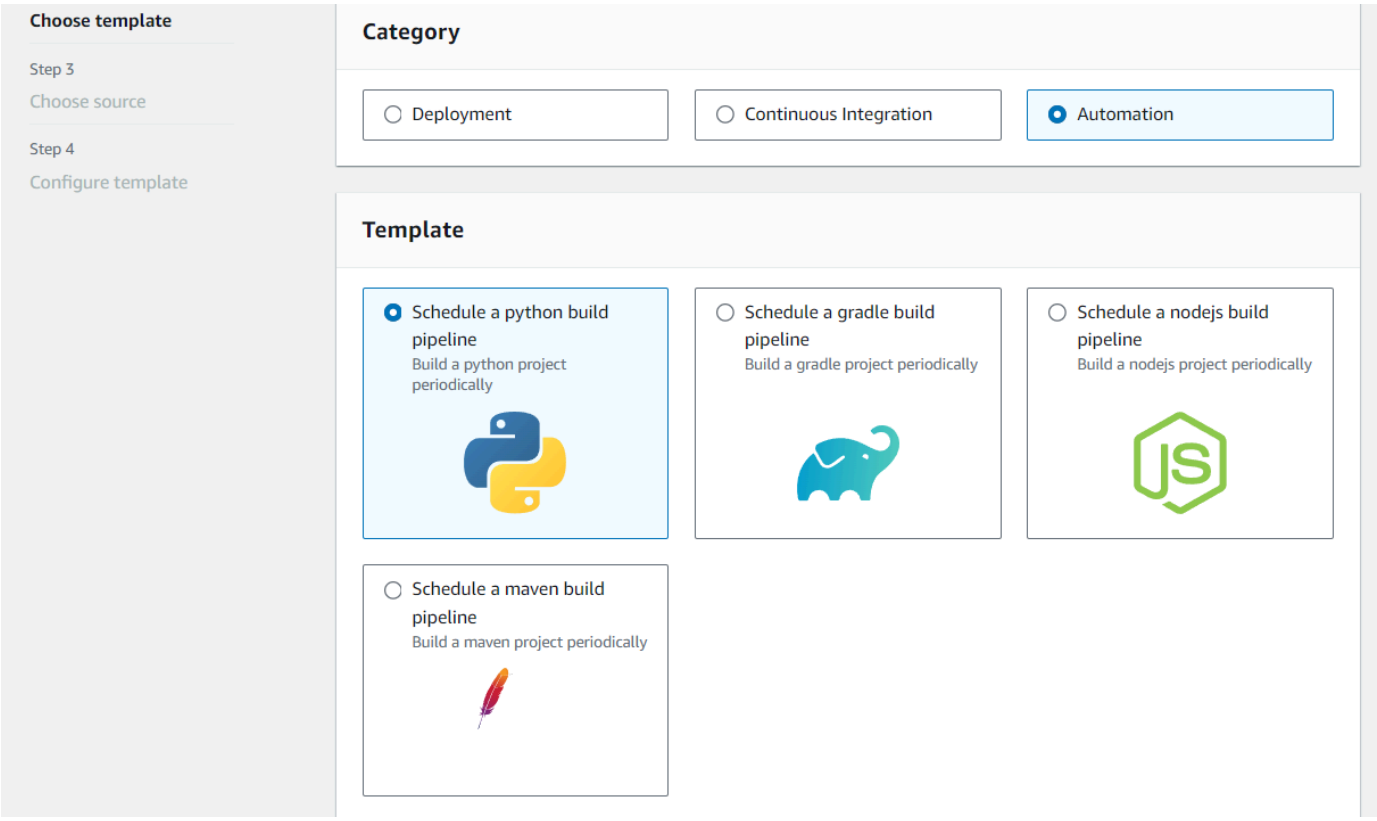
Pilih template untuk membuat pipeline dengan tahap penerapan, otomatisasi, atau pipeline CI.

1. Pada Langkah 1: Pilih halaman opsi pembuatan, lakukan salah satu hal berikut, lalu pilih Berikutnya:

   - Pilih Deployment jika Anda berencana untuk membuat tahap penerapan. Lihat opsi untuk templat yang disebarkan ke ECR atau. CloudFormation Untuk contoh ini, pilih Deployment, lalu pilih untuk menyebarkan ke. CloudFormation
   - Pilih Integrasi Berkelanjutan jika Anda berencana membuat pipeline CI. Lihat opsi untuk pipeline CI, seperti membangun ke Gradle.
   - Pilih Otomasi jika Anda berencana membuat pipeline otomatis. Lihat opsi untuk otomatisasi, seperti menjadwalkan build python.

2.


Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose creation option
Step 2
Choose template
Step 3
Choose source
Step 4
Configure template
Choose template Info
Step 2 of 4
Category
Deployment
Continuous Integration
Automation
Template
Push to ECR
Build and push a new container image to Amazon ECR
Deploy to ECS Fargate
Deploy a ECR image to Amazon ECS Fargate cluster
Deploy to CloudFormation
Deploy your cloud formation template
Cancel
Previous
Next
Choose template
Step 3
Choose source
Step 4
Configure template
Category
Deployment
Continuous Integration
Automation
Template
CI Build Gradle
Build a gradle project
CI Build NodeJS
Build a node js project
CI Build Maven
Build a maven project
CI Build Python
Build a python project
Cancel
Previous
Next



Langkah 3: Pilih sumber

- Pada Langkah 3: Pilih halaman sumber, di Penyedia sumber, pilih penyedia untuk repositori tempat kode sumber Anda disimpan, tentukan opsi yang diperlukan, lalu pilih Langkah berikutnya.

  - Untuk Bitbucket Cloud, GitHub (melalui GitHub Aplikasi), Server GitHub Perusahaan, GitLab .com, atau yang dikelola GitLab sendiri:

    1. Di bawah Koneksi, pilih koneksi yang ada atau buat yang baru. Untuk membuat atau mengelola koneksi untuk tindakan GitHub sumber Anda, lihatGitHub koneksi.
    2. Pilih repositori yang ingin Anda gunakan sebagai lokasi sumber untuk pipeline Anda.

       Pilih untuk menambahkan pemicu atau filter pada jenis pemicu untuk memulai pipeline Anda. Untuk informasi selengkapnya tentang bekerja dengan pemicu, lihatTambahkan pemicu dengan tipe acara permintaan push atau pull kode. Untuk informasi selengkapnya tentang pemfilteran dengan pola glob, lihat. Bekerja dengan pola glob dalam sintaks
    3. Dalam format artefak Output, pilih format untuk artefak Anda.

- Untuk menyimpan artefak keluaran dari GitHub tindakan menggunakan metode default, pilih CodePipelinedefault. Tindakan mengakses file dari GitHub repositori dan menyimpan artefak dalam file ZIP di toko artefak pipa.
- Untuk menyimpan file JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung, pilih klon Penuh. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.

  Jika Anda memilih opsi ini, Anda perlu memperbarui izin untuk peran layanan CodeBuild proyek Anda seperti yang ditunjukkan padaPemecahan masalah CodePipeline. Untuk tutorial yang menunjukkan cara menggunakan opsi klon Penuh, lihatTutorial: Gunakan klon lengkap dengan sumber GitHub pipa.

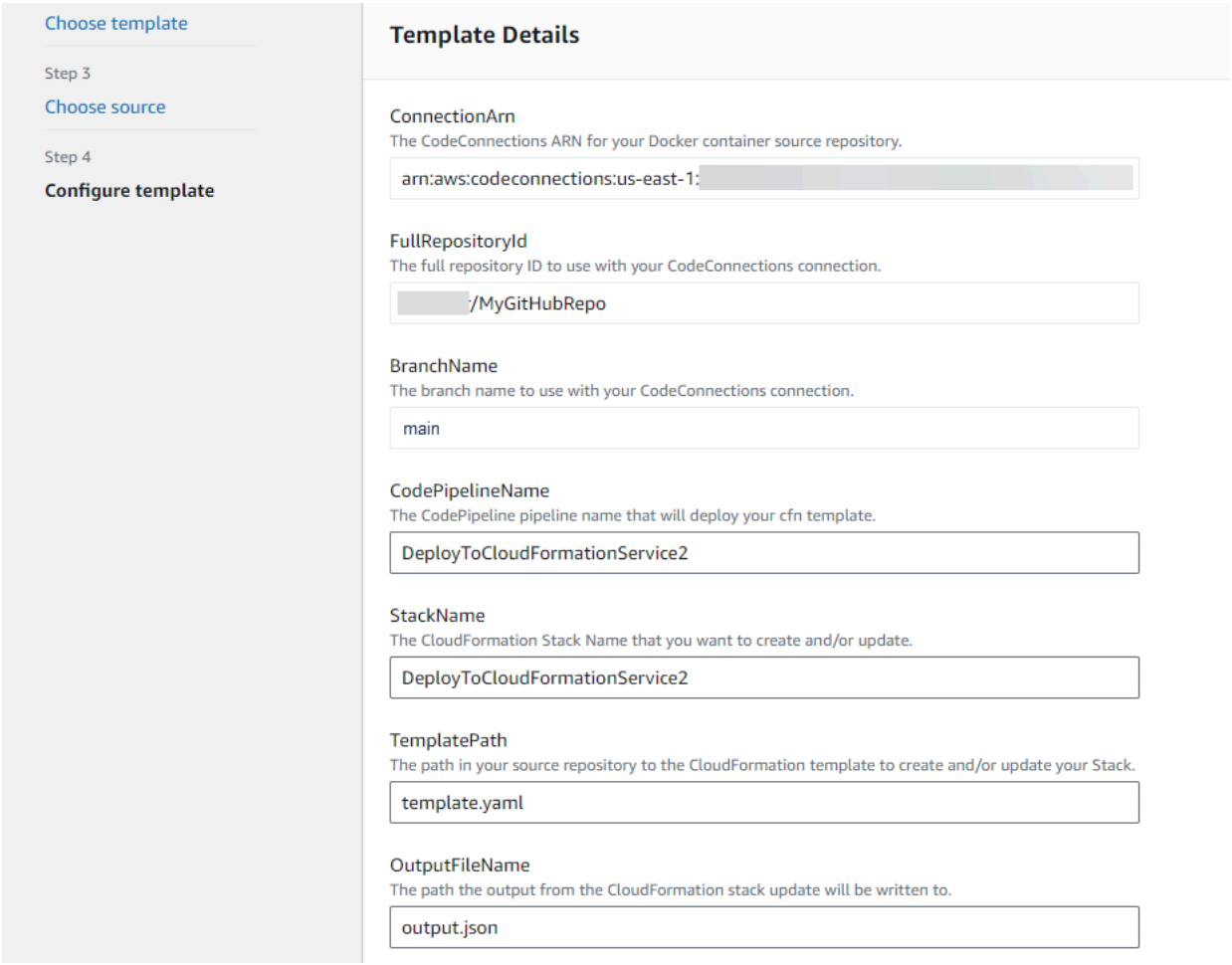
- Untuk Amazon ECR:
  - Dalam nama Repositori, pilih nama repositori Amazon ECR Anda.
  - Di tag Gambar, tentukan nama dan versi gambar, jika berbeda dari TERBARU.
  - Di artefak Output, pilih artefak keluaran default MyApp, seperti, yang berisi nama gambar dan informasi URI repositori yang ingin digunakan tahap selanjutnya.

  Saat Anda menyertakan tahap sumber ECR Amazon dalam pipeline, tindakan sumber akan menghasilkan `imageDetail.json` file sebagai artefak keluaran saat Anda melakukan perubahan. Untuk informasi tentang `imageDetail.json` file, lihatFile ImageDetail.json untuk tindakan penerapan Amazon ECS blue/green.

> **Note**
>
> Objek dan jenis file harus kompatibel dengan sistem penyebaran yang Anda rencanakan untuk digunakan (misalnya, Elastic Beanstalk atau). CodeDeploy Jenis file yang didukung mungkin termasuk file.zip, .tar, dan .tgz. Untuk informasi selengkapnya tentang jenis wadah yang didukung untuk Elastic Beanstalk, lihat Menyesuaikan dan Mengonfigurasi Lingkungan Elastic Beanstalk dan Platform yang Didukung. Untuk informasi selengkapnya tentang menerapkan revisi dengan CodeDeploy, lihat Mengunggah Revisi Aplikasi Anda dan Mempersiapkan Revisi.

Langkah 4: Konfigurasikan template

Untuk contoh ini, penyebaran ke CloudFormation dipilih. Pada langkah ini, tambahkan konfigurasi untuk template Anda.



1. Pada Langkah 4: Konfigurasikan template, dalam nama Stack, masukkan nama untuk pipeline Anda.
2. Edit kebijakan IAM placeholder untuk izin yang berlaku untuk template Anda.
3. Pilih Buat pipeline dari template
4. Sebuah pesan menampilkan yang menyatakan bahwa sumber daya pipeline Anda sedang dibuat.

Langkah 5: Lihat pipa

- Sekarang setelah Anda membuat pipeline, Anda dapat melihatnya di CodePipeline konsol dan melihat tumpukan di CloudFormation. Pipeline mulai berjalan setelah Anda membuatnya. Untuk

informasi selengkapnya, lihat Lihat saluran pipa dan detailnya di CodePipeline. Untuk informasi selengkapnya tentang membuat perubahan pada pipeline Anda, lihatEdit pipa di CodePipeline.

Edit pipa di CodePipeline

Pipeline menjelaskan proses rilis yang AWS CodePipeline ingin Anda ikuti, termasuk tahapan dan tindakan yang harus diselesaikan. Anda dapat mengedit pipeline untuk menambah atau menghapus elemen-elemen ini. Namun, saat Anda mengedit pipeline, nilai seperti nama pipeline atau metadata pipeline tidak dapat diubah.

Anda dapat mengedit jenis pipeline, variabel, dan pemicu menggunakan halaman edit pipeline. Anda juga dapat menambahkan atau mengubah tahapan dan tindakan dalam pipeline Anda.

Tidak seperti membuat pipeline, mengedit pipeline tidak menjalankan kembali revisi terbaru melalui pipeline. Jika Anda ingin menjalankan revisi terbaru melalui pipeline yang baru saja Anda edit, Anda harus menjalankannya kembali secara manual. Jika tidak, pipeline yang diedit berjalan saat berikutnya Anda membuat perubahan ke lokasi sumber yang dikonfigurasi di tahap sumber. Untuk informasi, lihat Mulai pipa secara manual.

Anda dapat menambahkan tindakan ke pipeline yang berada di AWS Wilayah yang berbeda dari pipeline Anda. Jika Layanan AWS adalah penyedia untuk suatu tindakan, dan jenis tindakan/tipe penyedia ini berada di AWS Wilayah yang berbeda dari pipeline Anda, ini adalah tindakan Lintas wilayah. Untuk informasi selengkapnya tentang tindakan lintas wilayah, lihatMenambahkan tindakan Lintas wilayah di CodePipeline.

CodePipeline menggunakan metode deteksi perubahan untuk memulai pipeline Anda saat perubahan kode sumber didorong. Metode deteksi ini didasarkan pada jenis sumber:

- CodePipeline menggunakan Amazon CloudWatch Events untuk mendeteksi perubahan di repositori CodeCommit sumber atau bucket sumber Amazon S3 Anda.

> **Note**
>
> Sumber daya deteksi perubahan dibuat secara otomatis saat Anda menggunakan konsol. Saat Anda menggunakan konsol untuk membuat atau mengedit pipeline, sumber daya tambahan dibuat untuk Anda. Jika Anda menggunakan AWS CLI untuk membuat pipa, Anda harus membuat sumber daya tambahan sendiri. Untuk informasi selengkapnya tentang membuat atau memperbarui CodeCommit pipeline, lihatBuat EventBridge aturan untuk

CodeCommit sumber (CLI). Untuk informasi selengkapnya tentang menggunakan CLI untuk membuat atau memperbarui pipeline Amazon S3, lihat. Buat EventBridge aturan untuk sumber Amazon S3 (CLI)

Topik

- Mengedit pipeline (konsol)
- Mengedit pipeline (AWS CLI)

Mengedit pipeline (konsol)

Anda dapat menggunakan CodePipeline konsol untuk menambah, mengedit, atau menghapus tahapan dalam pipeline dan untuk menambah, mengedit, atau menghapus tindakan dalam satu tahap.

Saat Anda memperbarui pipeline, selesaikan semua tindakan yang berjalan CodePipeline dengan baik dan kemudian gagal dalam tahapan dan eksekusi pipeline di mana tindakan yang sedang berjalan selesai. Saat pipeline diperbarui, Anda harus menjalankan kembali pipeline Anda. Untuk informasi selengkapnya tentang menjalankan pipeline, lihatMulai pipa secara manual.

Untuk mengedit pipeline

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda edit. Ini membuka tampilan rinci dari pipa, termasuk keadaan masing-masing tindakan di setiap tahap pipa.
3. Pada halaman detail pipeline, pilih Edit.
4. Untuk mengedit jenis pipeline, pilih Edit pada kartu Edit: Pipeline properties. Pilih salah satu opsi berikut, lalu pilih Selesai.

   - Pipa tipe V1 memiliki struktur JSON yang berisi pipa standar, tahap, dan parameter tingkat aksi.
   - Pipa tipe V2 memiliki struktur yang sama dengan tipe V1, bersama dengan dukungan parameter tambahan, seperti pemicu dan variabel tingkat pipa.

Jenis pipa berbeda dalam karakteristik dan harga. Untuk informasi selengkapnya, lihat Jenis pipa.

5. Untuk mengedit variabel pipeline, pilih Edit variabel pada kartu Edit: Variables. Tambahkan atau ubah variabel untuk level pipeline, lalu pilih Selesai.

   Untuk informasi lebih lanjut tentang variabel di tingkat pipa, lihatReferensi variabel. Untuk tutorial dengan variabel tingkat pipa yang dilewatkan pada saat eksekusi pipeline, lihat. Tutorial: Gunakan variabel tingkat pipa

   > **Note**
   >
   > Meskipun opsional untuk menambahkan variabel pada tingkat pipa, untuk pipeline yang ditentukan dengan variabel pada tingkat pipa di mana tidak ada nilai yang disediakan, eksekusi pipeline akan gagal.

6. Untuk mengedit pemicu pipeline, pilih Edit pemicu pada kartu Edit: Pemicu. Tambahkan atau ubah pemicu, lalu pilih Selesai.

   Untuk informasi selengkapnya tentang menambahkan pemicu, lihat langkah-langkah untuk membuat koneksi ke Bitbucket Cloud, GitHub (melalui GitHub Aplikasi), Server GitHub Perusahaan, GitLab .com, atau GitLab dikelola sendiri, seperti. GitHub koneksi

7. Untuk mengedit tahapan dan tindakan pada halaman Edit, lakukan salah satu hal berikut:

   - Untuk mengedit panggung, pilih Edit tahap. Anda dapat menambahkan tindakan secara serial dan paralel dengan tindakan yang ada:

     Anda juga dapat mengedit tindakan dalam tampilan ini dengan memilih ikon edit untuk tindakan tersebut. Untuk menghapus tindakan, pilih ikon hapus pada tindakan itu.
   - Untuk mengedit tindakan, pilih ikon edit untuk tindakan itu, lalu pada tindakan Edit, ubah nilainya. Item yang ditandai dengan tanda bintang (*) diperlukan.
     - Untuk nama dan cabang CodeCommit repositori, muncul pesan yang menunjukkan aturan Amazon CloudWatch Events yang akan dibuat untuk pipeline ini. Jika Anda menghapus CodeCommit sumbernya, muncul pesan yang menunjukkan aturan Amazon CloudWatch Events yang akan dihapus.
     - Untuk bucket sumber Amazon S3, muncul pesan yang menunjukkan aturan dan AWS CloudTrail jejak CloudWatch Acara Amazon yang akan dibuat untuk pipeline ini. Jika

Anda menghapus sumber Amazon S3, muncul pesan yang menunjukkan aturan dan AWS CloudTrail jejak CloudWatch Acara Amazon yang akan dihapus. Jika AWS CloudTrail jejak digunakan oleh jaringan pipa lain, jejak tidak dihapus dan peristiwa data dihapus.

- Untuk menambahkan tahap, pilih + Tambahkan tahap pada titik di pipeline tempat Anda ingin menambahkan panggung. Berikan nama untuk panggung, dan kemudian tambahkan setidaknya satu tindakan ke dalamnya. Item yang ditandai dengan tanda bintang (*) diperlukan.
- Untuk menghapus panggung, pilih ikon hapus pada tahap itu. Panggung dan semua tindakannya dihapus.
- Untuk mengonfigurasi tahap untuk memutar kembali secara otomatis pada kegagalan, pilih Edit tahap, lalu pilih kotak centang Konfigurasikan rollback otomatis pada kegagalan panggung.

Misalnya, jika Anda ingin menambahkan aksi serial ke tahap dalam pipeline:

1. Pada tahap di mana Anda ingin menambahkan tindakan Anda, pilih Edit tahap, lalu pilih + Tambahkan grup tindakan.
2. Di Edit tindakan, di Nama tindakan, masukkan nama tindakan Anda. Daftar penyedia Action menampilkan opsi penyedia berdasarkan kategori. Cari kategori (misalnya, Deploy). Di bawah kategori, pilih penyedia (misalnya, AWS CodeDeploy). Di Wilayah, pilih AWS Wilayah tempat sumber daya dibuat atau tempat Anda berencana membuatnya. Bidang Region menentukan tempat AWS sumber daya dibuat untuk tipe tindakan dan jenis penyedia ini. Bidang ini hanya menampilkan tindakan di mana penyedia tindakan adalah Layanan AWS. Bidang Region default ke AWS Region yang sama dengan pipeline Anda.

   Untuk contoh menambahkan penyedia tindakan dan menggunakan bidang default untuk setiap penyedia, lihatBuat pipeline khusus (konsol).

   Untuk menambahkan CodeBuild sebagai tindakan build atau tindakan pengujian ke tahap, lihat Menggunakan CodePipeline dengan CodeBuild untuk Menguji Kode dan Menjalankan Build di Panduan CodeBuild Pengguna.

   > **Note**
   >
   > Beberapa penyedia tindakan, seperti GitHub, mengharuskan Anda untuk terhubung ke situs web penyedia sebelum Anda dapat menyelesaikan konfigurasi tindakan. Saat

Anda terhubung ke situs web penyedia, pastikan Anda menggunakan kredensialnya untuk situs web tersebut. Jangan gunakan AWS kredensialnya.

3. Setelah selesai mengonfigurasi tindakan Anda, pilih Simpan.

> **Note**
>
> Anda tidak dapat mengganti nama panggung dalam tampilan konsol. Anda dapat menambahkan panggung dengan nama yang ingin Anda ubah, lalu hapus yang lama. Pastikan Anda telah menambahkan semua tindakan yang Anda inginkan pada tahap itu sebelum Anda menghapus yang lama.

8. Setelah selesai mengedit pipeline, pilih Simpan untuk kembali ke halaman ringkasan.

> **Important**
>
> Setelah Anda menyimpan perubahan Anda, Anda tidak dapat membatalkannya. Anda harus mengedit pipa lagi. Jika revisi berjalan melalui pipeline saat Anda menyimpan perubahan, proses tidak selesai. Jika Anda ingin komit atau perubahan tertentu dijalankan melalui pipeline yang diedit, Anda harus menjalankannya secara manual melalui pipeline. Jika tidak, komit atau perubahan berikutnya berjalan secara otomatis melalui pipeline.

9. Untuk menguji tindakan Anda, pilih Rilis perubahan untuk memproses komit itu melalui pipeline dan komit perubahan ke sumber yang ditentukan dalam tahap sumber pipeline. Atau ikuti langkah-langkah Mulai pipa secara manual untuk menggunakan AWS CLI untuk melepaskan perubahan secara manual.

Mengedit pipeline (AWS CLI)

Anda dapat menggunakan update-pipeline perintah untuk mengedit pipeline.

Saat Anda memperbarui pipeline, selesaikan semua tindakan yang berjalan CodePipeline dengan baik dan kemudian gagal dalam tahapan dan eksekusi pipeline di mana tindakan yang sedang berjalan selesai. Saat pipeline diperbarui, Anda harus menjalankan kembali pipeline Anda. Untuk informasi selengkapnya tentang menjalankan pipeline, lihatMulai pipa secara manual.

> **Important**
>
> Meskipun Anda dapat menggunakan AWS CLI untuk mengedit pipeline yang menyertakan tindakan mitra, Anda tidak boleh mengedit JSON dari tindakan mitra secara manual. Jika Anda melakukannya, tindakan mitra gagal setelah Anda memperbarui pipeline.

Untuk mengedit pipeline

1. Buka sesi terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan jalankan get-pipeline perintah untuk menyalin struktur pipeline ke file JSON. Misalnya, untuk pipeline bernama**`MyFirstPipeline`**, masukkan perintah berikut:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Perintah ini tidak mengembalikan apa pun, tetapi file yang Anda buat akan muncul di direktori tempat Anda menjalankan perintah.

2. Buka file JSON di editor teks biasa dan modifikasi struktur file untuk mencerminkan perubahan yang ingin Anda buat pada pipeline. Misalnya, Anda dapat menambah atau menghapus tahapan, atau menambahkan tindakan lain ke tahap yang ada.

   Contoh berikut menunjukkan bagaimana Anda akan menambahkan tahap penyebaran lain dalam file pipeline.json. Tahap ini berjalan setelah tahap penyebaran pertama bernama*`Staging`*.

   > **Note**
   >
   > Ini hanya sebagian dari file, bukan seluruh struktur. Untuk informasi selengkapnya, lihat CodePipeline referensi struktur pipa.

   ```
   ,
           {
               "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
                }
            ]
    }
```

```
    ]
}
```

Untuk informasi tentang penggunaan CLI untuk menambahkan tindakan persetujuan ke pipeline, lihat. Tambahkan tindakan persetujuan manual ke pipeline di CodePipeline

Pastikan `PollForSourceChanges` parameter dalam file JSON Anda diatur sebagai berikut:

```
                "PollForSourceChanges": "false",
```

CodePipeline menggunakan Amazon CloudWatch Events untuk mendeteksi perubahan di repositori CodeCommit sumber dan cabang atau bucket sumber Amazon S3 Anda. Langkah selanjutnya mencakup instruksi untuk membuat sumber daya ini secara manual. Menyetel bendera untuk `false` menonaktifkan pemeriksaan berkala, yang tidak diperlukan saat Anda menggunakan metode deteksi perubahan yang disarankan.

3. Untuk menambahkan tindakan build, test, atau deploy di Region yang berbeda dari pipeline, Anda harus menambahkan hal berikut ke struktur pipeline. Untuk petunjuk mendetail, lihat Menambahkan tindakan Lintas wilayah di CodePipeline.

   - Tambahkan `Region` parameter ke struktur pipeline tindakan Anda.
   - Gunakan `artifactStores` parameter untuk menentukan bucket artefak untuk setiap Wilayah tempat Anda memiliki tindakan.

4. Jika Anda bekerja dengan struktur pipa yang diambil menggunakan get-pipeline perintah, Anda harus memodifikasi struktur dalam file JSON. Anda harus menghapus `metadata` baris dari file sehingga update-pipeline perintah dapat menggunakannya. Hapus bagian dari struktur pipa di file JSON (`"metadata": { }`garis dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang).

   Misalnya, hapus baris berikut dari struktur:

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

   Simpan file tersebut.

5. Jika Anda menggunakan CLI untuk mengedit pipeline, Anda harus secara manual mengelola sumber daya deteksi perubahan yang disarankan untuk pipeline Anda:

   - Untuk CodeCommit repositori, Anda harus membuat aturan CloudWatch Acara, seperti yang dijelaskan dalam. Buat EventBridge aturan untuk CodeCommit sumber (CLI)
   - Untuk sumber Amazon S3, Anda harus membuat aturan dan AWS CloudTrail jejak CloudWatch Acara, seperti yang dijelaskan dalam. Menghubungkan ke tindakan sumber Amazon S3 yang menggunakan dan EventBridge AWS CloudTrail

6. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, dengan menentukan file JSON pipeline:

   > **Important**
   >
   > Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

   Perintah ini mengembalikan seluruh struktur pipa yang diedit.

   > **Note**
   >
   > update-pipelinePerintah menghentikan pipa. Jika revisi sedang dijalankan melalui pipeline saat Anda menjalankan update-pipeline perintah, proses itu dihentikan. Anda harus memulai pipeline secara manual untuk menjalankan revisi itu melalui pipeline yang diperbarui.

7. Buka CodePipeline konsol dan pilih pipeline yang baru saja Anda edit.

   Pipeline menunjukkan perubahan Anda. Lain kali Anda membuat perubahan ke lokasi sumber, pipa menjalankan revisi itu melalui struktur pipa yang direvisi.

8. Untuk menjalankan revisi terakhir secara manual melalui struktur pipa yang direvisi, jalankan perintah. start-pipeline-execution Untuk informasi selengkapnya, lihat Mulai pipa secara manual.

Untuk informasi selengkapnya tentang struktur pipeline dan nilai yang diharapkan, lihat CodePipeline referensi struktur pipa dan Referensi AWS CodePipeline API.

Lihat saluran pipa dan detailnya di CodePipeline

Anda dapat menggunakan AWS CodePipeline konsol atau AWS CLI untuk melihat detail tentang pipeline yang terkait dengan AWS akun Anda.

Topik

- Lihat saluran pipa (konsol)
- Melihat detail tindakan dalam pipeline (konsol)
- Lihat ARN pipeline dan peran layanan ARN (konsol)
- Lihat detail dan riwayat saluran pipa (CLI)
- Lihat hasil aturan untuk kondisi tahap dalam riwayat eksekusi

Lihat saluran pipa (konsol)

Anda dapat melihat status, transisi, dan pembaruan artefak untuk pipeline.

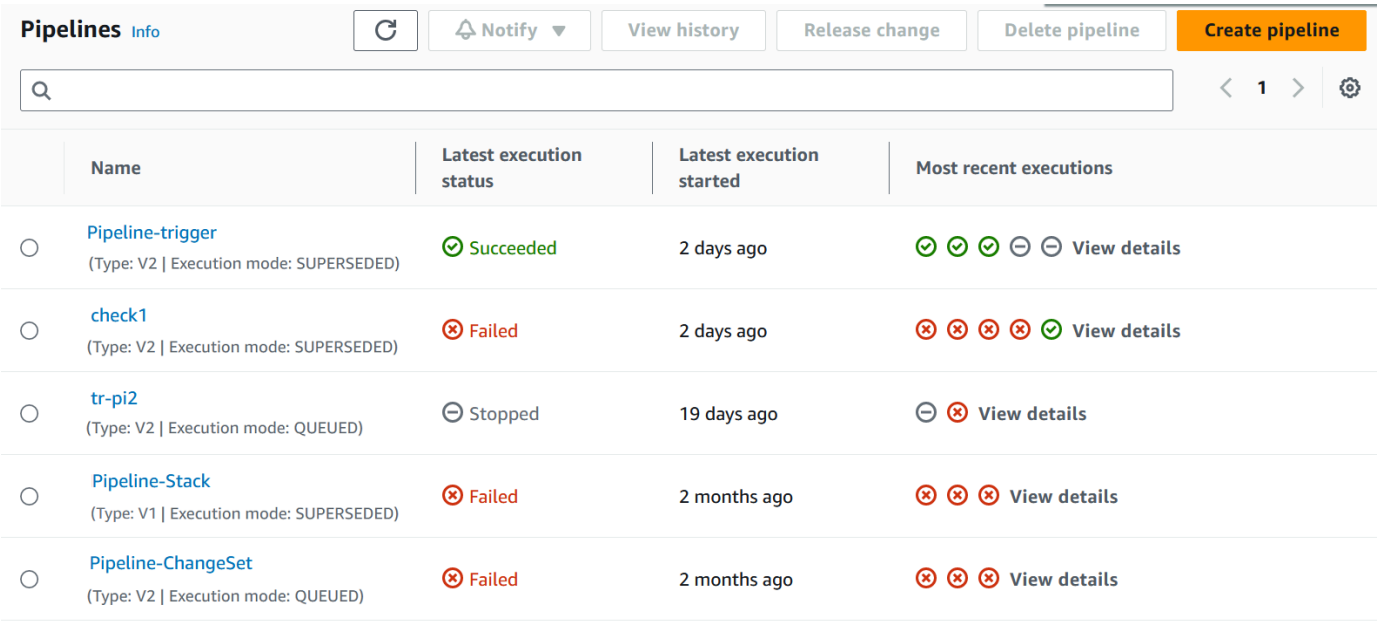
> **Note**
>
> Setelah satu jam, tampilan detail dari pipeline berhenti menyegarkan secara otomatis di browser Anda. Untuk melihat informasi terkini, segarkan halaman.

Untuk melihat jaringan pipa

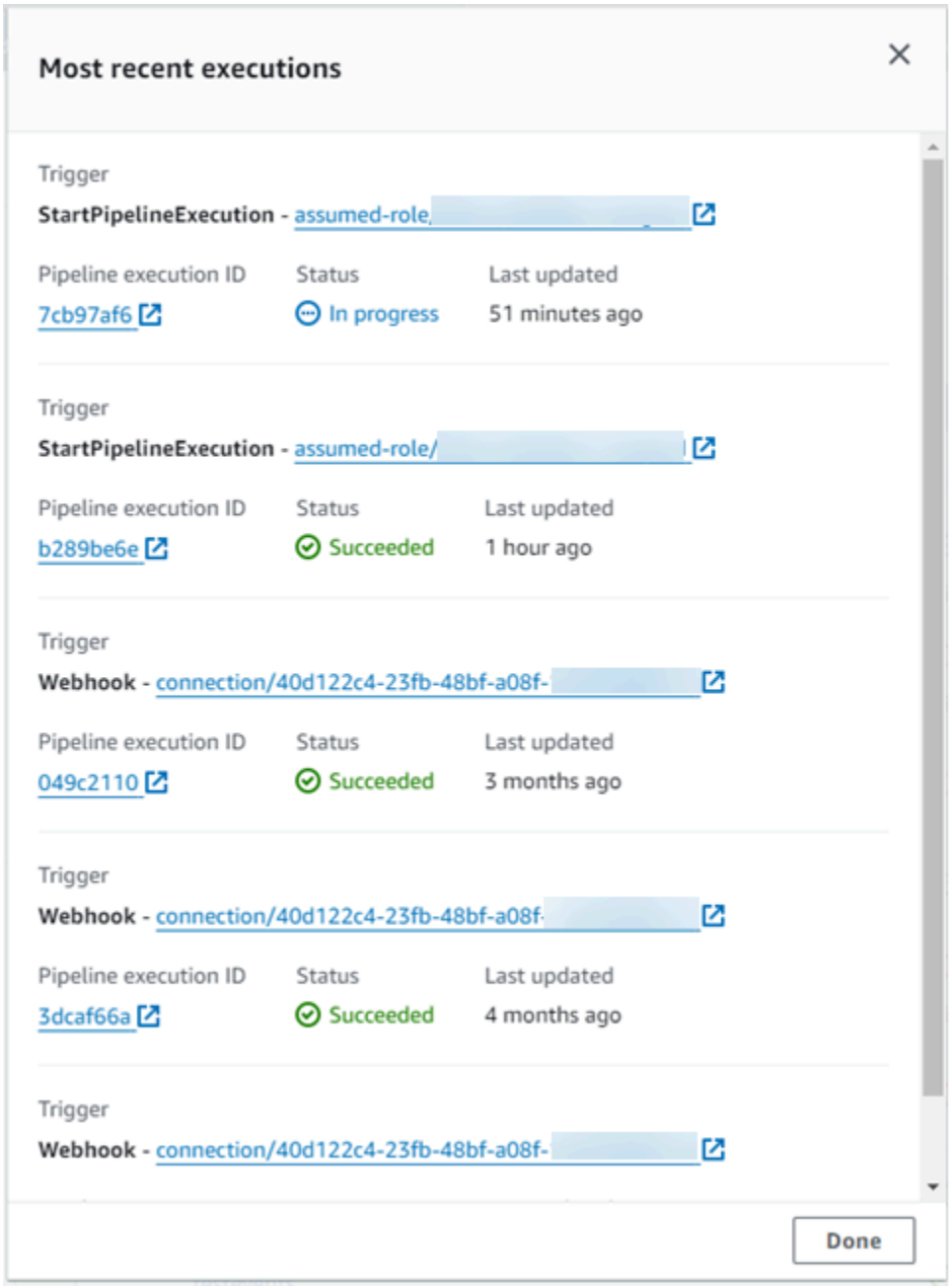
1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Halaman Pipelines ditampilkan. Daftar semua saluran pipa Anda untuk Wilayah tersebut ditampilkan.

   Nama, jenis, status, versi, tanggal pembuatan, dan tanggal ketika terakhir diubah dari semua pipeline yang terkait dengan AWS akun Anda ditampilkan, bersama dengan waktu eksekusi yang paling baru dimulai.
2. Status untuk lima eksekusi terbaru ditampilkan.



Pilih Lihat detail di samping baris tertentu untuk menampilkan kotak dialog detail yang mencantumkan eksekusi terbaru.

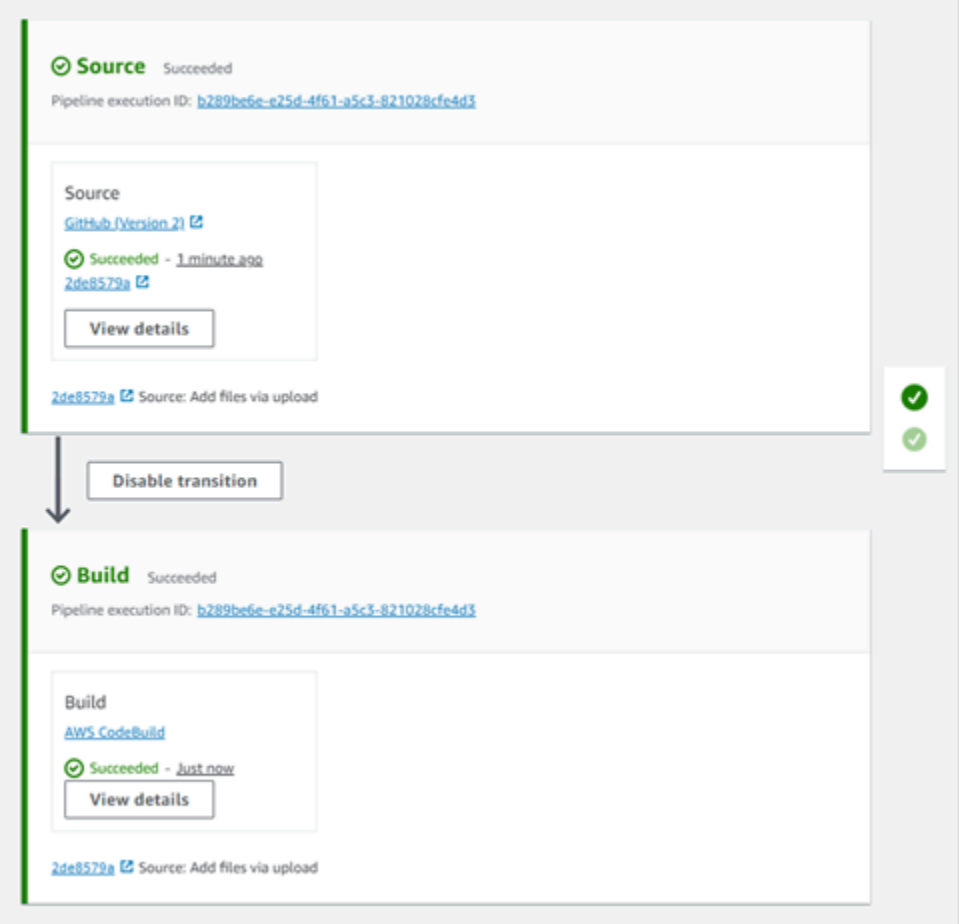


Untuk melihat detail tentang eksekusi terbaru untuk pipeline, pilih Lihat riwayat. Untuk eksekusi sebelumnya, Anda dapat melihat detail revisi yang terkait dengan artefak sumber, seperti eksekusi IDs, status, waktu mulai dan berakhir, durasi, dan komit IDs dan pesan.

> **Note**
>
> Untuk pipeline dalam mode eksekusi PARALLEL, tampilan pipeline utama tidak menunjukkan struktur pipa atau eksekusi yang sedang berlangsung. Untuk pipeline dalam mode eksekusi PARALLEL, Anda mengakses struktur pipeline dengan memilih

ID untuk eksekusi yang ingin Anda lihat dari halaman riwayat eksekusi. Pilih Riwayat di navigasi kiri, pilih ID eksekusi untuk eksekusi paralel, lalu lihat pipeline pada tab Visualisasi.

3. Untuk melihat detail untuk satu pipeline, di Nama, pilih pipeline. Tampilan rinci dari pipa, termasuk keadaan setiap tindakan di setiap tahap dan keadaan transisi, ditampilkan.



Tampilan grafis menampilkan informasi berikut untuk setiap tahap:

- Nama tahap.
- Setiap tindakan dikonfigurasi untuk panggung.
- Keadaan transisi antar tahapan (diaktifkan atau dinonaktifkan), seperti yang ditunjukkan oleh keadaan panah antar tahapan. Transisi yang diaktifkan ditunjukkan oleh panah dengan tombol Nonaktifkan transisi di sebelahnya. Transisi yang dinonaktifkan ditunjukkan oleh panah dengan coretan di bawahnya dan tombol Aktifkan transisi di sebelahnya.
- Bilah warna untuk menunjukkan status panggung:
  - Gray: Belum ada eksekusi

- Biru: Sedang berlangsung
- Hijau: Berhasil
- Merah: Gagal

Tampilan grafis juga menampilkan informasi berikut tentang tindakan di setiap tahap:

- Nama aksinya.
- Penyedia tindakan, seperti CodeDeploy.
- Saat aksi terakhir dijalankan.
- Apakah tindakan itu berhasil atau gagal.
- Tautan ke detail lain tentang tindakan terakhir, jika tersedia.
- Detail tentang revisi sumber yang berjalan melalui eksekusi pipeline terbaru di tahap atau, untuk CodeDeploy penerapan, revisi sumber terbaru yang digunakan untuk menargetkan instance.
- Tombol Lihat detail yang membuka kotak dialog dengan detail tentang eksekusi tindakan, log, dan konfigurasi tindakan.

> **Note**
>
> Tab Log tersedia untuk CodeBuild dan CloudFormation tindakan yang telah berjalan di akun pipeline.

4. Untuk melihat detail penyedia tindakan, pilih penyedia. Misalnya, di pipeline contoh sebelumnya, jika Anda memilih CodeDeploy di tahap Pementasan atau Produksi, halaman CodeDeploy konsol untuk grup penerapan yang dikonfigurasi untuk tahap tersebut akan ditampilkan.
5. Untuk melihat kemajuan suatu tindakan ditampilkan di samping tindakan yang sedang berlangsung (ditunjukkan oleh pesan Sedang Berlangsung). Jika tindakan sedang berlangsung, Anda melihat kemajuan tambahan dan langkah-langkah atau tindakan yang terjadi.
6. Untuk menyetujui atau menolak tindakan yang telah dikonfigurasi untuk persetujuan manual, pilih Tinjau.
7. Untuk mencoba lagi tindakan dalam tahap yang tidak berhasil diselesaikan, pilih Coba lagi.
8. Status dari terakhir kali tindakan dijalankan, termasuk hasil dari tindakan itu (Berhasil atau Gagal) ditampilkan.

Melihat detail tindakan dalam pipeline (konsol)

Anda dapat melihat detail untuk pipeline, termasuk detail untuk tindakan di setiap tahap.

> **Note**
>
> Setelah satu jam, tampilan detail dari pipeline berhenti menyegarkan secara otomatis di browser Anda. Untuk melihat informasi terkini, segarkan halaman.

Untuk melihat detail tindakan dalam pipeline

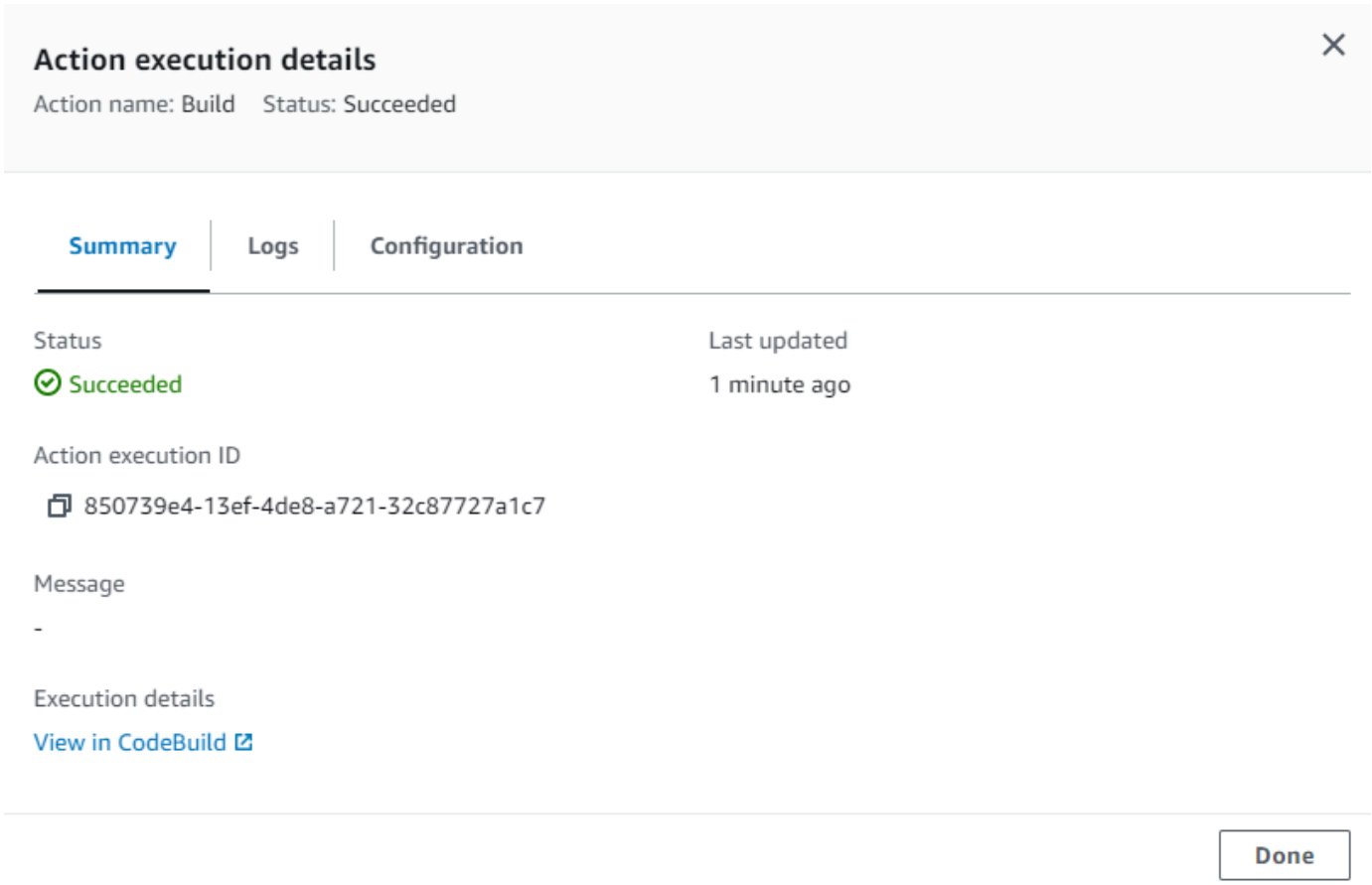
1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Halaman Pipelines ditampilkan.

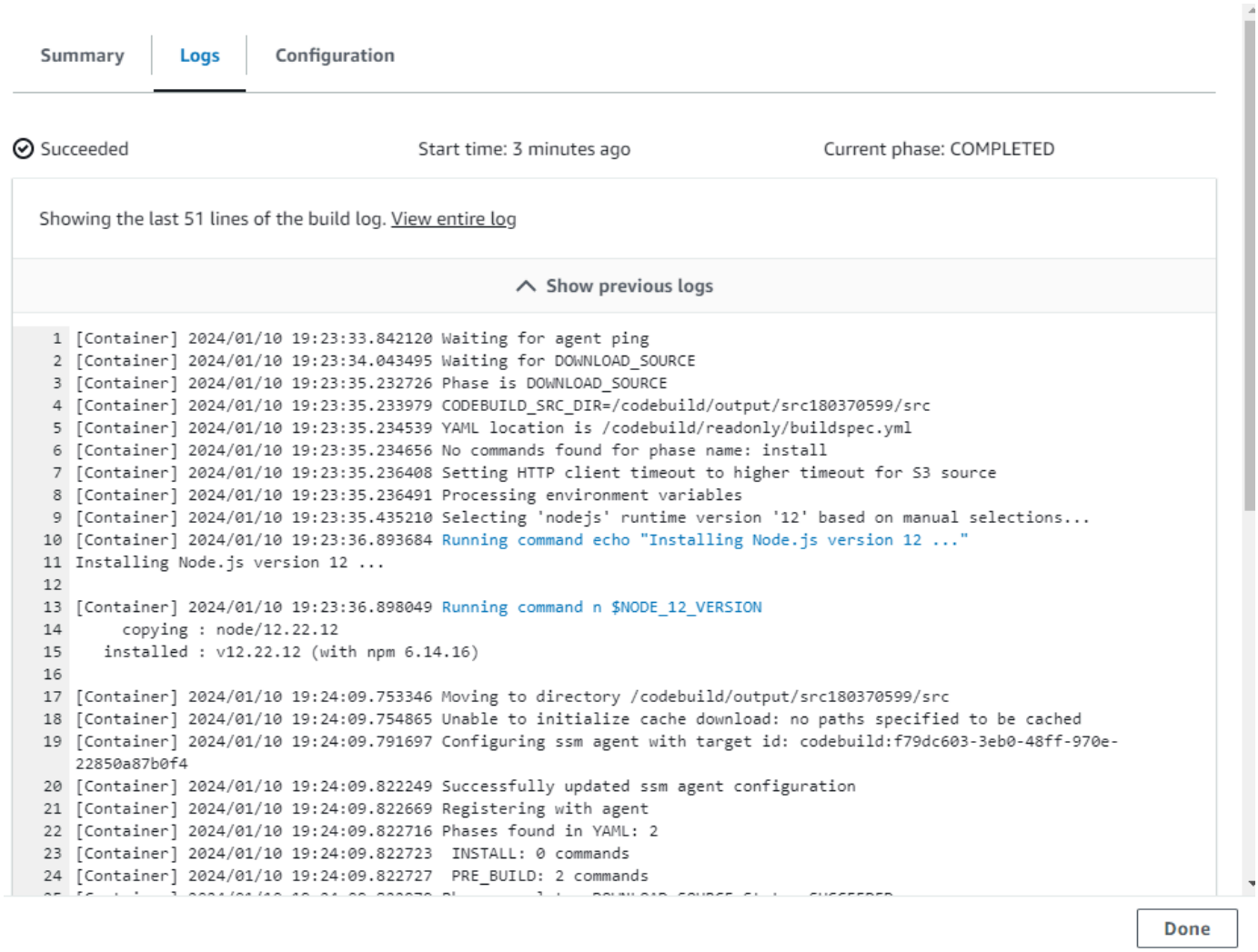
2. Pada tindakan apa pun, pilih Lihat detail untuk membuka kotak dialog dengan detail tentang eksekusi tindakan, log, dan konfigurasi tindakan.

   > **Note**
   >
   > Tab Log tersedia untuk CodeBuild dan CloudFormation tindakan.

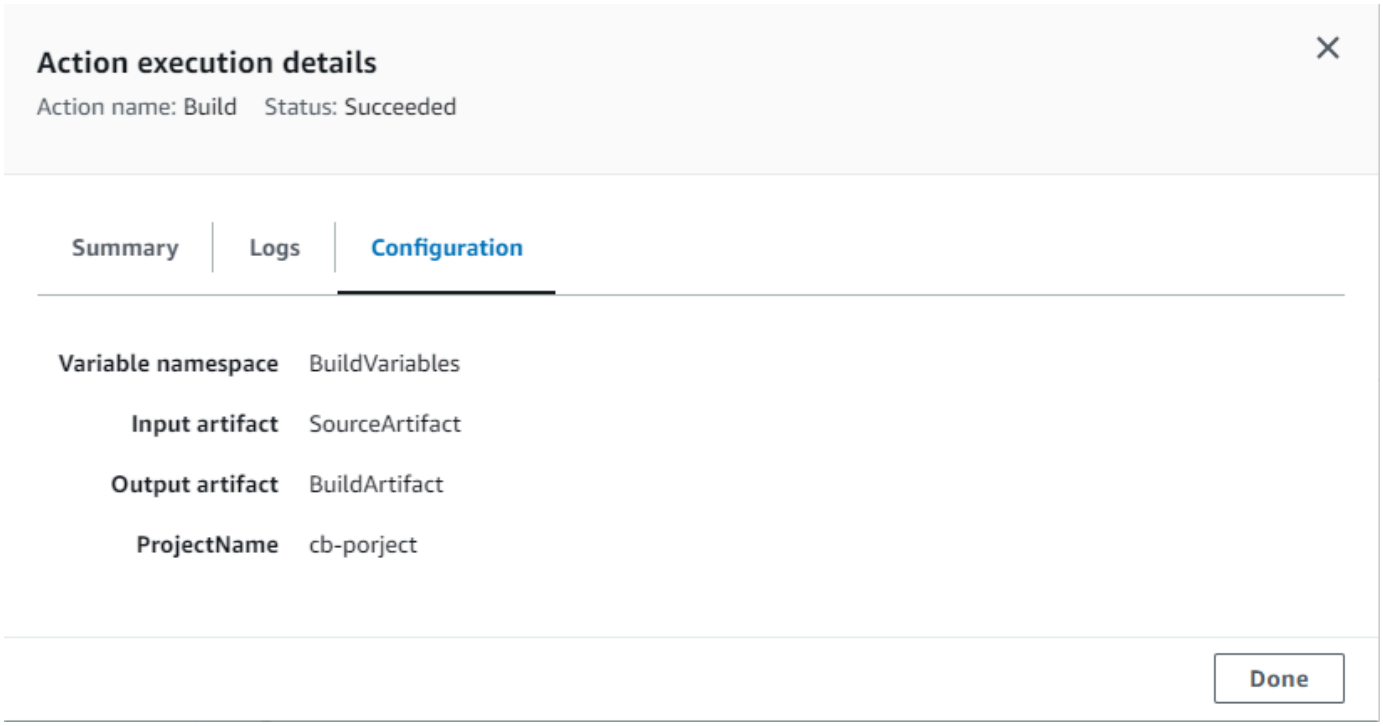
3. Untuk melihat ringkasan tindakan untuk tindakan dalam tahap pipeline, pilih Lihat detail tindakan, lalu pilih tab Ringkasan.



4. Untuk melihat log tindakan untuk tindakan dengan log, pilih Lihat detail pada tindakan, lalu pilih tab Log.



5. Untuk melihat detail konfigurasi untuk tindakan, pilih tab Konfigurasi.



Lihat ARN pipeline dan peran layanan ARN (konsol)

Anda dapat menggunakan konsol untuk melihat pengaturan pipeline, seperti ARN pipeline, ARN peran layanan, dan penyimpanan artefak pipeline.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda akan ditampilkan.
2. Pilih nama pipeline Anda, lalu pilih Pengaturan di panel navigasi sebelah kiri. Halaman ini menunjukkan yang berikut:

   - Nama pipa
   - Pipeline Nama Sumber Daya Amazon (ARN)

     ARN pipa dibangun dalam format ini:

     arn:aws:codepipeline:: *region account pipeline-name*

     Contoh pipa ARN:

     ```
     arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline
     ```

- Peran CodePipeline layanan ARN untuk pipeline Anda
- Versi pipa
- Nama dan lokasi penyimpanan artefak untuk pipa

Lihat detail dan riwayat saluran pipa (CLI)

Anda dapat menjalankan perintah berikut untuk melihat detail tentang pipeline dan eksekusi pipeline Anda:

- list-pipelinesperintah untuk melihat ringkasan semua pipeline yang terkait dengan AWS akun Anda.
- get-pipelineperintah untuk meninjau detail dari satu pipa.
- list-pipeline-executionsuntuk melihat ringkasan eksekusi terbaru untuk pipa.
- get-pipeline-executionuntuk melihat informasi tentang eksekusi pipeline, termasuk detail tentang artefak, ID eksekusi pipeline, dan nama, versi, dan status pipeline.
- get-pipeline-stateperintah untuk melihat pipeline, stage, dan status tindakan.
- list-action-executionsuntuk melihat detail eksekusi tindakan untuk pipeline.

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan perintah: list-pipelines

   ```
   aws codepipeline list-pipelines
   ```

   Perintah ini mengembalikan daftar semua pipeline yang terkait dengan AWS akun Anda.

2. Untuk melihat detail tentang pipeline, jalankan get-pipeline perintah, dengan menentukan nama unik dari pipeline. Misalnya, untuk melihat detail tentang pipeline bernama*MyFirstPipeline*, masukkan yang berikut ini:

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

   Perintah ini mengembalikan struktur pipa.

Lihat hasil aturan untuk kondisi tahap dalam riwayat eksekusi

Anda dapat melihat hasil aturan untuk eksekusi di mana kondisi tahap menjalankan aturan dan melibatkan hasil untuk tahap, seperti rollback atau kegagalan.

Nilai status yang valid untuk kondisi dan aturan adalah sebagai berikut: `InProgress | Failed | Errored | Succeeded | Cancelled | Abandoned | Overridden`

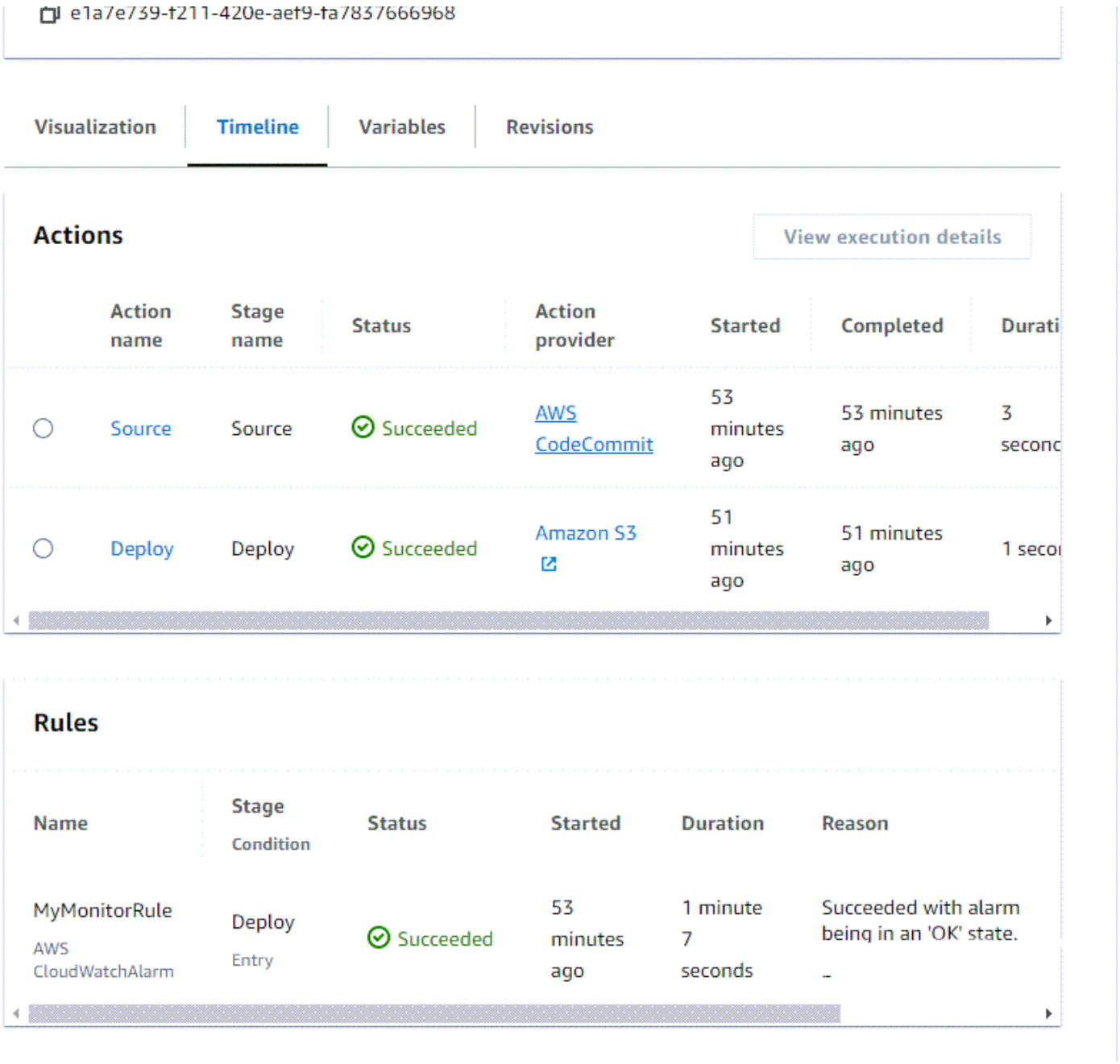
Lihat hasil aturan untuk kondisi tahap dalam riwayat eksekusi (konsol)

Anda dapat menggunakan konsol untuk melihat hasil aturan untuk eksekusi di mana kondisi tahap menjalankan aturan dan melibatkan hasil untuk panggung.

Lihat hasil aturan untuk kondisi panggung (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua saluran pipa yang terkait dengan Anda Akun AWS ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda lihat.
3. Pilih Riwayat, lalu pilih eksekusi. Pada halaman riwayat, pilih tab Timeline. Di bawah Aturan, lihat hasil aturan untuk eksekusi.



Lihat hasil aturan untuk kondisi tahap dengan **`list-rule-executions`** (CLI)

Anda dapat menggunakan CLI untuk melihat hasil aturan untuk eksekusi di mana kondisi tahap menjalankan aturan dan melibatkan hasil untuk tahap.

- Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan list-rule-executions perintah untuk pipeline bernama: *`MyPipeline`*

```
aws codepipeline list-rule-executions --pipeline-name MyFirstPipeline
```

Perintah ini mengembalikan daftar semua eksekusi aturan lengkap yang terkait dengan pipeline.

Contoh berikut menunjukkan data yang dikembalikan untuk pipeline dengan kondisi tahap di mana aturan diberi nama*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."
                }
```

```
            }
        }
```

Hapus pipa di CodePipeline

Anda selalu dapat mengedit pipeline untuk mengubah fungsinya, tetapi Anda mungkin memutuskan ingin menghapusnya. Anda dapat menggunakan AWS CodePipeline konsol atau delete-pipeline perintah di AWS CLI untuk menghapus pipa.

Topik

- Hapus pipeline (konsol)
- Hapus pipa (CLI)

Hapus pipeline (konsol)

Untuk menghapus pipeline

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda hapus.
3. Pada halaman detail pipeline, pilih Edit.
4. Pada halaman Edit, pilih Hapus.
5. Ketik **`delete`** kolom untuk mengonfirmasi, lalu pilih Hapus.

> **Important**
>
> Tindakan ini tidak dapat dibatalkan.

Hapus pipa (CLI)

Untuk menggunakan AWS CLI untuk menghapus pipeline secara manual, gunakan perintah delete-pipeline.

> **Important**
>
> Menghapus pipa tidak dapat diubah. Tidak ada kotak dialog konfirmasi. Setelah perintah dijalankan, pipeline dihapus, tetapi tidak ada sumber daya yang digunakan dalam pipeline yang dihapus. Ini membuatnya lebih mudah untuk membuat pipeline baru yang menggunakan sumber daya tersebut untuk mengotomatiskan rilis perangkat lunak Anda.

Untuk menghapus pipeline

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan delete-pipeline perintah, dengan menentukan nama pipeline yang ingin Anda hapus. Misalnya, untuk menghapus pipeline bernama*MyFirstPipeline*:

   ```
   aws codepipeline delete-pipeline --name MyFirstPipeline
   ```

   Perintah ini tidak mengembalikan apa pun.

2. Hapus sumber daya apa pun yang tidak lagi Anda butuhkan.

   > **Note**
   >
   > Menghapus pipeline tidak akan menghapus sumber daya yang digunakan dalam pipeline, seperti aplikasi Elastic Beanstalk CodeDeploy atau Elastic Beanstalk yang Anda gunakan untuk menyebarkan kode, atau, jika Anda membuat pipeline dari CodePipeline konsol, bucket Amazon S3 dibuat untuk menyimpan artefak pipeline Anda. CodePipeline Pastikan Anda menghapus sumber daya yang tidak lagi diperlukan sehingga Anda tidak dikenakan biaya untuk mereka di masa mendatang. Misalnya, saat Anda menggunakan konsol untuk membuat pipeline untuk pertama kalinya, CodePipeline buat satu bucket Amazon S3 untuk menyimpan semua artefak untuk semua pipeline Anda. Jika Anda telah menghapus semua pipeline, ikuti langkah-langkah dalam Menghapus Bucket.

Buat pipeline CodePipeline yang menggunakan sumber daya dari AWS akun lain

Anda mungkin ingin membuat pipeline yang menggunakan sumber daya yang dibuat atau dikelola oleh AWS akun lain. Misalnya, Anda mungkin ingin menggunakan satu akun untuk pipeline dan akun lainnya untuk CodeDeploy sumber daya Anda.

> **Note**
>
> Saat Anda membuat pipeline dengan tindakan dari beberapa akun, Anda harus mengonfigurasi tindakan Anda sehingga mereka masih dapat mengakses artefak dalam batasan pipeline lintas akun. Batasan berikut berlaku untuk tindakan lintas akun:
>
> - Secara umum, suatu tindakan hanya dapat mengkonsumsi artefak jika:
>   - Tindakan ini ada di akun yang sama dengan akun pipeline ATAU
>   - Artefak dibuat di akun pipeline untuk tindakan di akun lain ATAU
>   - Artefak itu diproduksi oleh aksi sebelumnya dalam akun yang sama dengan aksinya
>
>   Dengan kata lain, Anda tidak dapat meneruskan artefak dari satu akun ke akun lain jika tidak ada akun yang merupakan akun pipeline.
> - Tindakan lintas akun tidak didukung untuk jenis tindakan berikut:
>   - Jenkins membangun tindakan

Untuk contoh ini, Anda harus membuat kunci AWS Key Management Service (AWS KMS) untuk digunakan, menambahkan kunci ke pipeline, dan menyiapkan kebijakan dan peran akun untuk mengaktifkan akses lintas akun. Untuk kunci AWS KMS, Anda dapat menggunakan ID kunci, kunci ARN, atau alias ARN.

> **Note**
>
> Alias hanya dikenali di akun yang membuat kunci KMS. Untuk tindakan lintas akun, Anda hanya dapat menggunakan ID kunci atau ARN kunci untuk mengidentifikasi kunci. Tindakan lintas akun melibatkan penggunaan peran dari akun lain (accountB), sehingga menentukan ID kunci akan menggunakan kunci dari akun lain (accounTB).

Dalam panduan ini dan contohnya, *AccountA* adalah akun yang awalnya digunakan untuk membuat pipeline. Ini memiliki akses ke bucket Amazon S3 yang digunakan untuk menyimpan artefak pipa dan peran layanan yang digunakan oleh. AWS CodePipeline*AccountB*adalah akun yang awalnya digunakan untuk membuat CodeDeploy aplikasi, grup penyebaran, dan peran layanan yang digunakan oleh CodeDeploy.

*AccountA*Untuk mengedit pipeline untuk menggunakan CodeDeploy aplikasi yang dibuat oleh*AccountB*, *AccountA* harus:

- Minta ARN atau ID akun *AccountB* (dalam panduan ini, ID adalah*AccountB*). *012ID_ACCOUNT_B*
- Buat atau gunakan kunci terkelola AWS KMS pelanggan di Wilayah untuk pipeline, dan berikan izin untuk menggunakan kunci tersebut ke peran layanan (*CodePipeline_Service_Role*) dan*AccountB*.
- Buat kebijakan bucket Amazon S3 yang memberikan *AccountB* akses ke bucket Amazon S3 (misalnya,). *codepipeline-us-east-2-1234567890*
- Buat kebijakan yang memungkinkan *AccountA* untuk mengambil peran yang dikonfigurasi oleh*AccountB*, dan lampirkan kebijakan tersebut ke peran layanan (*CodePipeline_Service_Role*).
- Edit pipeline untuk menggunakan AWS KMS kunci yang dikelola pelanggan alih-alih kunci default.

*AccountB*Untuk memungkinkan akses ke sumber dayanya ke pipa yang dibuat di*AccountA*, *AccountB* harus:

- Minta ARN atau ID akun *AccountA* (dalam panduan ini, ID adalah*AccountA*). *012ID_ACCOUNT_A*
- Buat kebijakan yang diterapkan pada peran instans Amazon EC2 yang dikonfigurasi CodeDeploy agar memungkinkan akses ke bucket Amazon S3 (). *codepipeline-us-east-2-1234567890*
- Buat kebijakan yang diterapkan pada peran instans Amazon EC2 yang dikonfigurasi CodeDeploy agar memungkinkan akses ke kunci terkelola AWS KMS pelanggan yang digunakan untuk mengenkripsi artefak pipeline. *AccountA*
- Konfigurasikan dan lampirkan peran IAM (*CrossAccount_Role*) dengan kebijakan hubungan kepercayaan yang memungkinkan peran CodePipeline layanan *AccountA* untuk mengambil peran tersebut.

- Buat kebijakan yang memungkinkan akses ke sumber daya penerapan yang diperlukan oleh pipeline dan lampirkan*CrossAccount_Role*.
- Buat kebijakan yang memungkinkan akses ke bucket Amazon S3 (*codepipeline-us-east-2-1234567890*) dan lampirkan. *CrossAccount_Role*

Topik

- Prasyarat: Buat kunci enkripsi AWS KMS
- Langkah 1: Siapkan kebijakan dan peran akun
- Langkah 2: Edit pipa

Prasyarat: Buat kunci enkripsi AWS KMS

Kunci yang dikelola pelanggan khusus untuk Wilayah, seperti semua AWS KMS kunci. Anda harus membuat AWS KMS kunci terkelola pelanggan di Wilayah yang sama tempat pipeline dibuat (misalnya,`us-east-2`).

Untuk membuat kunci yang dikelola pelanggan di AWS KMS

1. Masuk ke Konsol Manajemen AWS with *AccountA* dan buka AWS KMS konsol.
2. Di sebelah kiri, pilih Kunci yang dikelola pelanggan.
3. Pilih Buat kunci. Di tombol Configure, biarkan default Symmetric dipilih dan pilih Next.
4. Di Alias, masukkan alias yang akan digunakan untuk kunci ini (misalnya,*PipelineName-Key*). Secara opsional, berikan deskripsi dan tag untuk kunci ini, lalu pilih Berikutnya.
5. Di Tentukan Izin Administratif Kunci, pilih peran atau peran yang ingin Anda lakukan sebagai administrator untuk kunci ini, lalu pilih Berikutnya.
6. Di Tentukan Izin Penggunaan Kunci, di bawah Akun Ini, pilih nama peran layanan untuk pipeline (misalnya, CodePipeline _Service_Role). Di bawah AWS Akun lain, pilih Tambahkan AWS akun lain. Masukkan ID akun ***AccountB*** untuk menyelesaikan ARN, lalu pilih Berikutnya.
7. Di Tinjau dan edit kebijakan utama, tinjau kebijakan, lalu pilih Selesai.
8. Dari daftar kunci, pilih alias kunci Anda dan salin ARN-nya (misalnya***arn:aws:kms:us-east-2:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE***,). Anda akan memerlukan ini saat mengedit pipeline dan mengonfigurasi kebijakan.

Langkah 1: Siapkan kebijakan dan peran akun

Setelah Anda membuat AWS KMS kunci, Anda harus membuat dan melampirkan kebijakan yang akan mengaktifkan akses lintas akun. Ini membutuhkan tindakan dari keduanya *AccountA* dan*AccountB*.

Topik

- Konfigurasikan kebijakan dan peran di akun yang akan membuat pipeline (AccountA)
- Konfigurasikan kebijakan dan peran di akun yang memiliki AWS resource () AccountB

Konfigurasikan kebijakan dan peran di akun yang akan membuat pipeline (*AccountA*)

Untuk membuat pipeline yang menggunakan CodeDeploy sumber daya yang terkait dengan AWS akun lain, *AccountA* harus mengonfigurasi kebijakan untuk bucket Amazon S3 yang digunakan untuk menyimpan artefak dan peran layanan. CodePipeline

Untuk membuat kebijakan bucket Amazon S3 yang memberikan akses ke accounTb (konsol)

1. Masuk ke Konsol Manajemen AWS with *AccountA* dan buka konsol Amazon S3 di. https://console.aws.amazon.com/s3/
2. Dalam daftar bucket Amazon S3, pilih bucket Amazon S3 tempat artefak untuk saluran pipa Anda disimpan. Bucket ini diberi nama`codepipeline-`*region*`-`*1234567EXAMPLE*, di mana *region* adalah AWS Wilayah tempat Anda membuat pipeline dan *1234567EXAMPLE* merupakan nomor acak sepuluh digit yang memastikan nama bucket unik (misalnya,*codepipeline-us-east-2-1234567890*).
3. Pada halaman detail untuk bucket Amazon S3, pilih Properties.
4. Di panel properti, perluas Izin, lalu pilih Tambahkan kebijakan bucket.

> **Note**
>
> Jika kebijakan sudah dilampirkan ke bucket Amazon S3 Anda, pilih Edit kebijakan bucket. Anda kemudian dapat menambahkan pernyataan dalam contoh berikut ke kebijakan yang ada. Untuk menambahkan kebijakan baru, pilih tautan, dan ikuti petunjuk di Generator AWS Kebijakan. Untuk informasi selengkapnya, lihat Ikhtisar Kebijakan IAM.

5. Di jendela Bucket Policy Editor, buat kebijakan yang memungkinkan *AccountB* akses ke artefak pipeline, dan akan memberikan *AccountB* kemampuan untuk menambahkan artefak keluaran jika suatu tindakan, seperti sumber kustom atau tindakan build, membuatnya.
6. Pilih Simpan, lalu tutup editor kebijakan.
7. Pilih Simpan untuk menyimpan izin untuk bucket Amazon S3.

Untuk membuat kebijakan untuk peran layanan untuk CodePipeline (konsol)

1. Masuk ke Konsol Manajemen AWS with *AccountA* dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran.
3. Dalam daftar peran, di bawah Nama Peran, pilih nama peran layanan untuk CodePipeline.
4. Pada tab Izin, pilih Tambahkan kebijakan inline.
5. Pilih tab JSON, dan masukkan kebijakan berikut *AccountB* untuk mengizinkan mengambil peran. Dalam contoh berikut, *012ID_ACCOUNT_B* adalah ARN untuk: *AccountB*

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": [
            "arn:aws:iam::111122223333:role/*"
        ]
    }
}
```

6. Pilih Tinjau kebijakan.
7. Di Nama, masukkan nama untuk kebijakan ini. Pilih Buat kebijakan.

Konfigurasikan kebijakan dan peran di akun yang memiliki AWS resource () *AccountB*

Saat membuat grup aplikasi, deployment, dan deployment CodeDeploy, Anda juga membuat peran instans Amazon EC2. (Peran ini dibuat untuk Anda jika Anda menggunakan panduan Run Deployment Walkthrough, tetapi Anda juga dapat membuatnya secara manual.) Untuk pipeline yang

dibuat *AccountA* untuk menggunakan CodeDeploy sumber daya yang dibuat*AccountB*, Anda harus:

- Konfigurasikan kebijakan untuk peran instance yang memungkinkannya mengakses bucket Amazon S3 tempat artefak pipeline disimpan.
- Buat peran kedua dalam *AccountB* konfigurasi untuk akses lintas akun.

  Peran kedua ini tidak hanya harus memiliki akses ke bucket Amazon S3*AccountA*, tetapi juga harus berisi kebijakan yang memungkinkan akses ke CodeDeploy sumber daya dan kebijakan hubungan kepercayaan yang memungkinkan peran CodePipeline layanan *AccountA* untuk mengambil peran tersebut.

  > **Note**
  >
  > Kebijakan ini khusus untuk menyiapkan CodeDeploy sumber daya yang akan digunakan dalam pipeline yang dibuat menggunakan AWS akun yang berbeda. AWS Sumber daya lain akan memerlukan kebijakan khusus untuk kebutuhan sumber daya mereka.

Untuk membuat kebijakan untuk peran instans Amazon EC2 yang dikonfigurasi untuk CodeDeploy (konsol)

1. Masuk ke Konsol Manajemen AWS with *AccountB* dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di panel navigasi, pilih Peran.
3. Dalam daftar peran, di bawah Nama Peran, pilih nama peran layanan yang digunakan sebagai peran instans Amazon EC2 untuk aplikasi. CodeDeploy Nama peran ini dapat bervariasi, dan lebih dari satu peran instance dapat digunakan oleh grup penyebaran. Untuk informasi selengkapnya, lihat Membuat Profil Instans IAM untuk Instans Amazon EC2 Anda.
4. Pada tab Izin, pilih Tambahkan kebijakan inline.
5. Pilih tab JSON, dan masukkan kebijakan berikut untuk memberikan akses ke bucket Amazon S3 yang digunakan *AccountA* untuk menyimpan artefak untuk pipeline (dalam contoh ini,): *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. Pilih Tinjau kebijakan.
7. Di Nama, masukkan nama untuk kebijakan ini. Pilih Buat kebijakan.
8. Buat kebijakan kedua untuk AWS KMS di ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** mana ARN kunci terkelola pelanggan dibuat *AccountA* dan dikonfigurasi *AccountB* untuk memungkinkan penggunaannya:

   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**
>
> Anda harus menggunakan ID akun *AccountA* dalam kebijakan ini sebagai bagian dari sumber ARN untuk AWS KMS kunci, seperti yang ditunjukkan di sini, atau kebijakan tidak akan berfungsi.

9. Pilih Tinjau kebijakan.
10. Di Nama, masukkan nama untuk kebijakan ini. Pilih Buat kebijakan.

Sekarang buat peran IAM yang akan digunakan untuk akses lintas akun, dan konfigurasikan sehingga peran CodePipeline layanan *AccountA* dapat mengambil peran tersebut. Peran ini harus berisi kebijakan yang memungkinkan akses ke CodeDeploy sumber daya dan bucket Amazon S3 yang digunakan untuk menyimpan artefak. *AccountA*

Untuk mengonfigurasi peran lintas akun di IAM

1. Masuk ke Konsol Manajemen AWS with *AccountB* dan buka konsol IAM di https://console.aws.amazon.com/iam.
2. Di panel navigasi, pilih Peran. Pilih Buat peran.
3. Di Pilih tipe entitas terpercaya, pilih Akun AWS lain. Di bawah Tentukan akun yang dapat menggunakan peran ini, di ID AWS Akun, masukkan ID akun untuk akun yang akan membuat pipeline di CodePipeline (*AccountA*), lalu pilih Berikutnya: Izin.

> **Important**
>
> Langkah ini menciptakan kebijakan hubungan kepercayaan antara *AccountB* dan*AccountA*. Namun, ini memberikan akses tingkat root ke akun, dan CodePipeline

merekomendasikan untuk melingkupinya ke peran CodePipeline layanan. *AccountA* Ikuti langkah 16 untuk membatasi izin.

4. Di bawah Lampirkan kebijakan izin, pilih AmazonS3 ReadOnlyAccess, lalu pilih Berikutnya: Tag.

> **Note**
>
> Ini bukan kebijakan yang akan Anda gunakan. Anda harus memilih kebijakan untuk menyelesaikan wizard.

5. Pilih Berikutnya: Tinjauan. Ketik nama untuk peran ini di Nama peran (misalnya,*CrossAccount_Role*). Anda dapat memberi nama peran ini apa pun yang Anda inginkan selama mengikuti konvensi penamaan di IAM. Pertimbangkan untuk memberi peran nama yang dengan jelas menyatakan tujuannya. Pilih Buat Peran.
6. Dari daftar peran, pilih peran yang baru saja Anda buat (misalnya,*CrossAccount_Role*) untuk membuka halaman Ringkasan peran tersebut.
7. Pada tab Izin, pilih Tambahkan kebijakan inline.
8. Pilih tab JSON, dan masukkan kebijakan berikut untuk mengizinkan akses ke CodeDeploy sumber daya:

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:GetApplicationRevision",
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource": "*"
    }
  ]
}
```

9. Pilih Tinjau kebijakan.
10. Di Nama, masukkan nama untuk kebijakan ini. Pilih Buat kebijakan.
11. Pada tab Izin, pilih Tambahkan kebijakan inline.
12. Pilih tab JSON, dan masukkan kebijakan berikut untuk mengizinkan peran ini mengambil artefak masukan dari, dan memasukkan artefak keluaran ke dalam bucket Amazon S3 di: *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. Pilih Tinjau kebijakan.
14. Di Nama, masukkan nama untuk kebijakan ini. Pilih Buat kebijakan.
15. Pada tab Izin, temukan AmazonS3 ReadOnlyAccess dalam daftar kebijakan di bawah Nama Kebijakan, dan pilih ikon hapus (X) di sebelah kebijakan. Saat diminta, pilih Lepaskan.
16. Pilih tab Trust Relationship, lalu pilih Edit kebijakan kepercayaan. Pilih opsi Tambahkan prinsipal di kolom kiri. Untuk tipe Principal, pilih IAM Roles, dan kemudian berikan ARN untuk CodePipeline peran layanan. *AccountA* Hapus **arn:aws:iam::Account_A:root** dari daftar untuk AWS Prinsipal, lalu pilih Perbarui kebijakan.

Langkah 2: Edit pipa

Anda tidak dapat menggunakan CodePipeline konsol untuk membuat atau mengedit pipeline yang menggunakan sumber daya yang terkait dengan AWS akun lain. Namun, Anda dapat menggunakan

konsol untuk membuat struktur umum pipa, dan kemudian menggunakan AWS CLI untuk mengedit pipeline dan menambahkan sumber daya tersebut. Atau, Anda dapat menggunakan struktur pipa yang ada dan menambahkan sumber daya secara manual ke dalamnya.

Untuk menambahkan sumber daya yang terkait dengan AWS akun lain (AWS CLI)

1. Di terminal (Linux, macOS, atau Unix) atau command prompt (Windows), jalankan get-pipeline perintah pada pipeline yang ingin Anda tambahkan sumber daya. Salin output perintah ke file JSON. Misalnya, untuk pipeline bernama MyFirstPipeline, Anda akan mengetik sesuatu yang mirip dengan berikut ini:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Output dikirim ke *pipeline.json* file.

2. Buka file JSON di editor teks biasa apa pun. Setelah `"type": "S3"` di toko artefak, tambahkan KMS EncryptionKey, ID, dan ketik informasi di mana *codepipeline-us-east-2-1234567890* nama bucket Amazon S3 yang digunakan untuk menyimpan artefak untuk pipeline dan merupakan ***arn:aws:kms:us-east-1:012ID_ACCOUNT_A:key/2222222-3333333-4444-556677EXAMPLE*** ARN dari kunci yang dikelola pelanggan yang baru saja Anda buat:

   ```
   {
     "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. Tambahkan tindakan penerapan di panggung untuk menggunakan CodeDeploy sumber daya yang terkait dengannya*AccountB*, termasuk `roleArn` nilai untuk peran lintas akun yang Anda buat ()*CrossAccount_Role*.

   Contoh berikut menunjukkan JSON yang menambahkan tindakan deploy bernama. *ExternalDeploy* Ini menggunakan CodeDeploy sumber daya yang dibuat *AccountB*

dalam tahap bernama*Staging*. Dalam contoh berikut, ARN untuk *AccountB* adalah: *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**
>
> Ini bukan JSON untuk seluruh pipeline, hanya struktur untuk aksi dalam satu tahap.

4. Anda harus menghapus `metadata` baris dari file sehingga update-pipeline perintah dapat menggunakannya. Hapus bagian dari struktur pipa di file JSON (`"metadata": { }`garis dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang).

   Misalnya, hapus baris berikut dari struktur:

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

Simpan file tersebut.

5. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, tentukan file JSON pipeline, mirip dengan berikut ini:

> **Important**
>
> Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

Perintah ini mengembalikan seluruh struktur pipa yang diedit.

Untuk menguji pipeline yang menggunakan sumber daya yang terkait dengan AWS akun lain

1. Di terminal (Linux, macOS, atau Unix) atau command prompt (Windows), jalankan start-pipeline-execution perintah, tentukan nama pipeline, mirip dengan yang berikut ini:

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

Untuk informasi selengkapnya, lihat Mulai pipa secara manual.

2. Masuk ke Konsol Manajemen AWS with *AccountA* dan buka CodePipeline konsol di http://console.aws.amazon. com/codesuite/codepipeline/home.

Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan.

3. Di Nama, pilih nama pipeline yang baru saja Anda edit. Ini membuka tampilan rinci dari pipa, termasuk keadaan setiap tindakan di setiap tahap pipa.
4. Perhatikan kemajuan melalui pipa. Tunggu pesan sukses pada tindakan yang menggunakan sumber daya yang terkait dengan AWS akun lain.

> **Note**
>
> Anda akan menerima kesalahan jika mencoba melihat detail tindakan saat masuk*AccountA*. Keluar, lalu masuk *AccountB* untuk melihat detail penerapan. CodeDeploy

Migrasi jaringan pemungutan suara untuk menggunakan deteksi perubahan berbasis peristiwa

AWS CodePipeline mendukung pengiriman penuh dan end-to-end berkelanjutan, yang mencakup memulai pipeline Anda setiap kali ada perubahan kode. Ada dua cara yang didukung untuk memulai pipeline Anda setelah perubahan kode: deteksi perubahan berbasis peristiwa dan polling. Sebaiknya gunakan deteksi perubahan berbasis peristiwa untuk saluran pipa.

Gunakan prosedur yang disertakan di sini untuk memigrasikan (memperbarui) jalur polling Anda ke metode deteksi perubahan berbasis peristiwa untuk pipeline Anda.

Metode deteksi perubahan berbasis peristiwa yang direkomendasikan untuk jaringan pipa ditentukan oleh sumber pipa, seperti. CodeCommit Dalam hal ini, misalnya, jalur pemungutan suara perlu bermigrasi ke deteksi perubahan berbasis peristiwa dengan. EventBridge

Cara memigrasi jaringan pipa pemungutan suara

Untuk memigrasi jaringan pemungutan suara, tentukan jalur pemungutan suara Anda dan kemudian tentukan metode deteksi perubahan berbasis peristiwa yang direkomendasikan:

- Gunakan langkah-langkah Melihat saluran pemungutan suara di akun Anda untuk menentukan jalur pemungutan suara Anda.
- Dalam tabel, temukan jenis sumber pipeline Anda, lalu pilih prosedur dengan implementasi yang ingin Anda gunakan untuk memigrasikan pipeline polling Anda. Setiap bagian berisi beberapa metode untuk migrasi, seperti menggunakan CLI atau. CloudFormation

Cara memigrasikan saluran pipa ke metode deteksi perubahan yang disarankan		
Sumber pipa	Metode deteksi berbasis peristiwa yang direkomendasikan	Prosedur migrasi
AWS CodeCommit	EventBridge (direkomendasikan).	Lihat Migrasi jaringan pemungutan suara dengan sumber CodeCommit.
Amazon S3	EventBridge dan bucket diaktifkan untuk pemberitahuan acara (disarankan).	Lihat Migrasikan jalur pemungutan suara dengan sumber S3 yang diaktifkan untuk acara.
Amazon S3	EventBridge dan sebuah AWS CloudTrail jejak.	Lihat Migrasi jaringan pemungutan suara dengan sumber dan jejak S3 CloudTrail.
GitHub (melalui GitHub Aplikasi)	Koneksi (disarankan)	Lihat Migrasikan pipeline polling untuk tindakan sumber GitHub (melalui OAuth aplikasi) ke koneksi.
GitHub (melalui OAuth aplikasi)	Webhook	Lihat Migrasikan jalur polling untuk tindakan sumber GitHub (melalui OAuth aplikasi) ke webhooks.

Important

Untuk pembaruan konfigurasi tindakan pipeline yang berlaku, seperti pipeline dengan tindakan GitHub (melalui OAuth aplikasi), Anda harus secara eksplisit menyetel `PollForSourceChanges` parameter ke false dalam konfigurasi tindakan Sumber Anda untuk menghentikan pipeline dari polling. Akibatnya, dimungkinkan untuk mengkonfigurasi pipeline secara keliru dengan deteksi perubahan berbasis peristiwa dan polling dengan, misalnya, mengonfigurasi EventBridge aturan dan juga menghilangkan parameter. `PollForSourceChanges` Ini menghasilkan eksekusi pipa duplikat, dan pipa dihitung menuju batas jumlah total jaringan pipa pemungutan suara, yang secara default jauh lebih

rendah daripada jaringan pipa berbasis peristiwa. Untuk informasi selengkapnya, lihat Kuota di AWS CodePipeline.

Melihat saluran pemungutan suara di akun Anda

Sebagai langkah pertama, gunakan salah satu skrip berikut untuk menentukan pipeline mana di akun Anda yang dikonfigurasi untuk polling. Ini adalah saluran pipa untuk bermigrasi ke deteksi perubahan berbasis peristiwa.

Melihat saluran pemungutan suara di akun Anda (skrip)

Ikuti langkah-langkah berikut untuk menggunakan skrip untuk menentukan pipeline di akun Anda yang menggunakan polling.

1. Buka jendela terminal, lalu lakukan salah satu hal berikut:

   - Jalankan perintah berikut untuk membuat skrip baru bernama PollingPipelinesExtractor.sh.

     ```
     vi PollingPipelinesExtractor.sh
     ```

   - Untuk menggunakan skrip python, jalankan perintah berikut untuk membuat skrip python baru bernama PollingPipelinesExtractor .py.

     ```
     vi PollingPipelinesExtractor.py
     ```

2. Salin dan tempel kode berikut ke dalam PollingPipelinesExtractorskrip. Lakukan salah satu tindakan berikut:

   - Salin dan tempel kode berikut ke dalam PollingPipelinesExtractorskrip.sh.

     ```
     #!/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."
```

- Salin dan tempel kode berikut ke dalam PollingPipelinesExtractorskrip.py.

```
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. Untuk setiap Wilayah di mana Anda memiliki saluran pipa, Anda harus menjalankan skrip untuk Wilayah itu. Untuk menjalankan skrip, lakukan salah satu hal berikut:

   - Jalankan perintah berikut untuk menjalankan skrip bernama PollingPipelinesExtractor.sh. Dalam contoh ini, Region adalah us-west-2.

     ```
     ./PollingPipelinesExtractor.sh us-west-2
     ```

   - Untuk skrip python, jalankan perintah berikut untuk menjalankan skrip python bernama PollingPipelinesExtractor .py. Dalam contoh ini, Region adalah us-west-2.

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

   Dalam contoh keluaran berikut dari skrip, Region us-west-2 mengembalikan daftar pipeline polling dan menunjukkan waktu eksekusi terakhir untuk setiap 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...%
   ```

   Analisis output skrip dan, untuk setiap pipeline dalam daftar, perbarui sumber polling ke metode deteksi perubahan berbasis peristiwa yang direkomendasikan.

   > **Note**
   >
   > Saluran pemungutan suara Anda ditentukan oleh konfigurasi tindakan pipa untuk parameter tersebut. `PollForSourceChanges` Jika konfigurasi sumber pipeline menghilangkan `PollForSourceChanges` parameter, maka CodePipeline default

untuk melakukan polling repositori Anda untuk perubahan sumber. Perilaku ini sama seperti jika `PollForSourceChanges` disertakan dan disetel ke true. Untuk informasi selengkapnya, lihat parameter konfigurasi untuk tindakan sumber pipeline Anda, seperti parameter konfigurasi tindakan sumber Amazon S3. Referensi tindakan sumber Amazon S3

Perhatikan bahwa skrip ini juga menghasilkan file.csv yang berisi daftar pipeline polling di akun Anda dan menyimpan file.csv ke folder kerja saat ini.

Migrasi jaringan pemungutan suara dengan sumber CodeCommit

Anda dapat memigrasikan pipeline polling yang akan digunakan EventBridge untuk mendeteksi perubahan di repositori CodeCommit sumber atau bucket sumber Amazon S3.

CodeCommit-- Untuk pipa dengan CodeCommit sumber, modifikasi pipa sehingga deteksi perubahan otomatis EventBridge. Pilih dari metode berikut untuk menerapkan migrasi:

- Konsol: Migrasikan saluran pemungutan suara (atau sumber Amazon CodeCommit S3) (konsol)
- CLI: Migrasi jalur pemungutan suara (CodeCommit sumber) (CLI)
- CloudFormation: Migrasikan jalur pemungutan suara (CodeCommit sumber) (templat)CloudFormation

Migrasikan saluran pemungutan suara (atau sumber Amazon CodeCommit S3) (konsol)

Anda dapat menggunakan CodePipeline konsol untuk memperbarui pipeline yang akan digunakan EventBridge untuk mendeteksi perubahan di repositori CodeCommit sumber atau bucket sumber Amazon S3.

> **Note**
>
> Saat Anda menggunakan konsol untuk mengedit pipeline yang memiliki repositori CodeCommit sumber atau bucket sumber Amazon S3, aturan dan peran IAM akan dibuat untuk Anda. Jika Anda menggunakan AWS CLI untuk mengedit pipeline, Anda harus

membuat EventBridge aturan dan peran IAM sendiri. Untuk informasi selengkapnya, lihat CodeCommit tindakan sumber dan EventBridge.

Gunakan langkah-langkah ini untuk mengedit pipeline yang menggunakan pemeriksaan berkala. Jika Anda ingin membuat pipeline, lihatBuat pipeline, tahapan, dan tindakan.

Untuk mengedit tahap sumber pipa

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda edit. Ini membuka tampilan rinci dari pipa, termasuk keadaan masing-masing tindakan di setiap tahap pipa.
3. Pada halaman detail pipeline, pilih Edit.
4. Di tahap Edit, pilih ikon edit pada aksi sumber.
5. Perluas Opsi Deteksi Ubah dan pilih Gunakan CloudWatch Acara untuk memulai pipeline saya secara otomatis saat terjadi perubahan (disarankan).

   Sebuah pesan muncul yang menunjukkan EventBridge aturan yang akan dibuat untuk pipeline ini. Pilih Perbarui.

   Jika Anda memperbarui pipeline yang memiliki sumber Amazon S3, Anda akan melihat pesan berikut. Pilih Perbarui.
6. Setelah selesai mengedit pipeline, pilih Simpan perubahan pipeline untuk kembali ke halaman ringkasan.

   Pesan menampilkan nama EventBridge aturan yang akan dibuat untuk pipeline Anda. Jangan pilih Save and continue (Simpan dan lanjutkan).
7. Untuk menguji tindakan Anda, lepaskan perubahan dengan menggunakan AWS CLI to commit perubahan ke sumber yang ditentukan dalam tahap sumber pipeline.

Migrasi jalur pemungutan suara (CodeCommit sumber) (CLI)

Ikuti langkah-langkah ini untuk mengedit pipeline yang menggunakan polling (pemeriksaan berkala) untuk menggunakan EventBridge aturan untuk memulai pipeline. Jika Anda ingin membuat pipeline, lihatBuat pipeline, tahapan, dan tindakan.

Untuk membangun pipeline berbasis peristiwa CodeCommit, Anda mengedit `PollForSourceChanges` parameter pipeline Anda dan kemudian membuat sumber daya berikut:

- EventBridge acara
- Peran IAM untuk memungkinkan acara ini memulai pipeline Anda

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

1. Jalankan get-pipeline perintah untuk menyalin struktur pipa ke file JSON. Misalnya, untuk pipeline bernama`MyFirstPipeline`, jalankan perintah berikut:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Perintah ini tidak mengembalikan apa pun, tetapi file yang Anda buat akan muncul di direktori tempat Anda menjalankan perintah.

2. Buka file JSON di editor teks biasa dan edit tahap sumber dengan mengubah `PollForSourceChanges` parameter menjadi`false`, seperti yang ditunjukkan dalam contoh ini.

   Mengapa saya membuat perubahan ini? Mengubah parameter ini untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

```
"configuration": {
    "PollForSourceChanges": "false",
    "BranchName": "main",
    "RepositoryName": "MyTestRepo"
},
```

3. Jika Anda bekerja dengan struktur pipa yang diambil menggunakan get-pipeline perintah, hapus `metadata` baris dari file JSON. Jika tidak, update-pipeline perintah tidak dapat menggunakannya. Hapus `"metadata": { }` garis dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang.

   Misalnya, hapus baris berikut dari struktur:

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

   Simpan file tersebut.

4. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, dengan menentukan file JSON pipeline:

   Important

   Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

   Perintah ini mengembalikan seluruh struktur pipa yang diedit.

   Note

   update-pipelinePerintah menghentikan pipa. Jika revisi sedang dijalankan melalui pipeline saat Anda menjalankan update-pipeline perintah, proses itu dihentikan. Anda harus memulai pipeline secara manual untuk menjalankan revisi itu melalui pipeline yang

diperbarui. Gunakan **`start-pipeline-execution`** perintah untuk memulai pipeline Anda secara manual.

Untuk membuat EventBridge aturan dengan CodeCommit sebagai sumber acara dan CodePipeline sebagai target

1. Tambahkan izin EventBridge untuk digunakan CodePipeline untuk menjalankan aturan. Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis sumber daya untuk Amazon. EventBridge

   a. Gunakan contoh berikut untuk membuat kebijakan kepercayaan yang memungkinkan EventBridge untuk mengambil peran layanan. Sebutkan kebijakan kepercayaan`trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Gunakan perintah berikut untuk membuat `Role-for-MyRule` peran dan melampirkan kebijakan kepercayaan.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Buat kebijakan izin JSON, seperti yang ditunjukkan dalam contoh ini, untuk pipeline bernama. `MyFirstPipeline` Beri nama kebijakan `permissionspolicyforEB.json` izin.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Gunakan perintah berikut untuk melampirkan kebijakan `CodePipeline-Permissions-Policy-for-EB` izin ke `Role-for-MyRule` peran.

   Mengapa saya membuat perubahan ini? Menambahkan kebijakan ini ke peran akan membuat izin untuk EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Panggil put-rule perintah dan sertakan`--name`,`--event-pattern`, dan `--role-arn` parameter.

   Mengapa saya membuat perubahan ini? Perintah ini CloudFormation memungkinkan untuk membuat acara.

   Perintah contoh berikut membuat aturan yang disebut`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. Untuk menambahkan CodePipeline sebagai target, panggil put-targets perintah dan sertakan parameter berikut:

   - `--rule`Parameter digunakan dengan yang `rule_name` Anda buat dengan menggunakanput-rule.
   - `--targets`Parameter digunakan dengan `Id` daftar target dalam daftar target dan pipa target. `ARN`

   Contoh perintah berikut menentukan bahwa untuk aturan yang dipanggil`MyCodeCommitRepoRule`, target `Id` terdiri dari nomor satu, menunjukkan bahwa dalam daftar target untuk aturan, ini adalah target 1. Perintah sample juga menentukan contoh ARN untuk pipeline. Pipeline dimulai ketika sesuatu berubah di repositori.

   ```
   aws events put-targets --rule MyCodeCommitRepoRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

4. (Opsional) Untuk mengonfigurasi transformator input dengan penggantian sumber untuk ID gambar tertentu, gunakan JSON berikut dalam perintah CLI Anda. Contoh berikut mengonfigurasi penggantian di mana:

   - `Source`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `actionName`
   - `COMMIT_ID`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `revisionType`
   - The`revisionValue`, **<** *`revisionValue`* **>** dalam contoh ini, berasal dari variabel peristiwa sumber.

   ```
   {
       "Rule": "my-rule",
       "Targets": [
           {
               "Id": "MyTargetId",
               "Arn": "pipeline-ARN",
               "InputTransformer": {
                   "sourceRevisions": {
                       "actionName": "Source",
                       "revisionType": "COMMIT_ID",
                       "revisionValue": "<revisionValue>"
   ```

```
                },
                "variables": [
                    {
                        "name": "Branch_Name",
                        "value": "value"
                    }
                ]
            }
        }
    ]
}
```

Migrasikan jalur pemungutan suara (CodeCommit sumber) (templat)CloudFormation

Untuk membangun pipeline berbasis peristiwa AWS CodeCommit, Anda mengedit `PollForSourceChanges` parameter pipeline Anda dan kemudian menambahkan sumber daya berikut ke template Anda:

- Sebuah EventBridge aturan
- Peran IAM untuk aturan Anda EventBridge

Jika Anda menggunakan CloudFormation untuk membuat dan mengelola pipeline Anda, template Anda menyertakan konten seperti berikut ini.

> **Note**
>
> `Configuration`Properti dalam tahap sumber disebut`PollForSourceChanges`. Jika properti itu tidak disertakan dalam template Anda, maka `PollForSourceChanges` diatur ke secara `true` default.

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
        }]
    },
```

Untuk memperbarui CloudFormation template pipeline Anda dan membuat EventBridge aturan

1. Di template, di bawah`Resources`, gunakan `AWS::IAM::Role` CloudFormation sumber daya untuk mengonfigurasi peran IAM yang memungkinkan acara Anda memulai pipeline. Entri ini membuat peran yang menggunakan dua kebijakan:

   - Kebijakan pertama memungkinkan peran diasumsikan.
   - Kebijakan kedua memberikan izin untuk memulai pipeline.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::IAM::Role` sumber daya memungkinkan CloudFormation untuk membuat izin untuk EventBridge. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. Dalam template, di bawah`Resources`, gunakan `AWS::Events::Rule` CloudFormation sumber daya untuk menambahkan EventBridge aturan. Pola acara ini membuat acara yang memantau perubahan push ke repositori Anda. Saat EventBridge mendeteksi perubahan status repositori, aturan akan muncul di pipeline target Anda`StartPipelineExecution`.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::Events::Rule` sumber daya memungkinkan CloudFormation untuk membuat acara. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. (Opsional) Untuk mengonfigurasi transformator input dengan penggantian sumber untuk ID gambar tertentu, gunakan cuplikan YAMAL berikut. Contoh berikut mengonfigurasi penggantian di mana:

   - `Source`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `actionName`
   - `COMMIT_ID`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `revisionType`
   - The`revisionValue`, **<** *`revisionValue`* **>** dalam contoh ini, berasal dari variabel peristiwa sumber.
   - Variabel output untuk `BranchName` dan `Value` ditentukan.

```
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
    sourceRevisions:
      actionName: Source
      revisionType: COMMIT_ID
      revisionValue: <revisionValue>
    variables:
    - name: BranchName
      value: value
```

4. Simpan template yang diperbarui ke komputer lokal Anda, lalu buka CloudFormation konsol.
5. Pilih tumpukan Anda, lalu pilih Create Change Set for Current Stack.
6. Unggah template, lalu lihat perubahan yang tercantum di dalamnya CloudFormation. Ini adalah perubahan yang harus dilakukan pada tumpukan. Anda harus melihat sumber daya baru Anda dalam daftar.
7. Pilih Eksekusi.

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Dalam banyak kasus, `PollForSourceChanges` parameter default ke true saat Anda membuat pipeline. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

- Dalam template, ubah `PollForSourceChanges` ke`false`. Jika Anda tidak menyertakan `PollForSourceChanges` dalam definisi pipeline Anda, tambahkan dan setel ke`false`.

  Mengapa saya membuat perubahan ini? Mengubah parameter ini untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

  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

Saat Anda membuat sumber daya ini CloudFormation, pipeline Anda dipicu saat file di repositori dibuat atau diperbarui. Berikut adalah cuplikan template terakhir:

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
                            }
                        ]
                    },

...
```

Migrasikan jalur pemungutan suara dengan sumber S3 yang diaktifkan untuk acara

Untuk pipeline dengan sumber Amazon S3, modifikasi pipeline sehingga deteksi perubahan otomatis melalui EventBridge dan dengan bucket sumber yang diaktifkan untuk pemberitahuan peristiwa. Ini adalah metode yang direkomendasikan jika Anda menggunakan CLI atau CloudFormation untuk memigrasikan pipeline Anda.

> **Note**
>
> Ini termasuk menggunakan bucket yang diaktifkan untuk pemberitahuan acara, di mana Anda tidak perlu membuat CloudTrail jejak terpisah. Jika Anda menggunakan konsol, maka aturan acara dan CloudTrail jejak disiapkan untuk Anda. Untuk langkah-langkah tersebut, lihatMigrasi jaringan pemungutan suara dengan sumber dan jejak S3 CloudTrail.

- CLI: Migrasi jaringan pemungutan suara dengan sumber dan jejak CloudTrail S3 (CLI)
- CloudFormation: Migrasikan jalur pemungutan suara dengan sumber dan CloudTrail jejak S3 (templat)CloudFormation

Migrasikan jalur pemungutan suara dengan sumber S3 yang diaktifkan untuk acara (CLI)

Ikuti langkah-langkah ini untuk mengedit pipeline yang menggunakan polling (pemeriksaan berkala) untuk menggunakan acara sebagai EventBridge gantinya. Jika Anda ingin membuat pipeline, lihatBuat pipeline, tahapan, dan tindakan.

Untuk membuat pipeline berbasis peristiwa dengan Amazon S3, Anda mengedit `PollForSourceChanges` parameter pipeline, lalu membuat sumber daya berikut:

- EventBridge aturan acara
- Peran IAM untuk memungkinkan EventBridge acara memulai pipeline Anda

Untuk membuat EventBridge aturan dengan Amazon S3 sebagai sumber peristiwa dan CodePipeline sebagai target dan menerapkan kebijakan izin

1. Berikan izin EventBridge untuk digunakan CodePipeline untuk menjalankan aturan. Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis sumber daya untuk Amazon. EventBridge

   a. Gunakan contoh berikut untuk membuat kebijakan kepercayaan agar memungkinkan EventBridge untuk mengambil peran layanan. Nama itu`trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Gunakan perintah berikut untuk membuat `Role-for-MyRule` peran dan melampirkan kebijakan kepercayaan.

      Mengapa saya membuat perubahan ini? Menambahkan kebijakan kepercayaan ini ke peran akan membuat izin untuk EventBridge.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Buat kebijakan izin JSON, seperti yang ditunjukkan di sini untuk pipeline bernama. `MyFirstPipeline` Beri nama kebijakan `permissionspolicyforEB.json` izin.

      JSON

      ```
      {
      ```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Gunakan perintah berikut untuk melampirkan kebijakan `CodePipeline-Permissions-Policy-for-EB` izin baru ke `Role-for-MyRule` peran yang Anda buat.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Panggil put-rule perintah dan sertakan`--name`,`--event-pattern`, dan `--role-arn` parameter.

   Perintah contoh berikut membuat aturan bernama`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. Untuk menambahkan CodePipeline sebagai target, panggil put-targets perintah dan sertakan `--targets` parameter `--rule` dan.

   Perintah berikut menentukan bahwa untuk aturan bernama`EnabledS3SourceRule`, target `Id` terdiri dari nomor satu, menunjukkan bahwa dalam daftar target untuk aturan, ini adalah target 1. Perintah ini juga menentukan contoh ARN untuk pipeline. Pipeline dimulai ketika sesuatu berubah di repositori.

```
aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-
AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

1. Jalankan get-pipeline perintah untuk menyalin struktur pipa ke file JSON. Misalnya, untuk pipeline bernama`MyFirstPipeline`, jalankan perintah berikut:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Perintah ini tidak mengembalikan apa pun, tetapi file yang Anda buat akan muncul di direktori tempat Anda menjalankan perintah.

2. Buka file JSON di editor teks biasa dan edit tahap sumber dengan mengubah `PollForSourceChanges` parameter untuk bucket bernama `amzn-s3-demo-source-bucketfalse`, seperti yang ditunjukkan dalam contoh ini.

   Mengapa saya membuat perubahan ini? Menyetel parameter ini untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Jika Anda bekerja dengan struktur pipa yang diambil menggunakan get-pipeline perintah, Anda harus menghapus `metadata` baris dari file JSON. Jika tidak, update-pipeline perintah tidak dapat menggunakannya. Hapus `"metadata": { }` garis dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang.

   Misalnya, hapus baris berikut dari struktur:

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

Simpan file tersebut.

4. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, dengan menentukan file JSON pipeline:

> **Important**
>
> Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

Perintah ini mengembalikan seluruh struktur pipa yang diedit.

> **Note**
>
> update-pipelinePerintah menghentikan pipa. Jika revisi sedang dijalankan melalui pipeline saat Anda menjalankan update-pipeline perintah, proses itu dihentikan. Anda harus memulai pipeline secara manual untuk menjalankan revisi itu melalui pipeline yang diperbarui. Gunakan start-pipeline-execution perintah untuk memulai pipeline Anda secara manual.

Migrasikan pipeline polling dengan sumber S3 yang diaktifkan untuk acara (template)CloudFormation

Prosedur ini untuk pipeline di mana bucket sumber mengaktifkan peristiwa.

Gunakan langkah-langkah ini untuk mengedit pipeline Anda dengan sumber Amazon S3 dari polling hingga deteksi perubahan berbasis peristiwa.

Untuk membuat pipeline berbasis peristiwa dengan Amazon S3, Anda mengedit `PollForSourceChanges` parameter pipeline, lalu menambahkan sumber daya berikut ke templat:

- EventBridge aturan dan peran IAM untuk memungkinkan acara ini memulai pipeline Anda.

Jika Anda menggunakan CloudFormation untuk membuat dan mengelola pipeline Anda, template Anda menyertakan konten seperti berikut ini.

> **Note**
>
> `Configuration`Properti dalam tahap sumber disebut`PollForSourceChanges`. Jika template Anda tidak menyertakan properti itu, maka `PollForSourceChanges` disetel ke secara `true` default.

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
                            }
                        ]
                    },
```

```
...
```

Untuk membuat EventBridge aturan dengan Amazon S3 sebagai sumber peristiwa dan CodePipeline sebagai target dan menerapkan kebijakan izin

1. Di template, di bawah`Resources`, gunakan `AWS::IAM::Role` CloudFormation sumber daya untuk mengonfigurasi peran IAM yang memungkinkan acara Anda memulai pipeline. Entri ini membuat peran yang menggunakan dua kebijakan:

   - Kebijakan pertama memungkinkan peran diasumsikan.
   - Kebijakan kedua memberikan izin untuk memulai pipeline.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::IAM::Role` sumber daya memungkinkan CloudFormation untuk membuat izin untuk EventBridge. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. Gunakan `AWS::Events::Rule` CloudFormation sumber daya untuk menambahkan EventBridge aturan. Pola peristiwa ini membuat peristiwa yang memantau pembuatan atau penghapusan objek di bucket sumber Amazon S3 Anda. Selain itu, sertakan target pipa Anda. Ketika sebuah objek dibuat, aturan ini dipanggil `StartPipelineExecution` pada pipeline target Anda.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::Events::Rule` sumber daya memungkinkan CloudFormation untuk membuat acara. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. Simpan template Anda yang diperbarui ke komputer lokal Anda, dan buka CloudFormation konsol.
4. Pilih tumpukan Anda, lalu pilih Create Change Set for Current Stack.
5. Unggah template Anda yang diperbarui, lalu lihat perubahan yang tercantum di dalamnya CloudFormation. Ini adalah perubahan yang akan dilakukan pada tumpukan. Anda harus melihat sumber daya baru Anda dalam daftar.
6. Pilih Eksekusi.

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan

mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

- Dalam template, ubah `PollForSourceChanges` ke`false`. Jika Anda tidak menyertakan `PollForSourceChanges` dalam definisi pipeline Anda, tambahkan dan setel ke`false`.

  Mengapa saya membuat perubahan ini? Mengubah `PollForSourceChanges` untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

  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

Saat Anda menggunakan CloudFormation untuk membuat sumber daya ini, pipeline Anda dipicu saat file di repositori dibuat atau diperbarui.

> **Note**
>
> Jangan berhenti di sini. Meskipun pipeline Anda dibuat, Anda harus membuat CloudFormation template kedua untuk pipeline Amazon S3 Anda. Jika Anda tidak membuat template kedua, pipeline Anda tidak memiliki fungsi deteksi perubahan.

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"
        }
    ]
}
```

Migrasi jaringan pemungutan suara dengan sumber dan jejak S3 CloudTrail

Untuk pipeline dengan sumber Amazon S3, modifikasi pipeline sehingga deteksi perubahan otomatis. EventBridge Pilih dari metode berikut untuk menerapkan migrasi:

- Konsol: Migrasikan saluran pemungutan suara (atau sumber Amazon CodeCommit S3) (konsol)
- CLI: Migrasi jaringan pemungutan suara dengan sumber dan jejak CloudTrail S3 (CLI)
- CloudFormation: Migrasikan jalur pemungutan suara dengan sumber dan CloudTrail jejak S3 (templat)CloudFormation

Migrasi jaringan pemungutan suara dengan sumber dan jejak CloudTrail S3 (CLI)

Ikuti langkah-langkah ini untuk mengedit pipeline yang menggunakan polling (pemeriksaan berkala) untuk menggunakan acara sebagai EventBridge gantinya. Jika Anda ingin membuat pipeline, lihatBuat pipeline, tahapan, dan tindakan.

Untuk membuat pipeline berbasis peristiwa dengan Amazon S3, Anda mengedit `PollForSourceChanges` parameter pipeline, lalu membuat sumber daya berikut:

- AWS CloudTrail kebijakan trail, bucket, dan bucket yang dapat digunakan Amazon S3 untuk mencatat peristiwa.

- EventBridge acara
- Peran IAM untuk memungkinkan EventBridge acara memulai pipeline Anda

Untuk membuat AWS CloudTrail jejak dan mengaktifkan logging

Untuk menggunakan AWS CLI untuk membuat jejak, panggil create-trail perintah, dengan menentukan:

- Nama jejak.
- Bucket tempat Anda telah menerapkan kebijakan bucket AWS CloudTrail.

Untuk informasi selengkapnya, lihat Membuat jejak dengan antarmuka baris AWS perintah.

1. Panggil create-trail perintah dan sertakan `--s3-bucket-name` parameter `--name` dan.

   Mengapa saya membuat perubahan ini? Ini menciptakan CloudTrail jejak yang diperlukan untuk bucket sumber S3 Anda.

   Perintah berikut menggunakan `--name` dan `--s3-bucket-name` untuk membuat jejak bernama `my-trail` dan ember bernama`amzn-s3-demo-source-bucket`.

   ```
   aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-source-bucket
   ```

2. Panggil start-logging perintah dan sertakan `--name` parameternya.

   Mengapa saya membuat perubahan ini? Perintah ini memulai CloudTrail pencatatan untuk bucket sumber Anda dan mengirimkan acara ke EventBridge.

   Contoh:

   Perintah berikut digunakan `--name` untuk memulai logging pada jejak bernama`my-trail`.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Panggil put-event-selectors perintah dan sertakan `--event-selectors` parameter `--trail-name` dan. Gunakan penyeleksi peristiwa untuk menentukan bahwa jejak Anda ingin mencatat peristiwa data untuk bucket sumber Anda dan mengirim peristiwa ke EventBridge aturan.

   Mengapa saya membuat perubahan ini? Perintah ini menyaring peristiwa.

Contoh:

Perintah berikut menggunakan `--trail-name` dan `--event-selectors` menentukan peristiwa data untuk bucket sumber dan awalan bernama`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"] }] }]'
```

Untuk membuat EventBridge aturan dengan Amazon S3 sebagai sumber peristiwa dan CodePipeline sebagai target dan menerapkan kebijakan izin

1. Berikan izin EventBridge untuk digunakan CodePipeline untuk menjalankan aturan. Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis sumber daya untuk Amazon. EventBridge

   a. Gunakan contoh berikut untuk membuat kebijakan kepercayaan agar memungkinkan EventBridge untuk mengambil peran layanan. Nama itu`trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Gunakan perintah berikut untuk membuat `Role-for-MyRule` peran dan melampirkan kebijakan kepercayaan.

Mengapa saya membuat perubahan ini? Menambahkan kebijakan kepercayaan ini ke peran akan membuat izin untuk EventBridge.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Buat kebijakan izin JSON, seperti yang ditunjukkan di sini untuk pipeline bernama. `MyFirstPipeline` Beri nama kebijakan `permissionspolicyforEB.json` izin.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Gunakan perintah berikut untuk melampirkan kebijakan `CodePipeline-Permissions-Policy-for-EB` izin baru ke `Role-for-MyRule` peran yang Anda buat.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Panggil put-rule perintah dan sertakan`--name`,`--event-pattern`, dan `--role-arn` parameter.

Perintah contoh berikut membuat aturan bernama`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. Untuk menambahkan CodePipeline sebagai target, panggil put-targets perintah dan sertakan `--targets` parameter `--rule` dan.

   Perintah berikut menentukan bahwa untuk aturan bernama`MyS3SourceRule`, target `Id` terdiri dari nomor satu, menunjukkan bahwa dalam daftar target untuk aturan, ini adalah target 1. Perintah ini juga menentukan contoh ARN untuk pipeline. Pipeline dimulai ketika sesuatu berubah di repositori.

   ```
   aws events put-targets --rule MyS3SourceRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

4. (Opsional) Untuk mengonfigurasi transformator input dengan penggantian sumber untuk ID gambar tertentu, gunakan JSON berikut dalam perintah CLI Anda. Contoh berikut mengonfigurasi penggantian di mana:

   - `Source`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `actionName`
   - `S3_OBJECT_VERSION_ID`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `revisionType`
   - The`revisionValue`, < *`revisionValue`* > dalam contoh ini, berasal dari variabel peristiwa sumber.

   ```
   {
       "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>"
                    }
                }
            }
        }
    ]
}
```

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

1. Jalankan get-pipeline perintah untuk menyalin struktur pipa ke file JSON. Misalnya, untuk pipeline bernama`MyFirstPipeline`, jalankan perintah berikut:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Perintah ini tidak mengembalikan apa pun, tetapi file yang Anda buat akan muncul di direktori tempat Anda menjalankan perintah.

2. Buka file JSON di editor teks biasa dan edit tahap sumber dengan mengubah `PollForSourceChanges` parameter untuk bucket bernama `amzn-s3-demo-source-bucketfalse`, seperti yang ditunjukkan dalam contoh ini.

   Mengapa saya membuat perubahan ini? Menyetel parameter ini untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
   ```

```
    "PollForSourceChanges": "false",
    "S3ObjectKey": "index.zip"
},
```

3. Jika Anda bekerja dengan struktur pipa yang diambil menggunakan get-pipeline perintah, Anda harus menghapus `metadata` baris dari file JSON. Jika tidak, update-pipeline perintah tidak dapat menggunakannya. Hapus `"metadata": { }` garis dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang.

   Misalnya, hapus baris berikut dari struktur:

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

   Simpan file tersebut.

4. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, dengan menentukan file JSON pipeline:

   > **Important**
   >
   > Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

   Perintah ini mengembalikan seluruh struktur pipa yang diedit.

   > **Note**
   >
   > update-pipelinePerintah menghentikan pipa. Jika revisi sedang dijalankan melalui pipeline saat Anda menjalankan update-pipeline perintah, proses itu dihentikan. Anda harus memulai pipeline secara manual untuk menjalankan revisi itu melalui pipeline yang diperbarui. Gunakan start-pipeline-execution perintah untuk memulai pipeline Anda secara manual.

Migrasikan jalur pemungutan suara dengan sumber dan CloudTrail jejak S3 (templat)CloudFormation

Gunakan langkah-langkah ini untuk mengedit pipeline Anda dengan sumber Amazon S3 dari polling hingga deteksi perubahan berbasis peristiwa.

Untuk membuat pipeline berbasis peristiwa dengan Amazon S3, Anda mengedit `PollForSourceChanges` parameter pipeline, lalu menambahkan sumber daya berikut ke templat:

- EventBridge mengharuskan semua peristiwa Amazon S3 harus dicatat. Anda harus membuat kebijakan AWS CloudTrail trail, bucket, dan bucket yang dapat digunakan Amazon S3 untuk mencatat peristiwa yang terjadi. Untuk informasi selengkapnya, lihat Mencatat peristiwa data untuk jejak dan peristiwa pengelolaan Logging untuk jejak.
- EventBridge aturan dan peran IAM untuk memungkinkan acara ini memulai pipeline kami.

Jika Anda menggunakan CloudFormation untuk membuat dan mengelola pipeline Anda, template Anda menyertakan konten seperti berikut ini.

> **Note**
>
> `ConfigurationProperti` dalam tahap sumber disebut`PollForSourceChanges`. Jika template Anda tidak menyertakan properti itu, maka `PollForSourceChanges` disetel ke secara `true` default.

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
                        }
                    ]
                },


...
```

Untuk membuat EventBridge aturan dengan Amazon S3 sebagai sumber peristiwa dan CodePipeline sebagai target dan menerapkan kebijakan izin

1. Di template, di bawah`Resources`, gunakan `AWS::IAM::Role` CloudFormation sumber daya untuk mengonfigurasi peran IAM yang memungkinkan acara Anda memulai pipeline. Entri ini membuat peran yang menggunakan dua kebijakan:

   - Kebijakan pertama memungkinkan peran diasumsikan.
   - Kebijakan kedua memberikan izin untuk memulai pipeline.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::IAM::Role` sumber daya memungkinkan CloudFormation untuk membuat izin untuk EventBridge. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. Gunakan `AWS::Events::Rule` CloudFormation sumber daya untuk menambahkan EventBridge aturan. Pola acara ini membuat acara yang memantau`CopyObject`, `PutObject` dan `CompleteMultipartUpload` di bucket sumber Amazon S3 Anda. Selain itu, sertakan target pipa Anda. Ketika`CopyObject`,`PutObject`, atau `CompleteMultipartUpload` terjadi, aturan ini muncul `StartPipelineExecution` di pipeline target Anda.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::Events::Rule` sumber daya memungkinkan CloudFormation untuk membuat acara. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. Tambahkan cuplikan ini ke template pertama Anda untuk memungkinkan fungsionalitas cross-stack:

   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. (Opsional) Untuk mengonfigurasi transformator input dengan penggantian sumber untuk ID gambar tertentu, gunakan cuplikan YAMAL berikut. Contoh berikut mengonfigurasi penggantian di mana:

   - `Source`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `actionName`

- `S3_OBJECT_VERSION_ID`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `revisionType`
- The`revisionValue`, **<** *`revisionValue`* **>** dalam contoh ini, berasal dari variabel peristiwa sumber.

```
---
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. Simpan template Anda yang diperbarui ke komputer lokal Anda, dan buka CloudFormation konsol.
6. Pilih tumpukan Anda, lalu pilih Create Change Set for Current Stack.
7. Unggah template Anda yang diperbarui, lalu lihat perubahan yang tercantum di dalamnya CloudFormation. Ini adalah perubahan yang akan dilakukan pada tumpukan. Anda harus melihat sumber daya baru Anda dalam daftar.
8. Pilih Eksekusi.

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

- Dalam template, ubah `PollForSourceChanges` ke`false`. Jika Anda tidak menyertakan `PollForSourceChanges` dalam definisi pipeline Anda, tambahkan dan setel ke`false`.

  Mengapa saya membuat perubahan ini? Mengubah `PollForSourceChanges` untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

  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
  }
```

Untuk membuat template kedua untuk sumber daya saluran Amazon S3 Anda CloudTrail

- Dalam templat terpisah, di bawah`Resources`, gunakan`AWS::S3::Bucket`,`AWS::S3::BucketPolicy`, dan `AWS::CloudTrail::Trail` CloudFormation sumber daya untuk memberikan definisi dan jejak bucket sederhana CloudTrail.

  Mengapa saya membuat perubahan ini? Mengingat batas saat ini lima jalur per akun, CloudTrail jejak harus dibuat dan dikelola secara terpisah. (Lihat Batas di AWS CloudTrail.) Namun, Anda dapat menyertakan banyak bucket Amazon S3 pada satu jalur, sehingga Anda dapat membuat jejak sekali dan kemudian menambahkan bucket Amazon S3 untuk saluran pipa lain seperlunya. Tempelkan berikut ini ke dalam file template sampel kedua Anda.

  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

Saat Anda menggunakan CloudFormation untuk membuat sumber daya ini, pipeline Anda dipicu saat file di repositori dibuat atau diperbarui.

> **Note**
>
> Jangan berhenti di sini. Meskipun pipeline Anda dibuat, Anda harus membuat CloudFormation template kedua untuk pipeline Amazon S3 Anda. Jika Anda tidak membuat template kedua, pipeline Anda tidak memiliki fungsi deteksi perubahan.

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"
            }
        }
    }
}


...
```

Migrasikan pipeline polling untuk tindakan sumber GitHub (melalui OAuth aplikasi) ke koneksi

Anda dapat memigrasikan tindakan sumber GitHub (melalui OAuth aplikasi) untuk menggunakan koneksi untuk repositori eksternal Anda. Ini adalah metode deteksi perubahan yang direkomendasikan untuk saluran pipa dengan tindakan sumber GitHub (melalui OAuth aplikasi).

Untuk pipeline dengan tindakan sumber GitHub (melalui OAuth aplikasi), kami sarankan untuk memodifikasi pipeline untuk menggunakan tindakan GitHub (melalui GitHub Aplikasi) sehingga deteksi perubahan otomatis. AWS CodeConnections Untuk informasi selengkapnya tentang bekerja dengan koneksi, lihatGitHub koneksi.

Buat koneksi ke GitHub (konsol)

Anda dapat menggunakan konsol untuk membuat koneksi ke GitHub.

Langkah 1: Ganti tindakan Anda GitHub (melalui OAuth aplikasi)

Gunakan halaman edit pipeline untuk mengganti tindakan GitHub (melalui OAuth aplikasi) Anda dengan tindakan GitHub (melalui GitHub Aplikasi).

Untuk mengganti tindakan Anda GitHub (melalui OAuth aplikasi)

1. Masuk ke CodePipeline konsol.
2. Pilih pipeline Anda, dan pilih Edit. Pilih Edit tahap pada tahap sumber Anda. Sebuah pesan menampilkan yang merekomendasikan Anda memperbarui tindakan Anda.

3. Di penyedia Tindakan, pilih GitHub (melalui GitHub Aplikasi).
4. Lakukan salah satu tindakan berikut:

   - Di bawah Koneksi, jika Anda belum membuat sambungan ke penyedia Anda, pilih Connect to GitHub. Lanjutkan ke Langkah 2: Buat koneksi ke GitHub.
   - Di bawah Koneksi, jika Anda telah membuat koneksi ke penyedia Anda, pilih koneksi. Lanjutkan ke Langkah 3: Simpan Tindakan Sumber untuk Koneksi Anda.

Langkah 2: Buat koneksi ke GitHub

Setelah Anda memilih untuk membuat koneksi, GitHub halaman Connect to ditampilkan.

Untuk membuat koneksi ke GitHub

1. Di bawah pengaturan GitHub koneksi, nama koneksi Anda ditampilkan di Nama koneksi.

   Di bawah GitHub Aplikasi, pilih penginstalan aplikasi atau pilih Instal aplikasi baru untuk membuatnya.

   > **Note**
   >
   > Anda menginstal satu aplikasi untuk semua koneksi Anda ke penyedia tertentu. Jika Anda telah menginstal GitHub aplikasi, pilih dan lewati langkah ini.

2. Jika halaman otorisasi untuk GitHub ditampilkan, masuk dengan kredensil Anda dan kemudian pilih untuk melanjutkan.
3. Di halaman penginstalan aplikasi, pesan menunjukkan bahwa AWS CodeStar aplikasi mencoba terhubung ke GitHub akun Anda.

   > **Note**
   >
   > Anda hanya menginstal aplikasi sekali untuk setiap GitHub akun. Jika sebelumnya Anda menginstal aplikasi, Anda dapat memilih Konfigurasi untuk melanjutkan ke halaman modifikasi untuk instalasi aplikasi Anda, atau Anda dapat menggunakan tombol kembali untuk kembali ke konsol.

4. Pada AWS CodeStar halaman Instal, pilih Instal.

5. Pada GitHub halaman Connect to, ID koneksi untuk instalasi baru Anda ditampilkan. Pilih Hubungkan.

Langkah 3: Simpan tindakan GitHub sumber Anda

Selesaikan pembaruan Anda di halaman Edit tindakan untuk menyimpan tindakan sumber baru Anda.

Untuk menyimpan tindakan GitHub sumber Anda

1. Di Repositori, masukkan nama repositori pihak ketiga Anda. Di Branch, masukkan cabang tempat Anda ingin pipeline mendeteksi perubahan sumber.

   > **Note**
   >
   > Di Repository, ketik `owner-name/repository-name` seperti yang ditunjukkan dalam contoh ini:
   >
   > ```
   > my-account/my-repository
   > ```

2. Dalam format artefak Output, pilih format untuk artefak Anda.

   - Untuk menyimpan artefak keluaran dari GitHub tindakan menggunakan metode default, pilih CodePipelinedefault. Tindakan mengakses file dari GitHub repositori dan menyimpan artefak dalam file ZIP di toko artefak pipa.
   - Untuk menyimpan file JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung, pilih klon Penuh. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.

     Jika Anda memilih opsi ini, Anda perlu memperbarui izin untuk peran layanan CodeBuild proyek Anda seperti yang ditunjukkan padaTambahkan CodeBuild GitClone izin untuk koneksi ke Bitbucket,, GitHub Enterprise Server GitHub, atau .com GitLab. Untuk tutorial yang menunjukkan cara menggunakan opsi klon Penuh, lihatTutorial: Gunakan klon lengkap dengan sumber GitHub pipa.

3. Di artefak Output, Anda dapat mempertahankan nama artefak keluaran untuk tindakan ini, seperti. `SourceArtifact` Pilih Selesai untuk menutup halaman tindakan Edit.

4. Pilih Selesai untuk menutup halaman pengeditan panggung. Pilih Simpan untuk menutup halaman pengeditan pipeline.

Buat koneksi ke GitHub (CLI)

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk membuat koneksi ke GitHub.

Untuk melakukannya, gunakan perintah create-connection.

> **Important**
>
> Koneksi yang dibuat melalui AWS CLI atau AWS CloudFormation dalam `PENDING` status secara default. Setelah Anda membuat koneksi dengan CLI atau CloudFormation, gunakan konsol untuk mengedit koneksi untuk membuat statusnya. `AVAILABLE`

Untuk membuat koneksi ke GitHub

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows). Gunakan AWS CLI untuk menjalankan create-connection perintah, menentukan `--provider-type` dan `--connection-name` untuk koneksi Anda. Dalam contoh ini, nama penyedia pihak ketiga adalah `GitHub` dan nama koneksi yang ditentukan adalah `MyConnection`.

   ```
   aws codeconnections create-connection --provider-type GitHub --connection-name
    MyConnection
   ```

   Jika berhasil, perintah ini mengembalikan informasi ARN koneksi seperti berikut ini.

   ```
   {
       "ConnectionArn": "arn:aws:codeconnections:us-west-2:account_id:connection/
   aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
   }
   ```

2. Gunakan konsol untuk menyelesaikan koneksi.

Migrasikan jalur polling untuk tindakan sumber GitHub (melalui OAuth aplikasi) ke webhooks

Anda dapat memigrasikan pipeline untuk menggunakan webhook guna mendeteksi perubahan di repositori sumber. GitHub Migrasi ke webhooks ini hanya untuk tindakan GitHub (melalui OAuth aplikasi).

- Konsol: Migrasikan pipeline polling ke webhooks ((melalui OAuth aplikasi) tindakan sumber) GitHub (konsol)
- CLI: Migrasikan jalur pemungutan suara ke tindakan sumber webhook (GitHub (melalui OAuth aplikasi)) (CLI)
- CloudFormation: Perbarui saluran pipa untuk acara push (GitHub (melalui OAuth aplikasi) tindakan sumber) (CloudFormation templat)

> **Important**
>
> Saat membuat CodePipeline webhook, jangan gunakan kredensialnya sendiri atau gunakan kembali token rahasia yang sama di beberapa webhook. Untuk keamanan optimal, buat token rahasia unik untuk setiap webhook yang Anda buat. Token rahasia adalah string arbitrer yang Anda berikan, yang GitHub digunakan untuk menghitung dan menandatangani muatan webhook yang dikirim ke CodePipeline, untuk melindungi integritas dan keaslian muatan webhook. Menggunakan kredensil Anda sendiri atau menggunakan kembali token yang sama di beberapa webhook dapat menyebabkan kerentanan keamanan.

Migrasikan pipeline polling ke webhooks ((melalui OAuth aplikasi) tindakan sumber) GitHub (konsol)

Untuk tindakan sumber GitHub (melalui OAuth aplikasi), Anda dapat menggunakan CodePipeline konsol untuk memperbarui pipeline agar menggunakan webhook guna mendeteksi perubahan di repositori GitHub sumber.

Ikuti langkah-langkah ini untuk mengedit pipeline yang menggunakan polling (pemeriksaan berkala) untuk digunakan EventBridge sebagai gantinya. Jika Anda ingin membuat pipeline, lihatBuat pipeline, tahapan, dan tindakan.

Saat Anda menggunakan konsol, `PollForSourceChanges` parameter untuk pipelined Anda diubah untuk Anda. GitHub Webhook dibuat dan terdaftar untuk Anda.

Untuk mengedit tahap sumber pipa

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda edit. Ini membuka tampilan rinci dari pipa, termasuk keadaan masing-masing tindakan di setiap tahap pipa.
3. Pada halaman detail pipeline, pilih Edit.
4. Di tahap Edit, pilih ikon edit pada aksi sumber.
5. Perluas Ubah opsi deteksi dan pilih Gunakan CloudWatch Acara Amazon untuk memulai pipeline saya secara otomatis saat terjadi perubahan (disarankan).

   Pesan ditampilkan untuk memberi saran yang CodePipeline membuat webhook GitHub untuk mendeteksi perubahan sumber: AWS CodePipeline akan membuat webhook untuk Anda. Anda dapat memilih keluar dalam opsi di bawah ini. Pilih Perbarui. Selain webhook, CodePipeline buat yang berikut:

   - Sebuah rahasia, dibuat secara acak dan digunakan untuk mengotorisasi koneksi ke. GitHub
   - URL webhook, dibuat menggunakan titik akhir publik untuk Wilayah.

   CodePipeline mendaftarkan webhook dengan. GitHub Ini berlangganan URL untuk menerima acara repositori.
6. Setelah selesai mengedit pipeline, pilih Simpan perubahan pipeline untuk kembali ke halaman ringkasan.

   Pesan menampilkan nama webhook yang akan dibuat untuk pipeline Anda. Jangan pilih Save and continue (Simpan dan lanjutkan).
7. Untuk menguji tindakan Anda, lepaskan perubahan dengan menggunakan AWS CLI to commit perubahan ke sumber yang ditentukan dalam tahap sumber pipeline.

Migrasikan jalur pemungutan suara ke tindakan sumber webhook (GitHub (melalui OAuth aplikasi)) (CLI)

Ikuti langkah-langkah ini untuk mengedit pipeline yang menggunakan pemeriksaan berkala untuk menggunakan webhook sebagai gantinya. Jika Anda ingin membuat pipeline, lihatBuat pipeline, tahapan, dan tindakan.

Untuk membangun pipeline yang digerakkan oleh peristiwa, Anda mengedit `PollForSourceChanges` parameter pipeline Anda dan kemudian membuat sumber daya berikut secara manual:

- GitHub webhook dan parameter otorisasi

Untuk membuat dan mendaftarkan webhook Anda

> **Note**
>
> Saat Anda menggunakan CLI atau CloudFormation untuk membuat pipeline dan menambahkan webhook, Anda harus menonaktifkan pemeriksaan berkala. Untuk menonaktifkan pemeriksaan berkala, Anda harus secara eksplisit menambahkan `PollForSourceChanges` parameter dan mengaturnya ke false, seperti yang dijelaskan dalam prosedur akhir di bawah ini. Jika tidak, default untuk CLI atau CloudFormation pipeline adalah `PollForSourceChanges` defaultnya true dan tidak ditampilkan dalam output struktur pipa. Untuk informasi selengkapnya tentang PollForSourceChanges default, lihat. Pengaturan yang valid untuk `PollForSourceChanges` parameter

1. Dalam editor teks, buat dan simpan file JSON untuk webhook yang ingin Anda buat. Gunakan file contoh ini untuk webhook bernama`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. Panggil put-webhook perintah dan sertakan `--region` parameter `--cli-input` dan.

   Contoh perintah berikut membuat webhook dengan file `webhook_json` JSON.

```
aws codepipeline put-webhook --cli-input-json file://webhook_json.json --region
 "eu-central-1"
```

3. Dalam output yang ditunjukkan dalam contoh ini, URL dan ARN dikembalikan untuk webhook bernama. `my-webhook`

```
{
    "webhook": {
        "url": "https://webhooks.domain.com/
trigger111111111EXAMPLE11111111111111111",
        "definition": {
            "authenticationConfiguration": {
                "SecretToken": "secret"
            },
            "name": "my-webhook",
            "authentication": "GITHUB_HMAC",
            "targetPipeline": "pipeline_name",
            "targetAction": "Source",
            "filters": [
                {
                    "jsonPath": "$.ref",
                    "matchEquals": "refs/heads/{Branch}"
                }
            ]
        },
        "arn": "arn:aws:codepipeline:eu-central-1:ACCOUNT_ID:webhook:my-webhook"
    },
    "tags": [{
      "key": "Project",
      "value": "ProjectA"
    }]
}
```

Contoh ini menambahkan penandaan ke webhook dengan menyertakan kunci `Project` tag dan `ProjectA` nilai pada webhook. Untuk informasi selengkapnya tentang menandai sumber daya CodePipeline, lihat[Penandaan pada sumber daya](#).

4. Panggil register-webhook-with-third-party perintah dan sertakan `--webhook-name` parameternya.

   Contoh perintah berikut mendaftarkan webhook bernama. `my-webhook`

   ```
   aws codepipeline register-webhook-with-third-party --webhook-name my-webhook
   ```

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di [Pengaturan yang valid untuk `PollForSourceChanges` parameter](#).

1. Jalankan get-pipeline perintah untuk menyalin struktur pipa ke file JSON. Misalnya, untuk pipeline bernama`MyFirstPipeline`, Anda akan mengetik perintah berikut:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Perintah ini tidak mengembalikan apa pun, tetapi file yang Anda buat akan muncul di direktori tempat Anda menjalankan perintah.

2. Buka file JSON di editor teks biasa dan edit tahap sumber dengan mengubah atau menambahkan parameter. `PollForSourceChanges` Dalam contoh ini, untuk repositori bernama`UserGitHubRepo`, parameter diatur ke. `false`

   Mengapa saya membuat perubahan ini? Mengubah parameter ini mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

```
"configuration": {
    "Owner": "name",
    "Repo": "UserGitHubRepo",
    "PollForSourceChanges": "false",
    "Branch": "main",
    "OAuthToken": "****"
},
```

3. Jika Anda bekerja dengan struktur pipa yang diambil menggunakan get-pipeline perintah, Anda harus mengedit struktur dalam file JSON dengan menghapus `metadata` baris dari file. Jika tidak, update-pipeline perintah tidak dapat menggunakannya. Hapus `"metadata"` bagian dari struktur pipa di file JSON, termasuk: `{ }` dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang.

   Misalnya, hapus baris berikut dari struktur:

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

   Simpan file tersebut.

4. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, tentukan file JSON pipeline, mirip dengan berikut ini:

   > **Important**
   >
   > Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

   Perintah ini mengembalikan seluruh struktur pipa yang diedit.

> **Note**
>
> update-pipelinePerintah menghentikan pipa. Jika revisi sedang dijalankan melalui pipeline saat Anda menjalankan update-pipeline perintah, proses itu dihentikan. Anda harus memulai pipeline secara manual untuk menjalankan revisi itu melalui pipeline yang diperbarui. Gunakan start-pipeline-execution perintah untuk memulai pipeline Anda secara manual.

Perbarui saluran pipa untuk acara push (GitHub (melalui OAuth aplikasi) tindakan sumber) (CloudFormation templat)

Ikuti langkah-langkah berikut untuk memperbarui pipeline Anda (dengan GitHub sumber) dari pemeriksaan berkala (polling) hingga deteksi perubahan berbasis peristiwa menggunakan webhook.

Untuk membangun pipeline berbasis peristiwa AWS CodeCommit, Anda mengedit `PollForSourceChanges` parameter pipeline Anda dan kemudian menambahkan sumber daya GitHub webhook ke template Anda.

Jika Anda menggunakan CloudFormation untuk membuat dan mengelola pipeline Anda, template Anda memiliki konten seperti berikut ini.

> **Note**
>
> Perhatikan properti `PollForSourceChanges` konfigurasi di tahap sumber. Jika template Anda tidak menyertakan properti itu, maka `PollForSourceChanges` disetel ke secara `true` default.

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
                            }
                        ]
                    },


...
```

Untuk menambahkan parameter dan membuat webhook di template Anda

Kami sangat menyarankan Anda menggunakan AWS Secrets Manager untuk menyimpan kredensil Anda. Jika Anda menggunakan Secrets Manager, Anda harus sudah mengkonfigurasi dan menyimpan parameter rahasia Anda di Secrets Manager. Contoh ini menggunakan referensi dinamis ke Secrets Manager untuk GitHub kredensialnya untuk webhook Anda. Untuk informasi selengkapnya, lihat Menggunakan Referensi Dinamis untuk Menentukan Nilai Templat.

> **Important**
>
> Saat meneruskan parameter rahasia, jangan masukkan nilainya langsung ke templat. Nilai tersebut diberikan sebagai teks biasa dan karena itu dapat dibaca. Untuk alasan keamanan, jangan gunakan plaintext di CloudFormation template Anda untuk menyimpan kredensi Anda.

Saat Anda menggunakan CLI atau CloudFormation untuk membuat pipeline dan menambahkan webhook, Anda harus menonaktifkan pemeriksaan berkala.

> **Note**
>
> Untuk menonaktifkan pemeriksaan berkala, Anda harus secara eksplisit menambahkan `PollForSourceChanges` parameter dan mengaturnya ke false, seperti yang dijelaskan dalam prosedur akhir di bawah ini. Jika tidak, default untuk CLI atau CloudFormation pipeline adalah `PollForSourceChanges` defaultnya true dan tidak ditampilkan dalam output struktur pipa. Untuk informasi selengkapnya tentang PollForSourceChanges default, lihat. Pengaturan yang valid untuk `PollForSourceChanges` parameter

1. Dalam template, di bawah`Resources`, tambahkan parameter Anda:

   YAML

   ```
   Parameters:
           GitHubOwner:
             Type: String

   ...
   ```

   JSON

   ```
   {
    "Parameters": {
     "BranchName": {
      "Description": "GitHub branch name",
      "Type": "String",
      "Default": "main"
     },
     "GitHubOwner": {
   ```

```
    "Type": "String"
  },

...
```

2. Gunakan `AWS::CodePipeline::Webhook` CloudFormation sumber daya untuk menambahkan webhook.

> **Note**
>
> Yang `TargetAction` Anda tentukan harus cocok dengan `Name` properti tindakan sumber yang ditentukan dalam pipeline.

Jika `RegisterWithThirdParty` disetel ke`true`, pastikan pengguna yang terkait dengan `OAuthToken` dapat mengatur cakupan yang diperlukan. GitHub Token dan webhook memerlukan GitHub cakupan berikut:

- `repo`- digunakan untuk kontrol penuh untuk membaca dan menarik artefak dari repositori publik dan swasta ke dalam pipa.
- `admin:repo_hook`- digunakan untuk kontrol penuh kait repositori.

Jika tidak, GitHub mengembalikan 404. Untuk informasi lebih lanjut tentang 404 yang dikembalikan, lihathttps://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. Simpan template yang diperbarui ke komputer lokal Anda, lalu buka CloudFormation konsol.
4. Pilih tumpukan Anda, lalu pilih Create Change Set for Current Stack.

5. Unggah template, lalu lihat perubahan yang tercantum di dalamnya CloudFormation. Ini adalah perubahan yang harus dilakukan pada tumpukan. Anda harus melihat sumber daya baru Anda dalam daftar.
6. Pilih Eksekusi.

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

- Dalam template, ubah `PollForSourceChanges` ke`false`. Jika Anda tidak menyertakan `PollForSourceChanges` dalam definisi pipeline Anda, tambahkan dan setel ke false.

  Mengapa saya membuat perubahan ini? Mengubah parameter ini untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

  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

Saat Anda membuat sumber daya ini CloudFormation, webhook yang ditentukan dibuat di GitHub repositori yang ditentukan. Pipeline Anda dipicu saat komit.

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

...
```

Buat peran CodePipeline layanan

Saat membuat pipeline, Anda membuat peran layanan atau menggunakan peran layanan yang ada.

Anda dapat menggunakan CodePipeline konsol atau AWS CLI untuk membuat peran CodePipeline layanan. Peran layanan diperlukan untuk membuat pipeline, dan pipeline selalu dikaitkan dengan peran layanan tersebut.

Sebelum Anda membuat pipeline dengan AWS CLI, Anda harus membuat peran CodePipeline layanan untuk pipeline Anda. Untuk contoh CloudFormation templat dengan peran layanan dan kebijakan yang ditentukan, lihat tutorial diTutorial: Buat pipeline yang menggunakan variabel dari AWS CloudFormation tindakan penerapan.

Peran layanan bukan peran AWS terkelola tetapi awalnya dibuat untuk pembuatan pipeline, dan kemudian saat izin baru ditambahkan ke kebijakan peran layanan, Anda mungkin perlu memperbarui peran layanan untuk pipeline Anda. Setelah pipeline dibuat dengan peran layanan, Anda tidak dapat menerapkan peran layanan yang berbeda ke pipeline tersebut. Lampirkan kebijakan yang direkomendasikan ke peran layanan.

Untuk informasi selengkapnya tentang peran layanan, lihatKelola peran CodePipeline layanan.

Buat peran CodePipeline layanan (konsol)

Saat menggunakan konsol untuk membuat pipeline, Anda membuat peran CodePipeline layanan dengan panduan pembuatan pipeline.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Pilih Buat pipeline dan selesaikan Langkah 1: Pilih halaman pengaturan pipeline di panduan pembuatan pipeline.

   > **Note**
   >
   > Setelah Anda membuat pipeline, Anda tidak dapat mengubah namanya. Untuk informasi tentang batasan lainnya, lihatKuota di AWS CodePipeline.

2. Dalam peran Layanan, pilih Peran layanan baru CodePipeline untuk memungkinkan membuat peran layanan baru di IAM.
3. Selesaikan pembuatan pipa. Peran layanan pipeline Anda tersedia untuk dilihat dalam daftar peran IAM, dan Anda dapat melihat peran layanan ARN yang terkait dengan pipeline dengan menjalankan `get-pipeline` perintah dengan CLI. AWS

Buat peran CodePipeline layanan (CLI)

Sebelum membuat pipeline dengan AWS CLI atau CloudFormation, Anda harus membuat peran CodePipeline layanan untuk pipeline Anda dan melampirkan kebijakan peran layanan dan kebijakan

kepercayaan. Untuk menggunakan CLI untuk membuat peran layanan Anda, gunakan langkah-langkah di bawah ini untuk terlebih dahulu membuat JSON kebijakan kepercayaan dan kebijakan peran JSON sebagai file terpisah di direktori tempat Anda akan menjalankan perintah CLI.

> **Note**
>
> Kami menyarankan Anda hanya mengizinkan pengguna administratif untuk membuat peran layanan apa pun. Seseorang yang memiliki izin untuk membuat peran dan melampirkan kebijakan apa pun dapat meningkatkan izinnya sendiri. Sebagai gantinya, buat kebijakan yang memungkinkan mereka hanya membuat peran yang mereka butuhkan atau minta administrator membuat peran layanan atas nama mereka.

1. Di jendela terminal, masukkan perintah berikut untuk membuat file bernama`TrustPolicy.json`, di mana Anda akan menempelkan kebijakan peran JSON. Contoh ini menggunakan VIM.

   ```
   vim TrustPolicy.json
   ```

2. Paste JSON berikut ke dalam file.

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": "codepipeline.amazonaws.com"
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```

   Untuk menyimpan dan keluar dari file, masukkan perintah VIM berikut:

```
:wq
```

3. Di jendela terminal, masukkan perintah berikut untuk membuat file bernama`RolePolicy.json`, di mana Anda akan menempelkan kebijakan peran JSON. Contoh ini menggunakan VIM.

```
vim RolePolicy.json
```

4. Tempelkan kebijakan JSON ke dalam file. Gunakan kebijakan peran layanan minimum seperti yang disediakan dalamCodePipeline kebijakan peran layanan. Selain itu, tambahkan izin yang sesuai ke peran layanan Anda berdasarkan tindakan yang Anda rencanakan untuk digunakan. Untuk daftar tindakan dan tautan ke izin peran layanan yang diperlukan setiap tindakan, lihatMenambahkan izin ke peran CodePipeline layanan.

   Pastikan untuk mengurangi izin sebanyak mungkin dengan meringkas ke tingkat sumber daya di lapangan. `Resource`

   Untuk menyimpan dan keluar dari file, masukkan perintah VIM berikut:

```
:wq
```

5. Masukkan perintah berikut untuk membuat peran dan lampirkan kebijakan peran kepercayaan. Format nama kebijakan biasanya sama dengan format nama peran. Contoh ini menggunakan nama peran `MyRole` dan kebijakan `TrustPolicy` yang dibuat sebagai file terpisah.

```
aws iam create-role --role-name MyRole --assume-role-policy-document file://
TrustPolicy.json
```

6. Masukkan perintah berikut untuk membuat kebijakan peran dan melampirkannya ke peran. Format nama kebijakan biasanya sama dengan format nama peran. Contoh ini menggunakan nama peran `MyRole` dan kebijakan `MyRole` yang dibuat sebagai file terpisah.

```
aws iam put-role-policy --role-name MyRole --policy-name RolePolicy --policy-
document file://RolePolicy.json
```

7. Untuk melihat nama peran dan kebijakan kepercayaan yang dibuat, masukkan perintah berikut untuk peran bernama`MyRole`:

```
aws iam get-role --role-name MyRole
```

8. Gunakan peran layanan ARN saat Anda membuat pipeline dengan CLI AWS atau. CloudFormation

Penandaan pada sumber daya

Tag adalah label atribut kustom yang Anda atau AWS tetapkan ke AWS sumber daya. Setiap AWS tag memiliki dua bagian:

- Kunci tag (misalnya, `CostCenter`, `Environment`, `Project`, atau `Secret`). Kunci tanda peka terhadap huruf besar dan kecil.
- Bidang opsional yang dikenal sebagai nilai tag (misalnya, `111122223333`, `Production`, atau nama tim). Mengabaikan nilai tag sama dengan menggunakan rangkaian kosong. Seperti kunci tanda, nilai tanda peka huruf besar dan kecil.

Bersama-sama ini dikenal sebagai pasangan nilai-kunci.

Tag membantu Anda mengidentifikasi dan mengatur AWS sumber daya Anda. Banyak penandaan Layanan AWS dukungan, sehingga Anda dapat menetapkan tag yang sama ke sumber daya dari layanan yang berbeda untuk menunjukkan bahwa sumber daya terkait. Misalnya, Anda dapat menetapkan tag yang sama ke pipeline yang ditetapkan ke bucket sumber Amazon S3.

Untuk tips menggunakan tanda, lihat postingan AWS Strategi Penandaan di blog AWS Jawaban.

Anda dapat menandai jenis sumber daya berikut di CodePipeline:

- Tandai pipa di CodePipeline
- Menandai tindakan kustom di CodePipeline

Anda dapat menggunakan AWS CLI, CodePipeline APIs, atau AWS SDKs untuk:

- Tambahkan tag ke pipeline, tindakan kustom, atau webhook saat Anda membuatnya.
- Menambahkan, mengelola, dan menghapus tag untuk pipeline, tindakan kustom, atau webhook.

Anda juga dapat menggunakan konsol untuk menambahkan, mengelola, dan menghapus tag untuk pipeline.

Selain mengidentifikasi, mengatur, dan melacak sumber daya Anda dengan tanda, Anda dapat menggunakan tanda dalam kebijakan IAM untuk membantu mengontrol siapa yang dapat melihat dan berinteraksi dengan sumber daya Anda. Untuk contoh kebijakan akses berbasis tag, lihat Menggunakan tag untuk mengontrol akses ke CodePipeline sumber daya.

Tandai pipa di CodePipeline

Tag adalah pasangan nilai kunci yang terkait dengan AWS sumber daya. Anda dapat menerapkan tag ke saluran pipa Anda di CodePipeline. Untuk informasi tentang penandaan CodePipeline sumber daya, kasus penggunaan, kunci tag dan batasan nilai, serta jenis sumber daya yang didukung, lihat. Penandaan pada sumber daya

Anda dapat menggunakan CLI untuk menentukan tag saat Anda membuat pipeline. Anda dapat menggunakan konsol atau CLI untuk menambah atau menghapus tag, dan memperbarui nilai tag dalam pipeline. Anda dapat menambahkan hingga 50 tag ke setiap pipeline.

Topik

- Menandai saluran pipa (konsol)
- Pipa Tag (CLI)

Menandai saluran pipa (konsol)

Anda dapat menggunakan konsol atau CLI untuk menandai sumber daya. Pipelines adalah satu-satunya CodePipeline sumber daya yang dapat dikelola dengan konsol atau CLI.

Topik

- Tambahkan tag ke pipeline (konsol)
- Lihat tag untuk pipeline (konsol)
- Edit tag untuk pipeline (konsol)
- Hapus tag dari pipeline (konsol)

Tambahkan tag ke pipeline (konsol)

Anda dapat menggunakan konsol untuk menambahkan tag ke pipeline yang ada.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

2. Pada halaman Pipelines, pilih pipeline tempat Anda ingin menambahkan tag.
3. Dari panel navigasi, pilih Pengaturan.
4. Di bawah tag Pipeline, pilih Edit.
5. Di bidang Kunci dan Nilai, masukkan pasangan kunci untuk setiap kumpulan tanda yang ingin Anda tambahkan. (Bidang Nilai adalah bersifat opsional.) Contohnya, dalam Key (Kunci), masukkan **`Project`**. Dalam Value (Nilai), masukkan **`ProjectA`**.
6. (Opsional) Pilih Tambahkan tanda untuk menambahkan lebih banyak baris dan masukkan lebih banyak tanda.
7. Pilih Kirim. Tag tercantum di bawah pengaturan pipa.

Lihat tag untuk pipeline (konsol)

Anda dapat menggunakan konsol untuk mencantumkan tag untuk pipeline yang ada.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Pipelines, pilih pipeline tempat Anda ingin melihat tag.
3. Dari panel navigasi, pilih Pengaturan.
4. Di bawah tag Pipeline, lihat tag untuk pipeline di bawah kolom Kunci dan Nilai.

Edit tag untuk pipeline (konsol)

Anda dapat menggunakan konsol untuk mengedit tag yang telah ditambahkan ke pipeline.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Pipelines, pilih pipeline tempat Anda ingin memperbarui tag.
3. Dari panel navigasi, pilih Pengaturan.
4. Di bawah tag Pipeline, pilih Edit.
5. Di bidang Key (Kunci) dan Value (Nilai), perbarui nilai di setiap bidang yang diperlukan. Misalnya, untuk kunci **`Project`**, di Nilai, ubah **`ProjectA`** ke **`ProjectB`**.
6. Pilih Kirim.

Hapus tag dari pipeline (konsol)

Anda dapat menggunakan konsol untuk menghapus tag dari pipeline. Saat Anda menghapus tanda dari sumber daya yang terkait, tanda akan dihapus.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Pipelines, pilih pipeline tempat Anda ingin menghapus tag.
3. Dari panel navigasi, pilih Pengaturan.
4. Di bawah tag Pipeline, pilih Edit.
5. Di samping kunci dan nilai untuk setiap tanda yang ingin Anda hapus, pilih Hapus tanda.
6. Pilih Kirim.

Pipa Tag (CLI)

Anda dapat menggunakan CLI untuk menandai sumber daya. Anda harus menggunakan konsol untuk mengelola tag di pipeline.

Topik

- Tambahkan tag ke pipeline (CLI)
- Lihat tag untuk pipeline (CLI)
- Edit tag untuk pipeline (CLI)
- Hapus tag dari pipeline (CLI)

Tambahkan tag ke pipeline (CLI)

Anda dapat menggunakan konsol atau AWS CLI untuk menandai pipeline.

Untuk menambahkan tag ke pipeline saat Anda membuatnya, lihatBuat pipeline, tahapan, dan tindakan.

Dalam langkah-langkah ini, kami menganggap bahwa Anda telah menginstal versi terbaru dari AWS CLI atau diperbarui ke versi terkini. Untuk informasi lebih lanjut, lihat Menginstal AWS Command Line Interface.

Di terminal atau baris perintah, jalankan tag-resource perintah, tentukan Nama Sumber Daya Amazon (ARN) dari pipeline tempat Anda ingin menambahkan tag dan kunci serta nilai

tag yang ingin Anda tambahkan. Anda dapat menambahkan lebih dari satu tag ke pipeline. Misalnya, untuk menandai pipeline bernama *MyPipeline* dengan dua tag, kunci tag bernama *DeploymentEnvironment* dengan nilai tag dari*Test*, dan kunci tag bernama *IscontainerBased* dengan nilai tag*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
```

Jika berhasil, perintah ini tidak mengembalikan apa pun.

Lihat tag untuk pipeline (CLI)

Ikuti langkah-langkah ini untuk menggunakan AWS CLI untuk melihat AWS tag untuk pipeline. Jika tidak ada tanda yang telah ditambahkan, daftar yang ditampilkan kosong.

Pada terminal atau baris perintah, jalankan perintah list-tags-for-resource. Misalnya, untuk melihat daftar kunci tag dan nilai tag untuk pipeline bernama *MyPipeline* dengan nilai `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
```

Jika berhasil, perintah ini menampilkan informasi yang serupa dengan yang berikut:

```
{
    "tags": {
        "Project": "ProjectA",
        "IscontainerBased": "true"
    }
}
```

Edit tag untuk pipeline (CLI)

Ikuti langkah-langkah ini untuk menggunakan AWS CLI untuk mengedit tag untuk pipeline. Anda dapat mengubah nilai untuk kunci yang ada atau menambahkan kunci lain. Anda juga dapat menghapus tag dari pipeline, seperti yang ditunjukkan pada bagian berikutnya.

Di terminal atau baris perintah, jalankan tag-resource perintah, tentukan ARN dari pipeline tempat Anda ingin memperbarui tag dan tentukan kunci tag dan nilai tag:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tags key=Project,value=ProjectA
```

Jika berhasil, perintah ini tidak mengembalikan apa pun.

Hapus tag dari pipeline (CLI)

Ikuti langkah-langkah ini untuk menggunakan AWS CLI untuk menghapus tag dari pipeline. Saat Anda menghapus tanda dari sumber daya yang terkait, tanda akan dihapus.

> **Note**
>
> Jika Anda menghapus pipeline, semua asosiasi tag akan dihapus dari pipeline yang dihapus. Anda tidak perlu menghapus tag sebelum menghapus pipeline.

Di terminal atau baris perintah, jalankan untag-resource perintah, tentukan ARN dari pipeline tempat Anda ingin menghapus tag dan kunci tag dari tag yang ingin Anda hapus. Misalnya, untuk menghapus beberapa tag pada pipeline bernama *MyPipeline* dengan kunci tag *Project* dan*IscontainerBased*:

```
aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:MyPipeline --tag-keys Project IscontainerBased
```

Jika berhasil, perintah ini tidak mengembalikan apa pun. Untuk memverifikasi tag yang terkait dengan pipeline, jalankan list-tags-for-resource perintah.

Membuat aturan notifikasi

Anda dapat menggunakan aturan notifikasi untuk memberi tahu pengguna tentang perubahan penting, seperti saat pipeline mulai dieksekusi. Aturan notifikasi menentukan peristiwa dan topik Amazon SNS yang digunakan untuk mengirim notifikasi. Untuk informasi selengkapnya, lihat Apa itu notifikasi?

Anda dapat menggunakan konsol atau AWS CLI untuk membuat aturan notifikasi untuk AWS CodePipeline.

Untuk membuat aturan notifikasi (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pilih Pipelines, lalu pilih pipeline tempat Anda ingin menambahkan notifikasi.
3. Pada halaman pipeline, pilih Beri tahu, lalu pilih Buat aturan pemberitahuan. Anda juga dapat pergi ke halaman Pengaturan untuk pipeline dan memilih Buat aturan pemberitahuan.
4. Di Nama notifikasi, masukkan nama untuk aturan.
5. Di Jenis detail, pilih Dasar jika Anda hanya menginginkan informasi yang diberikan ke Amazon yang EventBridge disertakan dalam notifikasi. Pilih Lengkap jika Anda ingin menyertakan informasi yang diberikan ke Amazon EventBridge dan informasi yang mungkin diberikan oleh CodePipeline atau pengelola notifikasi.

   Untuk informasi selengkapnya, lihat Memahami Konten Notifikasi dan Keamanan.
6. Di Peristiwa yang memicu notifikasi, pilih peristiwa yang ingin Anda kirimkan notifikasi. Untuk informasi selengkapnya, lihat Peristiwa untuk Aturan Pemberitahuan tentang Saluran Pipa.
7. Di Target, lakukan salah satu langkah berikut:

   - Jika Anda telah mengonfigurasi sumber daya untuk digunakan dengan notifikasi, di Pilih jenis target, pilih Pengembang Amazon Q di aplikasi obrolan (Slack) atau topik SNS. Di Pilih target, pilih nama klien (untuk klien Slack yang dikonfigurasi di Amazon Q Developer dalam aplikasi obrolan) atau Nama Sumber Daya Amazon (ARN) dari topik Amazon SNS (untuk topik Amazon SNS yang sudah dikonfigurasi dengan kebijakan yang diperlukan untuk pemberitahuan).
   - Jika Anda belum mengonfigurasi sumber daya untuk digunakan dengan notifikasi, pilih Buat target, lalu pilih Topik SNS. Berikan nama untuk topik setelah codestar-notifications-, lalu pilih Buat.

   > **Note**
   >
   > - Jika Anda membuat topik Amazon SNS sebagai bagian dari membuat aturan notifikasi, kebijakan yang memungkinkan fitur notifikasi untuk mempublikasikan peristiwa untuk topik diterapkan untuk Anda. Menggunakan topik yang dibuat untuk aturan notifikasi membantu memastikan bahwa Anda hanya berlangganan pengguna yang ingin Anda terima notifikasinya tentang sumber daya ini.

- Anda tidak dapat membuat Pengembang Amazon Q di klien aplikasi obrolan sebagai bagian dari pembuatan aturan notifikasi. Jika Anda memilih Pengembang Amazon Q dalam aplikasi obrolan (Slack), Anda akan melihat tombol yang mengarahkan Anda untuk mengonfigurasi klien di Amazon Q Developer dalam aplikasi obrolan. Memilih opsi itu membuka Pengembang Amazon Q di konsol aplikasi obrolan. Untuk informasi selengkapnya, lihat Mengonfigurasi Integrasi Antara Pemberitahuan dan Pengembang Amazon Q di aplikasi obrolan.
- Jika ingin menggunakan topik Amazon SNS yang ada sebagai target, Anda harus menambahkan kebijakan yang diperlukan untuk AWS CodeStar Pemberitahuan selain kebijakan lain yang mungkin ada untuk topik tersebut. Untuk informasi selengkapnya, lihat Konfigurasi Topik Amazon SNS untuk Notifikasi dan Memahami Konten Notifikasi dan Keamanan.

8. Untuk menyelesaikan pembuatan aturan, pilih Kirim.
9. Anda harus membuat pengguna berlangganan ke topik Amazon SNS untuk aturan tersebut sebelum mereka dapat menerima notifikasi. Untuk informasi selengkapnya, lihat Buat Pengguna Berlangganan ke Topik Amazon SNS yang Merupakan Target. Anda juga dapat mengatur integrasi antara notifikasi dan Pengembang Amazon Q di aplikasi obrolan untuk mengirim pemberitahuan ke ruang obrolan Amazon Chime atau saluran Slack. Untuk informasi selengkapnya, lihat Mengkonfigurasi Integrasi Antara Pemberitahuan dan Pengembang Amazon Q di aplikasi obrolan.

Membuat aturan notifikasi (AWS CLI)

1. Di terminal atau prompt perintah, jalankan perintah create-notification rule untuk menghasilkan kerangka JSON:

```
aws codestar-notifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

Anda dapat memberi nama file apa pun yang Anda inginkan. Dalam contoh ini, file dinamai *rule.json*.

2. Buka file JSON dalam editor plaintext dan edit untuk memasukkan sumber daya, jenis peristiwa, dan target yang Anda inginkan untuk aturan tersebut. Contoh berikut menunjukkan aturan notifikasi yang diberi nama **MyNotificationRule** untuk pipeline yang diberi nama *MyDemoPipeline* dalam AWS acccount dengan ID*123456789012*. Notifikasi dikirim dengan

tipe detail lengkap ke topik Amazon SNS yang diberi nama *codestar-notifications-MyNotificationTopic* saat eksekusi pipeline dimulai:

```
{
    "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"
}
```

Simpan file tersebut.

3. Menggunakan file yang baru saja Anda edit, di terminal atau baris perintah, jalankan create-notification-rule perintah lagi untuk membuat aturan notifikasi:

```
aws codestar-notifications create-notification-rule --cli-input-json
 file://rule.json
```

4. Jika berhasil, perintah tersebut mengembalikan ARN aturan notifikasi, yang serupa dengan berikut ini:

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

Tindakan sumber dan metode deteksi perubahan

Saat Anda menambahkan tindakan sumber ke pipeline, tindakan tersebut berfungsi dengan sumber daya tambahan yang dijelaskan dalam tabel.

> **Note**
>
> Tindakan sumber CodeCommit dan S3 memerlukan sumber daya deteksi perubahan yang dikonfigurasi (EventBridge aturan) atau menggunakan opsi untuk melakukan polling repositori untuk perubahan sumber. Untuk pipeline dengan Bitbucket, GitHub, atau tindakan sumber Server GitHub Perusahaan, Anda tidak perlu menyiapkan webhook atau default untuk polling. Tindakan koneksi mengelola deteksi perubahan untuk Anda.

Sumber	Menggunakan sumber daya tambahan?	Langkah-langkah	
Amazon S3 dengan sumber daya CloudTrail	Tindakan sumber ini menggunakan aturan acara dan CloudTrail sumber daya tambahan. Saat Anda menggunakan CLI atau CloudFormation untuk membuat tindakan ini, Anda juga membuat dan mengelola sumber daya ini.	Lihat Buat pipeline, tahapan, dan tindakan dan Menghubungkan ke tindakan sumber Amazon S3 yang menggunakan dan EventBridge AWS CloudTrail	
Amazon S3 tanpa sumber daya CloudTrail	Tindakan sumber ini menggunakan bucket yang diaktifkan untuk acara dengan aturan peristiwa tanpa memerlukan CloudTrail sumber daya tambahan. Saat Anda menggunakan	Lihat Buat pipeline, tahapan, dan tindakan dan Menyambungkan ke tindakan sumber Amazon S3 dengan sumber yang diaktifkan untuk acara	

Sumber	Menggunakan sumber daya tambahan?	Langkah-langkah	
	CLI atau CloudFormation untuk membuat tindakan ini, Anda juga membuat dan mengelola sumber daya ini.		
Bitbucket Cloud	Tindakan sumber ini menggunakan sumber daya koneksi.	Lihat Koneksi Bitbucket Cloud	
AWS CodeCommit	Amazon EventBridge (disarankan). Ini adalah default untuk pipeline dengan CodeCommit sumber yang dibuat atau diedit di konsol.	Lihat Buat pipeline, tahapan, dan tindakan dan CodeCommit tindakan sumber dan EventBridge	
Amazon ECR	Amazon EventBridge. Ini dibuat oleh wizard untuk saluran pipa dengan sumber ECR Amazon yang dibuat atau diedit di konsol.	Lihat Buat pipeline, tahapan, dan tindakan dan Tindakan sumber dan sumber daya Amazon ECR EventBridge.	
GitHub atau GitHub Enterprise Cloud	Tindakan sumber ini menggunakan sumber daya koneksi.	Lihat GitHub koneksi	
GitHub Server Perusahaan	Tindakan sumber ini menggunakan sumber daya koneksi dan sumber daya host.	Lihat GitHub Koneksi Enterprise Server	

Sumber	Menggunakan sumber daya tambahan?	Langkah-langkah	
GitLab.com	Tindakan sumber ini menggunakan sumber daya koneksi.	Lihat GitLabkoneksi .com	
GitLab dikelola sendiri	Tindakan sumber ini menggunakan sumber daya koneksi dan sumber daya host.	Lihat Koneksi untuk dikelola GitLab sendiri	

Jika Anda memiliki pipeline yang menggunakan polling, Anda dapat memperbaruinya untuk menggunakan metode deteksi yang disarankan. Untuk informasi selengkapnya, lihat Perbarui jalur pemungutan suara ke metode deteksi perubahan yang direkomendasikan.

Jika Anda ingin menonaktifkan deteksi perubahan untuk tindakan sumber yang menggunakan koneksi, lihatCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Connect ke penyedia sumber pihak pertama menggunakan tindakan sumber

Anda dapat menggunakan AWS CodePipeline konsol atau AWS CLI untuk terhubung ke penyedia tindakan sumber, seperti CodeCommit atau S3.

> **Note**
>
> Saat Anda menggunakan konsol untuk membuat atau mengedit pipeline, sumber daya deteksi perubahan akan dibuat untuk Anda. Jika Anda menggunakan AWS CLI untuk membuat pipeline, Anda harus membuat sumber daya tambahan sendiri. Untuk informasi selengkapnya, lihat CodeCommit tindakan sumber dan EventBridge.

Topik

- Tindakan sumber dan sumber daya Amazon ECR EventBridge
- Menyambungkan ke tindakan sumber Amazon S3 dengan sumber yang diaktifkan untuk acara
- Menghubungkan ke tindakan sumber Amazon S3 yang menggunakan dan EventBridge AWS CloudTrail
- CodeCommit tindakan sumber dan EventBridge

Tindakan sumber dan sumber daya Amazon ECR EventBridge

Untuk menambahkan aksi sumber ECR Amazon CodePipeline, Anda dapat memilih salah satu untuk:

- Gunakan wizard Create pipeline CodePipeline konsol (Buat pipeline khusus (konsol)) atau halaman tindakan Edit untuk memilih opsi penyedia Amazon ECR. Konsol membuat EventBridge aturan yang memulai pipeline Anda saat sumbernya berubah.
- Gunakan CLI untuk menambahkan konfigurasi tindakan untuk `ECR` tindakan dan membuat sumber daya tambahan sebagai berikut:
  - Gunakan `ECR` contoh konfigurasi tindakan Referensi tindakan sumber Amazon ECR untuk membuat tindakan Anda seperti yang ditunjukkan padaBuat pipeline (CLI).
  - Metode deteksi perubahan default untuk memulai pipeline dengan polling sumber. Anda harus menonaktifkan pemeriksaan berkala dan membuat aturan deteksi perubahan secara manual.

Gunakan salah satu metode berikut:Buat EventBridge aturan untuk sumber Amazon ECR (konsol),Buat EventBridge aturan untuk sumber Amazon ECR (CLI), atauBuat EventBridge aturan untuk sumber Amazon ECR (CloudFormation template).

Topik

- Buat EventBridge aturan untuk sumber Amazon ECR (konsol)
- Buat EventBridge aturan untuk sumber Amazon ECR (CLI)
- Buat EventBridge aturan untuk sumber Amazon ECR (CloudFormation template)

Buat EventBridge aturan untuk sumber Amazon ECR (konsol)

Untuk membuat EventBridge aturan untuk digunakan dalam CodePipeline operasi (sumber Amazon ECR)

1. Buka EventBridge konsol Amazon di https://console.aws.amazon.com/events/.
2. Di panel navigasi, pilih Peristiwa.
3. Pilih Buat aturan, lalu di bawah Sumber acara, dari Nama Layanan, pilih Elastic Container Registry (ECR).
4. Di Event Source, pilih Event Pattern.

   Pilih Edit, lalu tempel contoh pola peristiwa berikut di jendela Event Source untuk `eb-test` repositori dengan tag gambar: `cli-testing`

```
{
    "detail-type": [
        "ECR Image Action"
    ],
    "source": [
        "aws.ecr"
    ],
    "detail": {
        "action-type": [
            "PUSH"
        ],
        "image-tag": [
            "latest"
        ],
        "repository-name": [
```

```
            "eb-test"
        ],
        "result": [
            "SUCCESS"
        ]
    }
}
```

> **Note**
>
> Untuk melihat pola peristiwa lengkap yang didukung untuk peristiwa Amazon ECR, lihat Acara ECR Amazon dan atau Acara Registri Penampung Elastis EventBridge Amazon.

5. Pilih Simpan.

   Di panel Pratinjau Pola Peristiwa, lihat aturannya.

6. Di Target, pilih CodePipeline.
7. Masukkan ARN pipa agar pipa dimulai dengan aturan ini.

> **Note**
>
> Anda dapat menemukan ARN pipeline di output metadata setelah Anda menjalankan perintah. get-pipeline ARN pipa dibangun dalam format ini:
> arn:aws:codepipeline:: *region account pipeline-name*
> Contoh pipa ARN:
> `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

8. Buat atau tentukan peran layanan IAM yang memberikan EventBridge izin untuk memanggil target yang terkait dengan EventBridge aturan Anda (dalam hal ini, targetnya adalah). CodePipeline

   - Pilih Buat peran baru untuk sumber daya khusus ini untuk membuat peran layanan yang memberikan EventBridge izin untuk memulai eksekusi pipeline Anda.
   - Pilih Gunakan peran yang ada untuk memasukkan peran layanan yang memberikan EventBridge izin untuk memulai eksekusi pipeline Anda.

9. (Opsional) Untuk menentukan penggantian sumber dengan ID gambar tertentu, gunakan transformator input untuk meneruskan data sebagai parameter JSON. Anda juga dapat menggunakan transformator input untuk melewati variabel pipa.

- Perluas Pengaturan tambahan.

  Di bawah Konfigurasikan input target, pilih Konfigurasi transformator input.

  Di jendela dialog, pilih Enter my own. Dalam kotak jalur Input, ketik pasangan kunci-nilai berikut.

```
{"revisionValue": "$.detail.image-digest"}
```

- Dalam kotak Template, ketik pasangan kunci-nilai berikut.

```
{
    "sourceRevisions": [
        {
            "actionName": "Source",
            "revisionType": "IMAGE_DIGEST",
            "revisionValue": "<revisionValue>"
        }
    ],
     "variables": [
        {
            "name": "Variable_Name",
            "value": "Variable_Value"
        }
    ]
}
```

- Pilih Konfirmasi.

10. Tinjau pengaturan aturan Anda untuk memastikannya memenuhi persyaratan Anda.
11. Pilih Konfigurasikan detail.
12. Pada halaman Konfigurasi detail aturan, masukkan nama dan deskripsi untuk aturan, lalu pilih Status untuk mengaktifkan aturan.
13. Jika Anda puas dengan aturan, pilih Create rule (Buat aturan).

Buat EventBridge aturan untuk sumber Amazon ECR (CLI)

Panggil put-rule perintah, tentukan:

- Nama yang secara unik mengidentifikasi aturan yang Anda buat. Nama ini harus unik di semua pipeline yang Anda buat CodePipeline terkait dengan AWS akun Anda.
- Pola acara untuk bidang sumber dan detail yang digunakan oleh aturan. Untuk informasi selengkapnya, lihat Amazon EventBridge dan Pola Acara.

Untuk membuat EventBridge aturan dengan Amazon ECR sebagai sumber acara dan CodePipeline sebagai target

1. Tambahkan izin EventBridge untuk digunakan CodePipeline untuk menjalankan aturan. Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis sumber daya untuk Amazon. EventBridge

   a. Gunakan contoh berikut untuk membuat kebijakan kepercayaan yang memungkinkan EventBridge untuk mengambil peran layanan. Sebutkan kebijakan kepercayaan`trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Gunakan perintah berikut untuk membuat `Role-for-MyRule` peran dan melampirkan kebijakan kepercayaan.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

c. Buat kebijakan izin JSON, seperti yang ditunjukkan dalam contoh ini, untuk pipeline bernama. `MyFirstPipeline` Beri nama kebijakan `permissionspolicyforEB.json` izin.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Gunakan perintah berikut untuk melampirkan kebijakan `CodePipeline-Permissions-Policy-for-EB` izin ke `Role-for-MyRule` peran.

Mengapa saya membuat perubahan ini? Menambahkan kebijakan ini ke peran akan membuat izin untuk EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Panggil put-rule perintah dan sertakan`--name`,`--event-pattern`, dan `--role-arn` parameter.

Mengapa saya membuat perubahan ini? Anda harus membuat acara dengan aturan yang menentukan bagaimana push gambar harus dibuat, dan target yang memberi nama pipeline yang akan dimulai oleh acara tersebut.

Perintah contoh berikut membuat aturan yang disebut`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**
>
> Untuk melihat pola peristiwa lengkap yang didukung untuk peristiwa Amazon ECR, lihat Acara ECR Amazon dan atau Acara Registri Penampung Elastis EventBridge Amazon.

3. Untuk menambahkan CodePipeline sebagai target, panggil put-targets perintah dan sertakan parameter berikut:

   - `--rule`Parameter digunakan dengan yang `rule_name` Anda buat dengan menggunakanput-rule.
   - `--targets`Parameter digunakan dengan `Id` daftar target dalam daftar target dan pipa target. `ARN`

   Contoh perintah berikut menentukan bahwa untuk aturan yang dipanggil`MyECRRepoRule`, target `Id` terdiri dari nomor satu, menunjukkan bahwa dalam daftar target untuk aturan, ini adalah target 1. Perintah sample juga menentukan contoh `Arn` untuk pipeline dan contoh `RoleArn` untuk aturan. Pipeline dimulai ketika sesuatu berubah di repositori.

   ```
   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. (Opsional) Untuk mengonfigurasi transformator input dengan penggantian sumber untuk ID gambar tertentu, gunakan JSON berikut dalam perintah CLI Anda. Contoh berikut mengonfigurasi penggantian di mana:

   - `Source`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `actionName`
   - `IMAGE_DIGEST`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `revisionType`

- The`revisionValue`, < *revisionValue* > dalam contoh ini, berasal dari variabel peristiwa sumber.

```
{
    "Rule": "my-rule",
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "ARN",
            "InputTransformer": {
                "InputPathsMap": {
                    "revisionValue": "$.detail.image-digest"
                },
                "InputTemplate": {
                    "sourceRevisions": [
                        {
                            "actionName": "Source",
                            "revisionType": "IMAGE_DIGEST",
                            "revisionValue": "<revisionValue>"
                        }
                    ]
                }
            }
        }
    ]
}
```

Buat EventBridge aturan untuk sumber Amazon ECR (CloudFormation template)

Untuk digunakan CloudFormation untuk membuat aturan, gunakan cuplikan template seperti yang ditunjukkan di sini.

Untuk memperbarui CloudFormation template pipeline Anda dan membuat EventBridge aturan

1. Di template, di bawah`Resources`, gunakan `AWS::IAM::Role` CloudFormation sumber daya untuk mengonfigurasi peran IAM yang memungkinkan acara Anda memulai pipeline. Entri ini membuat peran yang menggunakan dua kebijakan:

- Kebijakan pertama memungkinkan peran diasumsikan.
- Kebijakan kedua memberikan izin untuk memulai pipeline.

Mengapa saya membuat perubahan ini? Anda harus membuat peran yang dapat diasumsikan oleh EventBridge untuk memulai eksekusi di pipeline kami.

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. Dalam template, di bawah`Resources`, gunakan `AWS::Events::Rule` CloudFormation sumber daya untuk menambahkan EventBridge aturan untuk sumber Amazon ECR. Pola acara ini membuat acara yang memantau komit ke repositori Anda. Saat EventBridge mendeteksi perubahan status repositori, aturan akan muncul di pipeline target Anda`StartPipelineExecution`.

Mengapa saya membuat perubahan ini? Anda harus membuat acara dengan aturan yang menentukan bagaimana push gambar harus dibuat, dan target yang memberi nama pipeline yang akan dimulai oleh acara tersebut.

Cuplikan ini menggunakan gambar bernama `eb-test` dengan tag. `latest`

YAML

```
EventRule:
  Type: 'AWS::Events::Rule'
  Properties:
    EventPattern:
      detail:
        action-type: [PUSH]
        image-tag: [latest]
        repository-name: [eb-test]
        result: [SUCCESS]
      detail-type: [ECR Image Action]
      source: [aws.ecr]
    Targets:
      - Arn: !Sub arn:aws:codepipeline:${AWS::Region}:${AWS::AccountId}:
${AppPipeline}
        RoleArn: !GetAtt
          - EventRole
          - Arn
        Id: codepipeline-AppPipeline
```

JSON

```
{
    "EventRule": {
        "Type": "AWS::Events::Rule",
        "Properties": {
            "EventPattern": {
                "detail": {
                    "action-type": [
                        "PUSH"
                    ],
                    "image-tag": [
                        "latest"
                    ],
```

```
                    "repository-name": [
                        "eb-test"
                    ],
                    "result": [
                        "SUCCESS"
                    ]
                },
                "detail-type": [
                    "ECR Image Action"
                ],
                "source": [
                    "aws.ecr"
                ]
            },
            "Targets": [
                {
                    "Arn": {
                        "Fn::Sub": "arn:aws:codepipeline:${AWS::Region}:
${AWS::AccountId}:${AppPipeline}"
                    },
                    "RoleArn": {
                        "Fn::GetAtt": [
                            "EventRole",
                            "Arn"
                        ]
                    },
                    "Id": "codepipeline-AppPipeline"
                }
            ]
        }
    }
},
```

> **Note**
>
> Untuk melihat pola peristiwa lengkap yang didukung untuk peristiwa Amazon ECR, lihat Acara ECR Amazon dan atau Acara Registri Penampung Elastis EventBridge Amazon.

3. (Opsional) Untuk mengonfigurasi transformator input dengan penggantian sumber untuk ID gambar tertentu, gunakan cuplikan YAMAL berikut. Contoh berikut mengonfigurasi penggantian di mana:

- `SourceDalam` contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `actionName`
- `IMAGE_DIGESTDalam` contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `revisionType`
- The`revisionValue`, **<** *`revisionValue`* **>** dalam contoh ini, berasal dari variabel peristiwa sumber.

```
---
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: ARN
  InputTransformer:
    InputPathsMap:
      revisionValue: "$.detail.image-digest"
    InputTemplate:
      sourceRevisions:
        actionName: Source
        revisionType: IMAGE_DIGEST
        revisionValue: '<revisionValue>'
```

4. Simpan template yang diperbarui ke komputer lokal Anda, lalu buka CloudFormation konsol.
5. Pilih tumpukan Anda, lalu pilih Create Change Set for Current Stack.
6. Unggah template, lalu lihat perubahan yang tercantum di dalamnya CloudFormation. Ini adalah perubahan yang harus dilakukan pada tumpukan. Anda harus melihat sumber daya baru Anda dalam daftar.
7. Pilih Eksekusi.

Menyambungkan ke tindakan sumber Amazon S3 dengan sumber yang diaktifkan untuk acara

Petunjuk di bagian ini memberikan langkah-langkah untuk membuat tindakan sumber S3 yang tidak mengharuskan Anda membuat atau mengelola AWS CloudTrail sumber daya.

> **Important**
>
> Prosedur untuk membuat tindakan ini tanpa AWS CloudTrail sumber daya tidak tersedia di konsol. Untuk menggunakan CLI, lihat prosedurnya di sini atau lihat. Migrasikan jalur pemungutan suara dengan sumber S3 yang diaktifkan untuk acara

Untuk pipeline dengan sumber Amazon S3, modifikasi pipeline sehingga deteksi perubahan otomatis melalui EventBridge dan dengan bucket sumber yang diaktifkan untuk pemberitahuan peristiwa. Ini adalah metode yang direkomendasikan jika Anda menggunakan CLI atau CloudFormation untuk memigrasikan pipeline Anda.

> **Note**
>
> Ini termasuk menggunakan bucket yang diaktifkan untuk pemberitahuan acara, di mana Anda tidak perlu membuat CloudTrail jejak terpisah. Jika Anda menggunakan konsol, maka aturan acara dan CloudTrail jejak disiapkan untuk Anda. Untuk langkah-langkah tersebut, lihatMigrasi jaringan pemungutan suara dengan sumber dan jejak S3 CloudTrail.

- CLI: Migrasi jaringan pemungutan suara dengan sumber dan jejak CloudTrail S3 (CLI)
- CloudFormation: Migrasikan jalur pemungutan suara dengan sumber dan CloudTrail jejak S3 (templat)CloudFormation

Buat pipeline dengan sumber S3 yang diaktifkan untuk acara (CLI)

Ikuti langkah-langkah ini untuk membuat pipeline dengan sumber S3 yang menggunakan peristiwa EventBridge untuk deteksi perubahan. Untuk langkah-langkah lengkap untuk membuat pipeline dengan CLI, lihat. Buat pipeline, tahapan, dan tindakan

Untuk membuat pipeline berbasis peristiwa dengan Amazon S3, Anda mengedit `PollForSourceChanges` parameter pipeline, lalu membuat sumber daya berikut:

- EventBridge aturan acara
- Peran IAM untuk memungkinkan EventBridge acara memulai pipeline Anda

Untuk membuat EventBridge aturan dengan Amazon S3 sebagai sumber peristiwa dan CodePipeline sebagai target dan menerapkan kebijakan izin

1. Berikan izin EventBridge untuk digunakan CodePipeline untuk menjalankan aturan. Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis sumber daya untuk Amazon. EventBridge

   a. Gunakan contoh berikut untuk membuat kebijakan kepercayaan agar memungkinkan EventBridge untuk mengambil peran layanan. Nama itu`trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Gunakan perintah berikut untuk membuat `Role-for-MyRule` peran dan melampirkan kebijakan kepercayaan.

      Mengapa saya membuat perubahan ini? Menambahkan kebijakan kepercayaan ini ke peran akan membuat izin untuk EventBridge.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Buat kebijakan izin JSON, seperti yang ditunjukkan di sini untuk pipeline bernama. `MyFirstPipeline` Beri nama kebijakan `permissionspolicyforEB.json` izin.

      JSON

      ```
      {
      ```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Gunakan perintah berikut untuk melampirkan kebijakan `CodePipeline-Permissions-Policy-for-EB` izin baru ke `Role-for-MyRule` peran yang Anda buat.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Panggil put-rule perintah dan sertakan`--name`,`--event-pattern`, dan `--role-arn` parameter.

   Perintah contoh berikut membuat aturan bernama`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. Untuk menambahkan CodePipeline sebagai target, panggil put-targets perintah dan sertakan `--targets` parameter `--rule` dan.

   Perintah berikut menentukan bahwa untuk aturan bernama`EnabledS3SourceRule`, target `Id` terdiri dari nomor satu, menunjukkan bahwa dalam daftar target untuk aturan, ini adalah target 1. Perintah ini juga menentukan contoh ARN untuk pipeline. Pipeline dimulai ketika sesuatu berubah di repositori.

```
aws events put-targets --rule EnabledS3SourceRule --targets Id=codepipeline-
AppPipeline,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

1. Jalankan get-pipeline perintah untuk menyalin struktur pipa ke file JSON. Misalnya, untuk pipeline bernama`MyFirstPipeline`, jalankan perintah berikut:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Perintah ini tidak mengembalikan apa pun, tetapi file yang Anda buat akan muncul di direktori tempat Anda menjalankan perintah.

2. Buka file JSON di editor teks biasa dan edit tahap sumber dengan mengubah `PollForSourceChanges` parameter untuk bucket bernama `amzn-s3-demo-source-bucketfalse`, seperti yang ditunjukkan dalam contoh ini.

   Mengapa saya membuat perubahan ini? Menyetel parameter ini untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Jika Anda bekerja dengan struktur pipa yang diambil menggunakan get-pipeline perintah, Anda harus menghapus `metadata` baris dari file JSON. Jika tidak, update-pipeline perintah tidak dapat menggunakannya. Hapus `"metadata": { }` garis dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang.

   Misalnya, hapus baris berikut dari struktur:

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

Simpan file tersebut.

4. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, dengan menentukan file JSON pipeline:

> **Important**
>
> Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

Perintah ini mengembalikan seluruh struktur pipa yang diedit.

> **Note**
>
> update-pipelinePerintah menghentikan pipa. Jika revisi sedang dijalankan melalui pipeline saat Anda menjalankan update-pipeline perintah, proses itu dihentikan. Anda harus memulai pipeline secara manual untuk menjalankan revisi itu melalui pipeline yang diperbarui. Gunakan start-pipeline-execution perintah untuk memulai pipeline Anda secara manual.

Buat pipeline dengan sumber S3 yang diaktifkan untuk acara (template)CloudFormation

Prosedur ini untuk pipeline di mana bucket sumber mengaktifkan peristiwa.

Gunakan langkah-langkah ini untuk membuat pipeline dengan sumber Amazon S3 untuk deteksi perubahan berbasis peristiwa.

Untuk membuat pipeline berbasis peristiwa dengan Amazon S3, Anda mengedit `PollForSourceChanges` parameter pipeline, lalu menambahkan sumber daya berikut ke templat:

- EventBridge aturan dan peran IAM untuk memungkinkan acara ini memulai pipeline Anda.

Jika Anda menggunakan CloudFormation untuk membuat dan mengelola pipeline Anda, template Anda menyertakan konten seperti berikut ini.

> **Note**
>
> `Configuration`Properti dalam tahap sumber disebut`PollForSourceChanges`. Jika template Anda tidak menyertakan properti itu, maka `PollForSourceChanges` disetel ke secara `true` default.

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
                            }
                        ]
                    },
```

```
...
```

Untuk membuat EventBridge aturan dengan Amazon S3 sebagai sumber peristiwa dan CodePipeline sebagai target dan menerapkan kebijakan izin

1. Di template, di bawah`Resources`, gunakan `AWS::IAM::Role` CloudFormation sumber daya untuk mengonfigurasi peran IAM yang memungkinkan acara Anda memulai pipeline. Entri ini membuat peran yang menggunakan dua kebijakan:

   - Kebijakan pertama memungkinkan peran diasumsikan.
   - Kebijakan kedua memberikan izin untuk memulai pipeline.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::IAM::Role` sumber daya memungkinkan CloudFormation untuk membuat izin untuk EventBridge. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. Gunakan `AWS::Events::Rule` CloudFormation sumber daya untuk menambahkan EventBridge aturan. Pola peristiwa ini membuat peristiwa yang memantau pembuatan atau penghapusan objek di bucket sumber Amazon S3 Anda. Selain itu, sertakan target pipa Anda. Ketika sebuah objek dibuat, aturan ini dipanggil `StartPipelineExecution` pada pipeline target Anda.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::Events::Rule` sumber daya memungkinkan CloudFormation untuk membuat acara. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. Simpan template Anda yang diperbarui ke komputer lokal Anda, dan buka CloudFormation konsol.
4. Pilih tumpukan Anda, lalu pilih Create Change Set for Current Stack.
5. Unggah template Anda yang diperbarui, lalu lihat perubahan yang tercantum di dalamnya CloudFormation. Ini adalah perubahan yang akan dilakukan pada tumpukan. Anda harus melihat sumber daya baru Anda dalam daftar.
6. Pilih Eksekusi.

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan

mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

- Dalam template, ubah `PollForSourceChanges` ke`false`. Jika Anda tidak menyertakan `PollForSourceChanges` dalam definisi pipeline Anda, tambahkan dan setel ke`false`.

  Mengapa saya membuat perubahan ini? Mengubah `PollForSourceChanges` untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

  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

Saat Anda menggunakan CloudFormation untuk membuat sumber daya ini, pipeline Anda dipicu saat file di repositori dibuat atau diperbarui.

> **Note**
>
> Jangan berhenti di sini. Meskipun pipeline Anda dibuat, Anda harus membuat CloudFormation template kedua untuk pipeline Amazon S3 Anda. Jika Anda tidak membuat template kedua, pipeline Anda tidak memiliki fungsi deteksi perubahan.

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"
        }
    ]
}
```

Menghubungkan ke tindakan sumber Amazon S3 yang menggunakan dan EventBridge AWS CloudTrail

Petunjuk di bagian ini memberikan langkah-langkah untuk membuat tindakan sumber S3 yang menggunakan sumber AWS CloudTrail daya yang harus Anda buat dan kelola. Untuk menggunakan tindakan sumber S3 dengan EventBridge itu tidak memerlukan AWS CloudTrail sumber daya tambahan, gunakan instruksi CLI di. Migrasikan jalur pemungutan suara dengan sumber S3 yang diaktifkan untuk acara

> **Important**
>
> Prosedur ini menyediakan langkah-langkah untuk membuat tindakan sumber S3 yang menggunakan sumber AWS CloudTrail daya yang harus Anda buat dan kelola. Prosedur untuk membuat tindakan ini tanpa AWS CloudTrail sumber daya tidak tersedia di konsol. Untuk menggunakan CLI, lihat. Migrasikan jalur pemungutan suara dengan sumber S3 yang diaktifkan untuk acara

Untuk menambahkan aksi sumber Amazon S3 CodePipeline, Anda memilih salah satu untuk:

- Gunakan CodePipeline konsol Create pipeline wizard (Buat pipeline khusus (konsol)) atau Edit halaman tindakan untuk memilih opsi penyedia S3. Konsol membuat EventBridge aturan dan CloudTrail jejak yang memulai pipeline Anda saat sumbernya berubah.
- Gunakan AWS CLI untuk menambahkan konfigurasi tindakan untuk S3 tindakan dan membuat sumber daya tambahan sebagai berikut:
  - Gunakan S3 contoh konfigurasi tindakan Referensi tindakan sumber Amazon S3 untuk membuat tindakan Anda seperti yang ditunjukkan padaBuat pipeline (CLI).
  - Metode deteksi perubahan default untuk memulai pipeline dengan polling sumber. Anda harus menonaktifkan pemeriksaan berkala dan membuat aturan deteksi perubahan dan jejak secara manual. Gunakan salah satu metode berikut:Buat EventBridge aturan untuk sumber Amazon S3 (konsol),Buat EventBridge aturan untuk sumber Amazon S3 (CLI), atauBuat EventBridge aturan untuk sumber Amazon S3 (template)CloudFormation.

AWS CloudTrail adalah layanan yang mencatat dan memfilter peristiwa di bucket sumber Amazon S3 Anda. Jejak mengirimkan perubahan sumber yang difilter ke EventBridge aturan. EventBridge Aturan mendeteksi perubahan sumber dan kemudian memulai pipeline Anda.

Persyaratan:

- Jika Anda tidak membuat jejak, gunakan jejak yang ada AWS CloudTrail untuk mencatat peristiwa di bucket sumber Amazon S3 dan mengirim peristiwa yang difilter ke aturan. EventBridge
- Buat atau gunakan bucket S3 yang ada di mana AWS CloudTrail dapat menyimpan file lognya. AWS CloudTrail harus memiliki izin yang diperlukan untuk mengirimkan file log ke bucket Amazon S3. Bucket tidak dapat dikonfigurasi sebagai bucket Requester Pays. Saat Anda membuat bucket Amazon S3 sebagai bagian dari membuat atau memperbarui jejak di konsol, AWS CloudTrail lampirkan izin yang diperlukan ke bucket untuk Anda. Untuk informasi selengkapnya, lihat Kebijakan Bucket Amazon S3 untuk. CloudTrail

Buat EventBridge aturan untuk sumber Amazon S3 (konsol)

Sebelum Anda membuat aturan EventBridge, Anda harus membuat AWS CloudTrail jejak. Untuk informasi selengkapnya, lihat Membuat Jejak di Konsol.

> **Important**
>
> Jika Anda menggunakan konsol untuk membuat atau mengedit pipeline, EventBridge aturan dan AWS CloudTrail jejak dibuat untuk Anda.

Untuk membuat jejak

1. Buka AWS CloudTrail konsol.
2. Di panel navigasi, pilih Jejak.
3. Pilih Buat jejak. Untuk nama Trail, masukkan nama untuk jejak Anda.
4. Di bawah Lokasi penyimpanan, buat atau tentukan bucket yang akan digunakan untuk menyimpan file log. Secara default, ember dan objek Amazon S3 bersifat pribadi. Hanya pemilik sumber daya (AWS akun yang membuat bucket) yang dapat mengakses bucket dan objeknya. Bucket harus memiliki kebijakan sumber daya yang memungkinkan AWS CloudTrail izin mengakses objek di bucket.
5. Di bawah Bucket dan folder log Trail, tentukan bucket Amazon S3 dan awalan objek (nama folder) untuk mencatat peristiwa data untuk semua objek di folder. Untuk setiap jejak, Anda dapat menambahkan hingga 250 objek Amazon S3. Lengkapi informasi kunci enkripsi yang diperlukan dan pilih Berikutnya.
6. Untuk jenis Acara, pilih Acara manajemen.
7. Untuk acara Manajemen, pilih Tulis. Jejak merekam aktivitas API tingkat objek Amazon S3 (misalnya, `GetObject` dan`PutObject`) pada bucket dan awalan yang ditentukan.
8. Pilih Tulis.
9. Jika Anda puas dengan jejak, pilih Buat jejak.

Untuk membuat EventBridge aturan yang menargetkan pipeline Anda dengan sumber Amazon S3

1. Buka EventBridge konsol Amazon di https://console.aws.amazon.com/events/.
2. Di panel navigasi, pilih Aturan. Biarkan bus default dipilih atau pilih bus acara. Pilih Buat aturan.
3. Di Nama, masukkan nama untuk aturan Anda.
4. Di bawah Jenis aturan, pilih Aturan dengan pola acara. Pilih Berikutnya.
5. Di bawah Sumber acara, pilih AWS acara atau acara EventBridge mitra.
6. Di bawah Contoh jenis acara, pilih AWS acara.

7. Dalam contoh peristiwa, ketik S3 sebagai kata kunci untuk memfilter. Pilih panggilan AWS API melalui CloudTrail.
8. Di bawah metode Creation, pilih pola Pelanggan (editor JSON).

   Tempel pola acara yang disediakan di bawah ini. Pastikan untuk menambahkan nama bucket dan kunci objek S3 (atau nama kunci) yang secara unik mengidentifikasi objek dalam ember sebagai. `requestParameters` Dalam contoh ini, aturan dibuat untuk bucket bernama `amzn-s3-demo-source-bucket` dan kunci objek dari`my-files.zip`. Saat Anda menggunakan jendela Edit untuk menentukan sumber daya, aturan Anda diperbarui untuk menggunakan pola peristiwa khusus.

   Berikut ini adalah contoh pola acara untuk menyalin dan menempel:

```
{
    "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. Pilih Berikutnya.

10. Di jenis Target, pilih AWS layanan.
11. Di Pilih target, pilih CodePipeline. Di ARN Pipeline, masukkan ARN pipa agar pipa dimulai dengan aturan ini.

> **Note**
>
> Untuk mendapatkan ARN pipeline, jalankan perintah. get-pipeline ARN pipa muncul di output. Itu dibangun dalam format ini:
> arn:aws:codepipeline:: *region account pipeline-name*
> Contoh pipa ARN:
> arn:aws:codepipeline:us-east- 2:80398 CONTOH: MyFirstPipeline

12. Untuk membuat atau menentukan peran layanan IAM yang memberikan EventBridge izin untuk memanggil target yang terkait dengan EventBridge aturan Anda (dalam hal ini, targetnya adalah): CodePipeline

    - Pilih Buat peran baru untuk sumber daya khusus ini untuk membuat peran layanan yang memberikan EventBridge izin untuk memulai eksekusi pipeline Anda.
    - Pilih Gunakan peran yang ada untuk memasukkan peran layanan yang memberikan EventBridge izin untuk memulai eksekusi pipeline Anda.

13. (Opsional) Untuk menentukan penggantian sumber dengan ID gambar tertentu, gunakan transformator input untuk meneruskan data sebagai parameter JSON. Anda juga dapat menggunakan transformator input untuk melewati variabel pipa.

    - Perluas Pengaturan tambahan.

      Di bawah Konfigurasikan input target, pilih Konfigurasi transformator input.

      Di jendela dialog, pilih Enter my own. Dalam kotak jalur Input, ketik pasangan kunci-nilai berikut.

      ```
      {"revisionValue": "$.detail.object.version-id"}
      ```

    - Dalam kotak Template, ketik pasangan kunci-nilai berikut.

      ```
                                {
          "sourceRevisions": [
      ```

```
        {
            "actionName": "Source",
            "revisionType": "S3_OBJECT_VERSION_ID",
            "revisionValue": "<revisionValue>"
        }
    ],
     "variables": [
        {
            "name": "Variable_Name",
            "value": "Variable_Value"
        }
    ]
}
```

- Pilih Konfirmasi.

14. Pilih Berikutnya.
15. Pada halaman Tag, pilih Berikutnya.
16. Pada halaman Tinjau dan buat, tinjau konfigurasi aturan. Jika Anda puas dengan aturan, pilih Create rule (Buat aturan).

Buat EventBridge aturan untuk sumber Amazon S3 (CLI)

Untuk membuat AWS CloudTrail jejak dan mengaktifkan logging

Untuk menggunakan AWS CLI untuk membuat jejak, panggil create-trail perintah, dengan menentukan:

- Nama jejak.
- Bucket tempat Anda telah menerapkan kebijakan bucket AWS CloudTrail.

Untuk informasi selengkapnya, lihat Membuat jejak dengan antarmuka baris AWS perintah.

1. Panggil create-trail perintah dan sertakan `--s3-bucket-name` parameter `--name` dan.

   Mengapa saya membuat perubahan ini? Ini menciptakan CloudTrail jejak yang diperlukan untuk bucket sumber S3 Anda.

Perintah berikut menggunakan `--name` dan `--s3-bucket-name` untuk membuat jejak bernama `my-trail` dan ember bernama`amzn-s3-demo-source-bucket`.

```
aws cloudtrail create-trail --name my-trail --s3-bucket-name amzn-s3-demo-source-bucket
```

2. Panggil start-logging perintah dan sertakan `--name` parameternya.

   Mengapa saya membuat perubahan ini? Perintah ini memulai CloudTrail pencatatan untuk bucket sumber Anda dan mengirimkan acara ke EventBridge.

   Contoh:

   Perintah berikut digunakan `--name` untuk memulai logging pada jejak bernama`my-trail`.

   ```
   aws cloudtrail start-logging --name my-trail
   ```

3. Panggil put-event-selectors perintah dan sertakan `--event-selectors` parameter `--trail-name` dan. Gunakan penyeleksi peristiwa untuk menentukan bahwa jejak Anda ingin mencatat peristiwa data untuk bucket sumber Anda dan mengirim peristiwa ke EventBridge aturan.

   Mengapa saya membuat perubahan ini? Perintah ini menyaring peristiwa.

   Contoh:

   Perintah berikut menggunakan `--trail-name` dan `--event-selectors` menentukan peristiwa data untuk bucket sumber dan awalan bernama`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"] }] }]'
   ```

Untuk membuat EventBridge aturan dengan Amazon S3 sebagai sumber peristiwa dan CodePipeline sebagai target dan menerapkan kebijakan izin

1. Berikan izin EventBridge untuk digunakan CodePipeline untuk menjalankan aturan. Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis sumber daya untuk Amazon. EventBridge

   a. Gunakan contoh berikut untuk membuat kebijakan kepercayaan agar memungkinkan EventBridge untuk mengambil peran layanan. Nama itu`trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Gunakan perintah berikut untuk membuat `Role-for-MyRule` peran dan melampirkan kebijakan kepercayaan.

      Mengapa saya membuat perubahan ini? Menambahkan kebijakan kepercayaan ini ke peran akan membuat izin untuk EventBridge.

      ```
      aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
       file://trustpolicyforEB.json
      ```

   c. Buat kebijakan izin JSON, seperti yang ditunjukkan di sini untuk pipeline bernama. `MyFirstPipeline` Beri nama kebijakan `permissionspolicyforEB.json` izin.

      JSON

      ```
      {
      ```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Gunakan perintah berikut untuk melampirkan kebijakan `CodePipeline-Permissions-Policy-for-EB` izin baru ke `Role-for-MyRule` peran yang Anda buat.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Panggil put-rule perintah dan sertakan`--name`,`--event-pattern`, dan `--role-arn` parameter.

   Perintah contoh berikut membuat aturan bernama`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. Untuk menambahkan CodePipeline sebagai target, panggil put-targets perintah dan sertakan `--targets` parameter `--rule` dan.

   Perintah berikut menentukan bahwa untuk aturan bernama`MyS3SourceRule`, target `Id` terdiri dari nomor satu, menunjukkan bahwa dalam daftar target untuk aturan, ini adalah target 1. Perintah ini juga menentukan contoh `ARN` untuk pipeline. Pipeline dimulai ketika sesuatu berubah di repositori.

```
aws events put-targets --rule MyS3SourceRule --targets
 Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
```

4. (Opsional) Untuk mengonfigurasi transformator input dengan penggantian sumber untuk ID gambar tertentu, gunakan JSON berikut dalam perintah CLI Anda. Contoh berikut mengonfigurasi penggantian di mana:

   - `Source`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `actionName`
   - `S3_OBJECT_VERSION_ID`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `revisionType`
   - The`revisionValue`, **<** *`revisionValue`* **>** dalam contoh ini, berasal dari variabel peristiwa sumber.

```
{
    "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>"
                    }
                }
            }
        }
    ]
}
```

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

1. Jalankan get-pipeline perintah untuk menyalin struktur pipa ke file JSON. Misalnya, untuk pipeline bernama`MyFirstPipeline`, jalankan perintah berikut:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Perintah ini tidak mengembalikan apa pun, tetapi file yang Anda buat akan muncul di direktori tempat Anda menjalankan perintah.

2. Buka file JSON di editor teks biasa dan edit tahap sumber dengan mengubah `PollForSourceChanges` parameter untuk bucket bernama `amzn-s3-demo-source-bucketfalse`, seperti yang ditunjukkan dalam contoh ini.

   Mengapa saya membuat perubahan ini? Menyetel parameter ini untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

   ```
   "configuration": {
       "S3Bucket": "amzn-s3-demo-source-bucket",
       "PollForSourceChanges": "false",
       "S3ObjectKey": "index.zip"
   },
   ```

3. Jika Anda bekerja dengan struktur pipa yang diambil menggunakan get-pipeline perintah, Anda harus menghapus `metadata` baris dari file JSON. Jika tidak, update-pipeline perintah tidak dapat menggunakannya. Hapus `"metadata": { }` garis dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang.

   Misalnya, hapus baris berikut dari struktur:

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

Simpan file tersebut.

4. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, dengan menentukan file JSON pipeline:

> **Important**
>
> Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

Perintah ini mengembalikan seluruh struktur pipa yang diedit.

> **Note**
>
> update-pipelinePerintah menghentikan pipa. Jika revisi sedang dijalankan melalui pipeline saat Anda menjalankan update-pipeline perintah, proses itu dihentikan. Anda harus memulai pipeline secara manual untuk menjalankan revisi itu melalui pipeline yang diperbarui. Gunakan start-pipeline-execution perintah untuk memulai pipeline Anda secara manual.

Buat EventBridge aturan untuk sumber Amazon S3 (template)CloudFormation

Untuk digunakan CloudFormation untuk membuat aturan, perbarui template Anda seperti yang ditunjukkan di sini.

Untuk membuat EventBridge aturan dengan Amazon S3 sebagai sumber peristiwa dan CodePipeline sebagai target dan menerapkan kebijakan izin

1. Di template, di bawah`Resources`, gunakan `AWS::IAM::Role` CloudFormation sumber daya untuk mengonfigurasi peran IAM yang memungkinkan acara Anda memulai pipeline. Entri ini membuat peran yang menggunakan dua kebijakan:

   - Kebijakan pertama memungkinkan peran diasumsikan.
   - Kebijakan kedua memberikan izin untuk memulai pipeline.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::IAM::Role` sumber daya memungkinkan CloudFormation untuk membuat izin untuk EventBridge. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. Gunakan `AWS::Events::Rule` CloudFormation sumber daya untuk menambahkan EventBridge aturan. Pola acara ini membuat acara yang memantau`CopyObject`, `PutObject` dan `CompleteMultipartUpload` di bucket sumber Amazon S3 Anda. Selain itu, sertakan target pipa Anda. Ketika`CopyObject`,`PutObject`, atau `CompleteMultipartUpload` terjadi, aturan ini muncul `StartPipelineExecution` di pipeline target Anda.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::Events::Rule` sumber daya memungkinkan CloudFormation untuk membuat acara. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. Tambahkan cuplikan ini ke template pertama Anda untuk memungkinkan fungsionalitas cross-stack:

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. (Opsional) Untuk mengonfigurasi transformator input dengan penggantian sumber untuk ID gambar tertentu, gunakan cuplikan YAMAL berikut. Contoh berikut mengonfigurasi penggantian di mana:

   - `Source`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `actionName`
   - `S3_OBJECT_VERSION_ID`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `revisionType`
   - The`revisionValue`, **<** *`revisionValue`* **>** dalam contoh ini, berasal dari variabel peristiwa sumber.

```
---
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. Simpan template Anda yang diperbarui ke komputer lokal Anda, dan buka CloudFormation konsol.
6. Pilih tumpukan Anda, lalu pilih Create Change Set for Current Stack.
7. Unggah template Anda yang diperbarui, lalu lihat perubahan yang tercantum di dalamnya CloudFormation. Ini adalah perubahan yang akan dilakukan pada tumpukan. Anda harus melihat sumber daya baru Anda dalam daftar.
8. Pilih Eksekusi.

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

- Dalam template, ubah `PollForSourceChanges` k`efalse`. Jika Anda tidak menyertakan `PollForSourceChanges` dalam definisi pipeline Anda, tambahkan dan setel k`efalse`.

  Mengapa saya membuat perubahan ini? Mengubah `PollForSourceChanges` untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

  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
 }
```

Untuk membuat template kedua untuk sumber daya saluran Amazon S3 Anda CloudTrail

- Dalam templat terpisah, di bawah`Resources`, gunakan`AWS::S3::Bucket`,`AWS::S3::BucketPolicy`, dan `AWS::CloudTrail::Trail` CloudFormation sumber daya untuk memberikan definisi dan jejak bucket sederhana CloudTrail.

  Mengapa saya membuat perubahan ini? Mengingat batas saat ini lima jalur per akun, CloudTrail jejak harus dibuat dan dikelola secara terpisah. (Lihat Batas di AWS CloudTrail.) Namun, Anda dapat menyertakan banyak bucket Amazon S3 pada satu jalur, sehingga Anda dapat membuat jejak sekali dan kemudian menambahkan bucket Amazon S3 untuk saluran pipa lain seperlunya. Tempelkan berikut ini ke dalam file template sampel kedua Anda.

  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 tindakan sumber dan EventBridge

Untuk menambahkan aksi CodeCommit sumber CodePipeline, Anda dapat memilih salah satu untuk:

- Gunakan CodePipeline konsol Create pipeline wizard (Buat pipeline khusus (konsol)) atau Edit halaman tindakan untuk memilih opsi CodeCommitpenyedia. Konsol membuat EventBridge aturan yang memulai pipeline Anda saat sumbernya berubah.
- Gunakan AWS CLI untuk menambahkan konfigurasi tindakan untuk `CodeCommit` tindakan dan membuat sumber daya tambahan sebagai berikut:
  - Gunakan `CodeCommit` contoh konfigurasi tindakan CodeCommit referensi tindakan sumber untuk membuat tindakan Anda seperti yang ditunjukkan padaBuat pipeline (CLI).

- Metode deteksi perubahan default untuk memulai pipeline dengan polling sumber. Anda harus menonaktifkan pemeriksaan berkala dan membuat aturan deteksi perubahan secara manual. Gunakan salah satu metode berikut:Buat EventBridge aturan untuk CodeCommit sumber (konsol),Buat EventBridge aturan untuk CodeCommit sumber (CLI), atauBuat EventBridge aturan untuk CodeCommit sumber (CloudFormation template).

Topik

- Buat EventBridge aturan untuk CodeCommit sumber (konsol)
- Buat EventBridge aturan untuk CodeCommit sumber (CLI)
- Buat EventBridge aturan untuk CodeCommit sumber (CloudFormation template)

Buat EventBridge aturan untuk CodeCommit sumber (konsol)

> **Important**
>
> Jika Anda menggunakan konsol untuk membuat atau mengedit pipeline, EventBridge aturan dibuat untuk Anda.

Untuk membuat EventBridge aturan untuk digunakan dalam CodePipeline operasi

1. Buka EventBridge konsol Amazon di https://console.aws.amazon.com/events/.
2. Di panel navigasi, pilih Aturan. Biarkan bus default dipilih atau pilih bus acara. Pilih Buat aturan.
3. Di Nama, masukkan nama untuk aturan Anda.
4. Di bawah Jenis aturan, pilih Aturan dengan pola acara. Pilih Berikutnya.
5. Di bawah Sumber acara, pilih AWS acara atau acara EventBridge mitra.
6. Di bawah Contoh jenis acara, pilih AWS acara.
7. Dalam contoh peristiwa, ketik CodeCommit sebagai kata kunci untuk memfilter. Pilih Perubahan Status CodeCommit Repositori.
8. Di bawah metode Creation, pilih pola Pelanggan (editor JSON).

   Tempel pola acara yang disediakan di bawah ini. Berikut ini adalah contoh pola CodeCommit peristiwa di jendela Event untuk `MyTestRepo` repositori dengan cabang bernama: `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. Di Target, pilih CodePipeline.
10. Masukkan ARN pipa agar pipa dimulai dengan aturan ini.

> **Note**
>
> Anda dapat menemukan ARN pipeline di output metadata setelah Anda menjalankan perintah. get-pipeline ARN pipa dibangun dalam format ini: arn:aws:codepipeline:: *region account pipeline-name*
> Contoh pipa ARN:
> `arn:aws:codepipeline:us-east-2:80398EXAMPLE:MyFirstPipeline`

11. Untuk membuat atau menentukan peran layanan IAM yang memberikan EventBridge izin untuk memanggil target yang terkait dengan EventBridge aturan Anda (dalam hal ini, targetnya adalah): CodePipeline

    - Pilih Buat peran baru untuk sumber daya khusus ini untuk membuat peran layanan yang memberikan EventBridge izin untuk memulai eksekusi pipeline Anda.
    - Pilih Gunakan peran yang ada untuk memasukkan peran layanan yang memberikan EventBridge izin untuk memulai eksekusi pipeline Anda.

12. (Opsional) Untuk menentukan penggantian sumber dengan ID gambar tertentu, gunakan transformator input untuk meneruskan data sebagai parameter JSON. Anda juga dapat menggunakan transformator input untuk melewati variabel pipa.

    - Perluas Pengaturan tambahan.

      Di bawah Konfigurasikan input target, pilih Konfigurasi transformator input.

      Di jendela dialog, pilih Enter my own. Dalam kotak jalur Input, ketik pasangan kunci-nilai berikut.

      ```
      {"revisionValue": "$.detail.image-digest"}
      ```

    - Dalam kotak Template, ketik pasangan kunci-nilai berikut.

      ```
      {
          "sourceRevisions": [
              {
                  "actionName": "Source",
                  "revisionType": "COMMIT_ID",
                  "revisionValue": "<revisionValue>"
              }
          ],
          "variables": [
              {
                  "name": "Branch_Name",
                  "value": "value"
              }
          ]
      }
      ```

    - Pilih Konfirmasi.

13. Pilih Berikutnya.
14. Pada halaman Tag, pilih Berikutnya.
15. Pada halaman Tinjau dan buat, tinjau konfigurasi aturan. Jika Anda puas dengan aturan, pilih Create rule (Buat aturan).

Buat EventBridge aturan untuk CodeCommit sumber (CLI)

Panggil put-rule perintah, tentukan:

- Nama yang secara unik mengidentifikasi aturan yang Anda buat. Nama ini harus unik di semua pipeline yang Anda buat CodePipeline terkait dengan AWS akun Anda.
- Pola acara untuk bidang sumber dan detail yang digunakan oleh aturan. Untuk informasi selengkapnya, lihat Amazon EventBridge dan Pola Acara.

Untuk membuat EventBridge aturan dengan CodeCommit sebagai sumber acara dan CodePipeline sebagai target

1. Tambahkan izin EventBridge untuk digunakan CodePipeline untuk menjalankan aturan. Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis sumber daya untuk Amazon. EventBridge

   a. Gunakan contoh berikut untuk membuat kebijakan kepercayaan yang memungkinkan EventBridge untuk mengambil peran layanan. Sebutkan kebijakan kepercayaan`trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
                  },
                  "Action": "sts:AssumeRole"
              }
          ]
      }
      ```

   b. Gunakan perintah berikut untuk membuat `Role-for-MyRule` peran dan melampirkan kebijakan kepercayaan.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Buat kebijakan izin JSON, seperti yang ditunjukkan dalam contoh ini, untuk pipeline bernama. `MyFirstPipeline` Beri nama kebijakan `permissionspolicyforEB.json` izin.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Gunakan perintah berikut untuk melampirkan kebijakan `CodePipeline-Permissions-Policy-for-EB` izin ke `Role-for-MyRule` peran.

Mengapa saya membuat perubahan ini? Menambahkan kebijakan ini ke peran akan membuat izin untuk EventBridge.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforEB.json
```

2. Panggil put-rule perintah dan sertakan`--name`,`--event-pattern`, dan `--role-arn` parameter.

Mengapa saya membuat perubahan ini? Perintah ini CloudFormation memungkinkan untuk membuat acara.

Perintah contoh berikut membuat aturan yang disebut`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. Untuk menambahkan CodePipeline sebagai target, panggil put-targets perintah dan sertakan parameter berikut:

   - `--rule`Parameter digunakan dengan yang `rule_name` Anda buat dengan menggunakan`put-rule`.
   - `--targets`Parameter digunakan dengan `Id` daftar target dalam daftar target dan pipa target. ARN

   Contoh perintah berikut menentukan bahwa untuk aturan yang dipanggil`MyCodeCommitRepoRule`, target `Id` terdiri dari nomor satu, menunjukkan bahwa dalam daftar target untuk aturan, ini adalah target 1. Perintah sample juga menentukan contoh ARN untuk pipeline. Pipeline dimulai ketika sesuatu berubah di repositori.

   ```
   aws events put-targets --rule MyCodeCommitRepoRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

4. (Opsional) Untuk mengonfigurasi transformator input dengan penggantian sumber untuk ID gambar tertentu, gunakan JSON berikut dalam perintah CLI Anda. Contoh berikut mengonfigurasi penggantian di mana:

   - `Source`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `actionName`
   - `COMMIT_ID`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `revisionType`
   - The`revisionValue`, **<** *`revisionValue`* **>** dalam contoh ini, berasal dari variabel peristiwa sumber.

   ```
   {
       "Rule": "my-rule",
   ```

```
    "Targets": [
        {
            "Id": "MyTargetId",
            "Arn": "pipeline-ARN",
            "InputTransformer": {
                "sourceRevisions": {
                    "actionName": "Source",
                    "revisionType": "COMMIT_ID",
                    "revisionValue": "<revisionValue>"
                },
                "variables": [
                    {
                        "name": "Branch_Name",
                        "value": "value"
                    }
                ]
            }
        }
    ]
}
```

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Saat Anda membuat pipeline dengan metode ini, `PollForSourceChanges` parameter default ke true jika tidak secara eksplisit disetel ke false. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

1. Jalankan get-pipeline perintah untuk menyalin struktur pipa ke file JSON. Misalnya, untuk pipeline bernama`MyFirstPipeline`, jalankan perintah berikut:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Perintah ini tidak mengembalikan apa pun, tetapi file yang Anda buat akan muncul di direktori tempat Anda menjalankan perintah.

2. Buka file JSON di editor teks biasa dan edit tahap sumber dengan mengubah `PollForSourceChanges` parameter menjadi`false`, seperti yang ditunjukkan dalam contoh ini.

   Mengapa saya membuat perubahan ini? Mengubah parameter ini untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

   ```
   "configuration": {
       "PollForSourceChanges": "false",
       "BranchName": "main",
       "RepositoryName": "MyTestRepo"
   },
   ```

3. Jika Anda bekerja dengan struktur pipa yang diambil menggunakan get-pipeline perintah, hapus `metadata` baris dari file JSON. Jika tidak, update-pipeline perintah tidak dapat menggunakannya. Hapus `"metadata": { }` garis dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang.

   Misalnya, hapus baris berikut dari struktur:

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

   Simpan file tersebut.

4. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, dengan menentukan file JSON pipeline:

   > **Important**
   >
   > Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

   Perintah ini mengembalikan seluruh struktur pipa yang diedit.

> **Note**
>
> update-pipelinePerintah menghentikan pipa. Jika revisi sedang dijalankan melalui pipeline saat Anda menjalankan update-pipeline perintah, proses itu dihentikan. Anda harus memulai pipeline secara manual untuk menjalankan revisi itu melalui pipeline yang diperbarui. Gunakan **`start-pipeline-execution`** perintah untuk memulai pipeline Anda secara manual.

Buat EventBridge aturan untuk CodeCommit sumber (CloudFormation template)

Untuk digunakan CloudFormation untuk membuat aturan, perbarui template Anda seperti yang ditunjukkan di sini.

Untuk memperbarui CloudFormation template pipeline Anda dan membuat EventBridge aturan

1. Di template, di bawah`Resources`, gunakan `AWS::IAM::Role` CloudFormation sumber daya untuk mengonfigurasi peran IAM yang memungkinkan acara Anda memulai pipeline. Entri ini membuat peran yang menggunakan dua kebijakan:

   - Kebijakan pertama memungkinkan peran diasumsikan.
   - Kebijakan kedua memberikan izin untuk memulai pipeline.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::IAM::Role` sumber daya memungkinkan CloudFormation untuk membuat izin untuk EventBridge. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. Dalam template, di bawah`Resources`, gunakan `AWS::Events::Rule` CloudFormation sumber daya untuk menambahkan EventBridge aturan. Pola acara ini membuat acara yang memantau perubahan push ke repositori Anda. Saat EventBridge mendeteksi perubahan status repositori, aturan akan muncul di pipeline target Anda`StartPipelineExecution`.

   Mengapa saya membuat perubahan ini? Menambahkan `AWS::Events::Rule` sumber daya memungkinkan CloudFormation untuk membuat acara. Sumber daya ini ditambahkan ke CloudFormation tumpukan Anda.

   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. (Opsional) Untuk mengonfigurasi transformator input dengan penggantian sumber untuk ID gambar tertentu, gunakan cuplikan YAMAL berikut. Contoh berikut mengonfigurasi penggantian di mana:

   - `Source`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `actionName`
   - `COMMIT_ID`Dalam contoh ini, adalah nilai dinamis, yang didefinisikan pada pembuatan pipeline, bukan berasal dari peristiwa sumber. `revisionType`
   - The`revisionValue`, **<** *`revisionValue`* **>** dalam contoh ini, berasal dari variabel peristiwa sumber.
   - Variabel output untuk `BranchName` dan `Value` ditentukan.

```
Rule: my-rule
Targets:
- Id: MyTargetId
  Arn: pipeline-ARN
  InputTransformer:
    sourceRevisions:
      actionName: Source
      revisionType: COMMIT_ID
      revisionValue: <revisionValue>
    variables:
    - name: BranchName
      value: value
```

4. Simpan template yang diperbarui ke komputer lokal Anda, lalu buka CloudFormation konsol.
5. Pilih tumpukan Anda, lalu pilih Create Change Set for Current Stack.

6. Unggah template, lalu lihat perubahan yang tercantum di dalamnya CloudFormation. Ini adalah perubahan yang harus dilakukan pada tumpukan. Anda harus melihat sumber daya baru Anda dalam daftar.
7. Pilih Eksekusi.

Untuk mengedit PollForSourceChanges parameter pipeline

> **Important**
>
> Dalam banyak kasus, `PollForSourceChanges` parameter default ke true saat Anda membuat pipeline. Saat Anda menambahkan deteksi perubahan berbasis peristiwa, Anda harus menambahkan parameter ke output Anda dan mengaturnya ke false untuk menonaktifkan polling. Jika tidak, pipeline Anda dimulai dua kali untuk satu perubahan sumber. Lihat perinciannya di Pengaturan yang valid untuk `PollForSourceChanges` parameter.

- Dalam template, ubah `PollForSourceChanges` ke`false`. Jika Anda tidak menyertakan `PollForSourceChanges` dalam definisi pipeline Anda, tambahkan dan setel ke`false`.

  Mengapa saya membuat perubahan ini? Mengubah parameter ini untuk `false` mematikan pemeriksaan berkala sehingga Anda hanya dapat menggunakan deteksi perubahan berbasis peristiwa.

  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
    }
  ]
},
```

Tambahkan penyedia sumber pihak ketiga ke saluran pipa menggunakan CodeConnections

Anda dapat menggunakan AWS CodePipeline konsol atau AWS CLI untuk menghubungkan pipeline Anda ke repositori pihak ketiga.

> **Note**
>
> Saat Anda menggunakan konsol untuk membuat atau mengedit pipeline, sumber daya deteksi perubahan akan dibuat untuk Anda. Jika Anda menggunakan AWS CLI untuk membuat pipeline, Anda harus membuat sumber daya tambahan sendiri. Lihat informasi yang lebih lengkap di CodeCommit tindakan sumber dan EventBridge.

Topik

- Koneksi Azure DevOps
- Koneksi Bitbucket Cloud
- GitHub koneksi
- GitHub Koneksi Enterprise Server
- GitLabkoneksi .com
- Koneksi untuk dikelola GitLab sendiri
- Menggunakan koneksi yang dibagikan dengan akun lain

Koneksi Azure DevOps

Koneksi memungkinkan Anda untuk mengotorisasi dan membuat konfigurasi yang mengaitkan penyedia pihak ketiga Anda dengan sumber daya Anda AWS . Untuk mengaitkan repositori pihak ketiga Anda sebagai sumber untuk pipeline Anda, Anda menggunakan koneksi.

> **Note**
>
> Alih-alih membuat atau menggunakan koneksi yang ada di akun Anda, Anda dapat menggunakan koneksi bersama antara yang lain Akun AWS. Lihat Menggunakan koneksi yang dibagikan dengan akun lain.

> **Note**
>
> Fitur ini tidak tersedia di Asia Pasifik (Hong Kong), Asia Pasifik (Hyderabad), Asia Pasifik (Jakarta), Asia Pasifik (Melbourne), Asia Pasifik (Osaka), Afrika (Cape Town), Timur Tengah (Bahrain), Timur Tengah (UEA), Eropa (Spanyol), Eropa (Zurich), Israel (Tel Aviv), atau (AS-Barat) Wilayah. AWS GovCloud Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline. Untuk pertimbangan dengan tindakan ini di Wilayah Eropa (Milan), lihat catatan diCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Untuk menambahkan aksi DevOps sumber Azure CodePipeline, Anda dapat memilih salah satu untuk:

- Gunakan wizard Buat pipeline CodePipeline konsol atau halaman tindakan Edit untuk memilih opsi DevOps penyedia Azure. Lihat Buat koneksi ke Bitbucket Cloud (konsol) untuk menambahkan tindakan. Konsol membantu Anda membuat sumber daya koneksi.
- Gunakan CLI untuk menambahkan konfigurasi tindakan untuk `CreateSourceConnection` tindakan dengan `Azure DevOps` penyedia sebagai berikut:
  - Untuk membuat sumber daya koneksi, lihat Buat koneksi ke Azure DevOps (CLI) untuk membuat sumber daya koneksi dengan CLI.
  - Gunakan `CreateSourceConnection` contoh konfigurasi tindakan CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri untuk menambahkan tindakan Anda seperti yang ditunjukkan padaBuat pipeline (CLI).

> **Note**
>
> Anda juga dapat membuat koneksi menggunakan konsol Alat Pengembang di bawah Pengaturan. Lihat Membuat Koneksi.

Sebelum Anda memulai:

- Anda harus telah membuat akun dengan penyedia repositori pihak ketiga, seperti Azure DevOps

- Anda harus sudah membuat repositori kode pihak ketiga, seperti repositori DevOps Azure.

> **Note**
>
> DevOps Koneksi Azure hanya menyediakan akses ke repositori yang dimiliki oleh DevOps akun Azure yang digunakan untuk membuat koneksi.
> Untuk menginstal koneksi, Anda harus memiliki izin Kelola untuk akun Azure.

Topik

- Buat koneksi ke Azure DevOps (konsol)
- Buat koneksi ke Azure DevOps (CLI)

Buat koneksi ke Azure DevOps (konsol)

Gunakan langkah-langkah ini untuk menggunakan CodePipeline konsol guna menambahkan tindakan koneksi untuk DevOps repositori Azure Anda.

Langkah 1: Buat atau edit pipeline Anda

Untuk membuat atau mengedit pipeline Anda

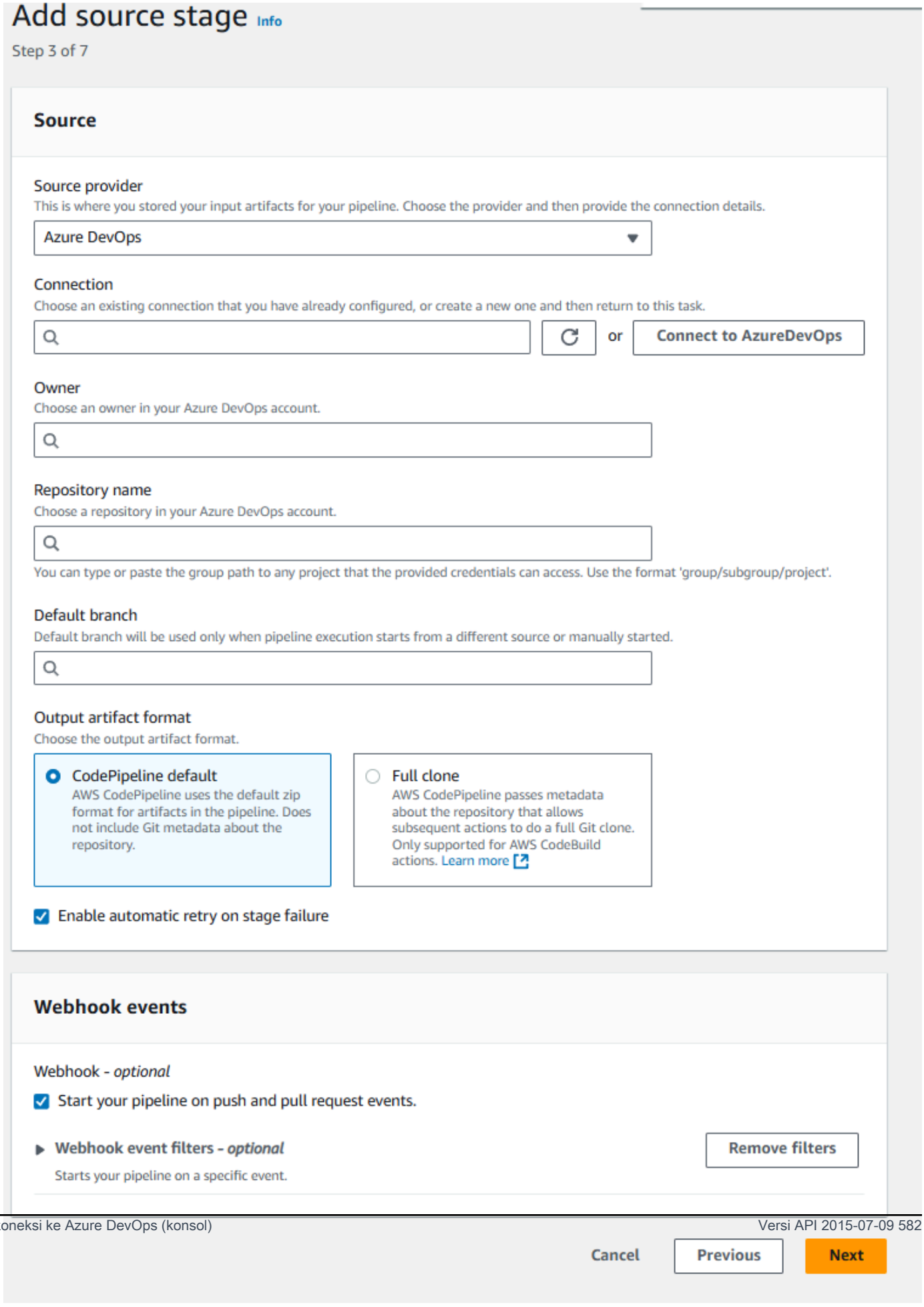
1. Masuk ke CodePipeline konsol.
2. Pilih salah satu dari berikut ini.
   - Pilih untuk membuat pipeline. Ikuti langkah-langkah di Create a Pipeline untuk menyelesaikan layar pertama dan pilih Berikutnya. Pada halaman Sumber, di bawah Penyedia Sumber, pilih Azure DevOps.
   - Pilih untuk mengedit pipeline yang ada. Pilih Edit, lalu pilih Edit tahap. Pilih untuk menambahkan atau mengedit tindakan sumber Anda. Pada halaman Edit tindakan, di bawah Nama tindakan, masukkan nama untuk tindakan Anda. Di penyedia Action, pilih Azure DevOps.
3. Lakukan salah satu tindakan berikut:
   - Di bawah Koneksi, jika Anda belum membuat koneksi ke penyedia Anda, pilih Connect to Azure DevOps. Lanjutkan ke Langkah 2: Buat Koneksi ke Azure DevOps.

- Di bawah Koneksi, jika Anda telah membuat koneksi ke penyedia Anda, pilih koneksi. Lanjutkan ke Langkah 3: Simpan Tindakan Sumber untuk Koneksi Anda.

Langkah 2: Buat koneksi ke Azure DevOps

Untuk membuat koneksi ke Azure DevOps

1. Untuk membuat sambungan ke DevOps repositori Azure, di bawah Pilih penyedia, pilih Azure. DevOps Di Nama koneksi, masukkan nama untuk koneksi yang ingin Anda buat. Pilih Connect to Azure DevOps, dan lanjutkan ke Langkah 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. Pada halaman DevOps Pengaturan Connect to Azure, nama koneksi Anda akan ditampilkan.

   Di bawah DevOps aplikasi Azure, pilih penginstalan aplikasi atau pilih Instal aplikasi baru untuk membuatnya.

   > **Note**
   >
   > Anda hanya menginstal aplikasi sekali untuk setiap DevOps akun Azure. Jika Anda telah menginstal aplikasi konektor, pilih dan pindah ke langkah terakhir di bagian ini.

3. Jika halaman login untuk Microsoft ditampilkan, masuk dengan kredensional Anda dan kemudian pilih untuk melanjutkan.
4. Di halaman penginstalan aplikasi, pesan menunjukkan bahwa aplikasi konektor mencoba terhubung ke DevOps akun Azure Anda.

   Pilih Berikan akses.
5. Pada halaman koneksi, ID koneksi untuk instalasi baru Anda ditampilkan. Pilih Hubungkan. Koneksi yang dibuat ditampilkan dalam daftar koneksi.

Langkah 3: Simpan aksi DevOps sumber Azure Anda

Gunakan langkah-langkah ini di halaman wizard atau Edit tindakan untuk menyimpan tindakan sumber Anda dengan informasi koneksi Anda.

Untuk menyelesaikan dan menyimpan tindakan sumber Anda dengan koneksi Anda

1. Dalam nama Repositori, pilih nama repositori pihak ketiga Anda.
2. Di bawah pemicu Pipeline, Anda dapat menambahkan pemicu jika tindakan Anda adalah anCodeConnections tindakan. Untuk mengonfigurasi konfigurasi pemicu pipeline dan memfilter secara opsional dengan pemicu, lihat detail selengkapnya di. Tambahkan pemicu dengan tipe acara permintaan push atau pull kode
3. Dalam format artefak Output, Anda harus memilih format untuk artefak Anda.

   - Untuk menyimpan artefak keluaran dari DevOps tindakan Azure menggunakan metode default, pilih CodePipeline default. Tindakan mengakses file dari repositori dan menyimpan artefak dalam file ZIP di toko artefak pipa.

- Untuk menyimpan file JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung, pilih klon Penuh. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.

  Jika Anda memilih opsi ini, Anda perlu memperbarui izin untuk peran layanan CodeBuild proyek Anda seperti yang ditunjukkan pada[Tambahkan CodeBuild GitClone izin untuk koneksi ke Bitbucket,, GitHub Enterprise Server GitHub, atau .com GitLab](#).

4. Pilih Berikutnya di wizard atau Simpan di halaman Edit tindakan.

Buat koneksi ke Azure DevOps (CLI)

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk membuat koneksi.

Untuk melakukannya, gunakan perintah create-connection.

> **Important**
>
> Koneksi yang dibuat melalui AWS CLI atau AWS CloudFormation dalam `PENDING` status secara default. Setelah Anda membuat koneksi dengan CLI atau CloudFormation, gunakan konsol untuk mengedit koneksi untuk membuat statusnya. `AVAILABLE`

Untuk membuat koneksi

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows). Gunakan AWS CLI untuk menjalankan create-connection perintah, menentukan `--provider-type` dan `--connection-name` untuk koneksi Anda. Dalam contoh ini, nama penyedia pihak ketiga adalah `AzureDevOps` dan nama koneksi yang ditentukan adalah `MyConnection`.

   ```
   aws codeconnections create-connection --provider-type AzureDevOps --connection-name
    MyConnection
   ```

   Jika berhasil, perintah ini mengembalikan informasi ARN koneksi seperti berikut ini.

   ```
   {
       "ConnectionArn": "arn:aws:codeconnections:us-west-2:account_id:connection/
   aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
   }
   ```

2. Gunakan konsol untuk menyelesaikan koneksi. Untuk informasi selengkapnya, lihat Memperbarui sambungan yang tertunda.
3. Pipeline default untuk mendeteksi perubahan pada push kode ke repositori sumber koneksi. Untuk mengonfigurasi konfigurasi pemicu pipeline untuk rilis manual atau untuk tag Git, lakukan salah satu hal berikut:

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memulai dengan rilis manual saja, tambahkan baris berikut ke konfigurasi:

     ```
     "DetectChanges": "false",
     ```

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memfilter dengan pemicu, lihat detail selengkapnya diTambahkan pemicu dengan tipe acara permintaan push atau pull kode. Misalnya, berikut ini menambahkan tag Git ke level pipeline definisi JSON pipeline. Dalam contoh ini, `release-v0` dan `release-v1` merupakan tag Git untuk disertakan, dan `release-v2` merupakan tag Git untuk dikecualikan.

     ```
     "triggers": [
                 {
                     "providerType": "CodeStarSourceConnection",
                     "gitConfiguration": {
                         "sourceActionName": "Source",
                         "push": [
                             {
                                 "tags": {
                                     "includes": [
                                         "release-v0", "release-v1"
                                     ],
                                     "excludes": [
                                         "release-v2"
                                     ]
                                 }
                             }
                         ]
                     }
                 }
             ]
     ```

Koneksi Bitbucket Cloud

Koneksi memungkinkan Anda untuk mengotorisasi dan membuat konfigurasi yang mengaitkan penyedia pihak ketiga Anda dengan sumber daya Anda AWS . Untuk mengaitkan repositori pihak ketiga Anda sebagai sumber untuk pipeline Anda, Anda menggunakan koneksi.

> **Note**
>
> Alih-alih membuat atau menggunakan koneksi yang ada di akun Anda, Anda dapat menggunakan koneksi bersama antara yang lain Akun AWS. Lihat Menggunakan koneksi yang dibagikan dengan akun lain.

> **Note**
>
> Fitur ini tidak tersedia di Asia Pasifik (Hong Kong), Asia Pasifik (Hyderabad), Asia Pasifik (Jakarta), Asia Pasifik (Melbourne), Asia Pasifik (Osaka), Afrika (Cape Town), Timur Tengah (Bahrain), Timur Tengah (UEA), Eropa (Spanyol), Eropa (Zurich), Israel (Tel Aviv), atau (AS-Barat) Wilayah. AWS GovCloud Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline. Untuk pertimbangan dengan tindakan ini di Wilayah Eropa (Milan), lihat catatan diCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Untuk menambahkan aksi sumber Bitbucket Cloud CodePipeline, Anda dapat memilih salah satu untuk:

- Gunakan wizard Buat pipeline CodePipeline konsol atau halaman tindakan Edit untuk memilih opsi penyedia Bitbucket. Lihat Buat koneksi ke Bitbucket Cloud (konsol) untuk menambahkan tindakan. Konsol membantu Anda membuat sumber daya koneksi.

  > **Note**
  >
  > Anda dapat membuat koneksi ke repositori Bitbucket Cloud. Jenis penyedia Bitbucket yang diinstal, seperti Bitbucket Server, tidak didukung.

- Gunakan CLI untuk menambahkan konfigurasi tindakan untuk `CreateSourceConnection` tindakan dengan `Bitbucket` penyedia sebagai berikut:
  - Untuk membuat sumber daya koneksi, lihat Buat koneksi ke Bitbucket Cloud (CLI) untuk membuat sumber daya koneksi dengan CLI.
  - Gunakan `CreateSourceConnection` contoh konfigurasi tindakan CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri untuk menambahkan tindakan Anda seperti yang ditunjukkan padaBuat pipeline (CLI).

> **Note**
>
> Anda juga dapat membuat koneksi menggunakan konsol Alat Pengembang di bawah Pengaturan. Lihat Membuat Koneksi.

Sebelum Anda memulai:

- Anda harus telah membuat akun dengan penyedia repositori pihak ketiga, seperti Bitbucket Cloud.
- Anda harus sudah membuat repositori kode pihak ketiga, seperti repositori Bitbucket Cloud.

> **Note**
>
> Koneksi Bitbucket Cloud hanya menyediakan akses ke repositori yang dimiliki oleh akun Bitbucket Cloud yang digunakan untuk membuat koneksi.
> Jika aplikasi sedang diinstal di ruang kerja Bitbucket Cloud, Anda perlu Mengelola izin ruang kerja. Jika tidak, opsi untuk menginstal aplikasi tidak akan ditampilkan.

Topik

- Buat koneksi ke Bitbucket Cloud (konsol)
- Buat koneksi ke Bitbucket Cloud (CLI)

Buat koneksi ke Bitbucket Cloud (konsol)

Gunakan langkah-langkah ini untuk menggunakan CodePipeline konsol untuk menambahkan tindakan koneksi untuk repositori Bitbucket Anda.

> **Note**
>
> Anda dapat membuat koneksi ke repositori Bitbucket Cloud. Jenis penyedia Bitbucket yang diinstal, seperti Bitbucket Server, tidak didukung.

Langkah 1: Buat atau edit pipeline Anda

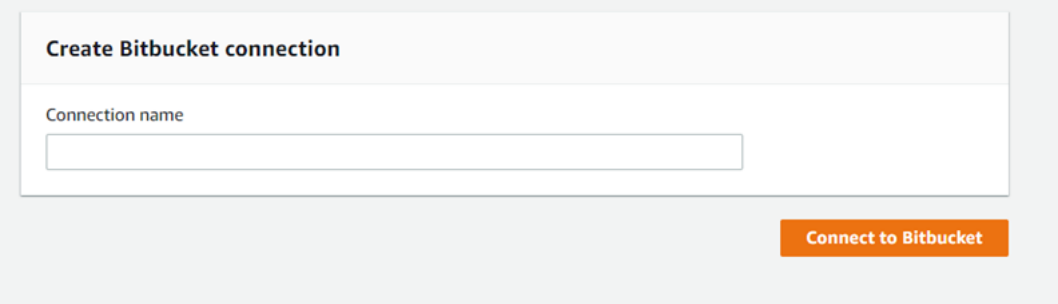
Untuk membuat atau mengedit pipeline Anda

1. Masuk ke CodePipeline konsol.
2. Pilih salah satu dari berikut ini.
   - Pilih untuk membuat pipeline. Ikuti langkah-langkah di Create a Pipeline untuk menyelesaikan layar pertama dan pilih Berikutnya. Pada halaman Sumber, di bawah Penyedia Sumber, pilih Bitbucket.
   - Pilih untuk mengedit pipeline yang ada. Pilih Edit, lalu pilih Edit tahap. Pilih untuk menambahkan atau mengedit tindakan sumber Anda. Pada halaman Edit tindakan, di bawah Nama tindakan, masukkan nama untuk tindakan Anda. Di penyedia Action, pilih Bitbucket.
3. Lakukan salah satu tindakan berikut:
   - Di bawah Koneksi, jika Anda belum membuat koneksi ke penyedia Anda, pilih Connect to Bitbucket. Lanjutkan ke Langkah 2: Buat Koneksi ke Bitbucket.
   - Di bawah Koneksi, jika Anda telah membuat koneksi ke penyedia Anda, pilih koneksi. Lanjutkan ke Langkah 3: Simpan Tindakan Sumber untuk Koneksi Anda.

Langkah 2: Buat koneksi ke Bitbucket Cloud

Untuk membuat koneksi ke Bitbucket Cloud

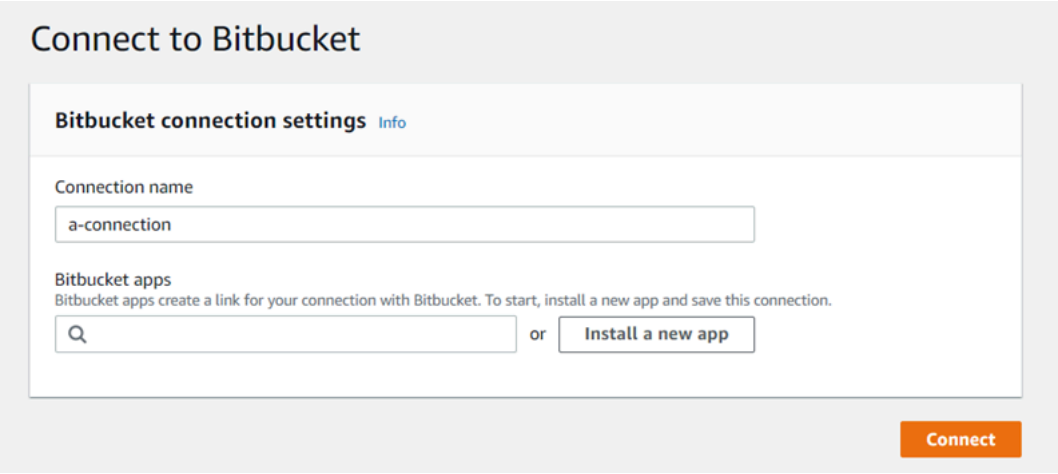
1. Pada halaman Pengaturan Connect to Bitbucket, masukkan nama koneksi Anda dan pilih Connect to Bitbucket.



Bidang aplikasi Bitbucket muncul.

2. Pada Aplikasi Bitbucket, pilih penginstalan aplikasi atau pilih Instal aplikasi baru untuk membuatnya.

> **Note**
>
> Anda hanya menginstal aplikasi sekali untuk setiap ruang kerja atau akun Bitbucket Cloud. Jika Anda telah menginstal aplikasi Bitbucket, pilih dan pindah ke langkah 4.

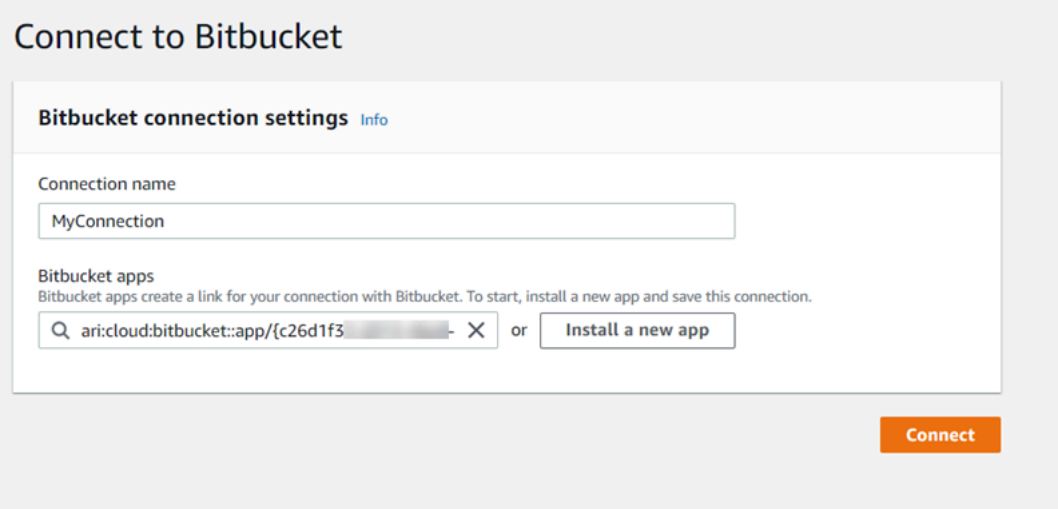


3. Jika halaman login untuk Bitbucket Cloud ditampilkan, masuk dengan kredensional Anda dan kemudian pilih untuk melanjutkan.
4. Pada halaman instalasi aplikasi, sebuah pesan menunjukkan bahwa AWS CodeStar aplikasi sedang mencoba untuk terhubung ke akun Bitbucket Anda.

   Jika Anda menggunakan ruang kerja Bitbucket, ubah opsi Otorisasi untuk ke ruang kerja. Hanya ruang kerja di mana Anda memiliki akses administrator yang akan ditampilkan.

   Pilih Berikan akses.

5. Di Aplikasi Bitbucket, ID koneksi untuk instalasi baru Anda ditampilkan. Pilih Hubungkan. Koneksi yang dibuat ditampilkan dalam daftar koneksi.



Langkah 3: Simpan aksi sumber Bitbucket Cloud Anda

Gunakan langkah-langkah ini di halaman wizard atau Edit tindakan untuk menyimpan tindakan sumber Anda dengan informasi koneksi Anda.

Untuk menyelesaikan dan menyimpan tindakan sumber Anda dengan koneksi Anda

1. Dalam nama Repositori, pilih nama repositori pihak ketiga Anda.
2. Di bawah pemicu Pipeline, Anda dapat menambahkan pemicu jika tindakan Anda adalah CodeConnections tindakan. Untuk mengonfigurasi konfigurasi pemicu pipeline dan memfilter secara opsional dengan pemicu, lihat detail selengkapnya di. Tambahkan pemicu dengan tipe acara permintaan push atau pull kode
3. Dalam format artefak Output, Anda harus memilih format untuk artefak Anda.

   - Untuk menyimpan artefak keluaran dari tindakan Bitbucket Cloud menggunakan metode default, pilih CodePipeline default. Tindakan mengakses file dari repositori Bitbucket Cloud dan menyimpan artefak dalam file ZIP di toko artefak pipeline.
   - Untuk menyimpan file JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung, pilih klon Penuh. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.

Jika Anda memilih opsi ini, Anda perlu memperbarui izin untuk peran layanan CodeBuild proyek Anda seperti yang ditunjukkan pada[Tambahkan CodeBuild GitClone izin untuk koneksi ke Bitbucket,, GitHub Enterprise Server GitHub, atau .com GitLab](#).

4. Pilih Berikutnya di wizard atau Simpan di halaman Edit tindakan.

Buat koneksi ke Bitbucket Cloud (CLI)

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk membuat koneksi.

> **Note**
>
> Anda dapat membuat koneksi ke repositori Bitbucket Cloud. Jenis penyedia Bitbucket yang diinstal, seperti Bitbucket Server, tidak didukung.

Untuk melakukannya, gunakan perintah create-connection.

> **Important**
>
> Koneksi yang dibuat melalui AWS CLI atau AWS CloudFormation dalam `PENDING` status secara default. Setelah Anda membuat koneksi dengan CLI atau CloudFormation, gunakan konsol untuk mengedit koneksi untuk membuat statusnya. `AVAILABLE`

Untuk membuat koneksi

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows). Gunakan AWS CLI untuk menjalankan create-connection perintah, menentukan `--provider-type` dan `--connection-name` untuk koneksi Anda. Dalam contoh ini, nama penyedia pihak ketiga adalah `Bitbucket` dan nama koneksi yang ditentukan adalah `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type Bitbucket --connection-name MyConnection
   ```

   Jika berhasil, perintah ini mengembalikan informasi ARN koneksi seperti berikut ini.

   ```
   {
   ```

```
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Gunakan konsol untuk menyelesaikan koneksi. Untuk informasi selengkapnya, lihat Memperbarui sambungan yang tertunda.
3. Pipeline default untuk mendeteksi perubahan pada push kode ke repositori sumber koneksi. Untuk mengonfigurasi konfigurasi pemicu pipeline untuk rilis manual atau untuk tag Git, lakukan salah satu hal berikut:

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memulai dengan rilis manual saja, tambahkan baris berikut ke konfigurasi:

     ```
     "DetectChanges": "false",
     ```

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memfilter dengan pemicu, lihat detail selengkapnya diTambahkan pemicu dengan tipe acara permintaan push atau pull kode. Misalnya, berikut ini menambahkan tag Git ke level pipeline definisi JSON pipeline. Dalam contoh ini, `release-v0` dan `release-v1` merupakan tag Git untuk disertakan, dan `release-v2` merupakan tag Git untuk dikecualikan.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]
     ```

GitHub koneksi

Anda menggunakan koneksi untuk mengotorisasi dan membuat konfigurasi yang mengaitkan penyedia pihak ketiga Anda dengan sumber daya Anda AWS .

> **Note**
>
> Alih-alih membuat atau menggunakan koneksi yang ada di akun Anda, Anda dapat menggunakan koneksi bersama antara yang lain Akun AWS. Lihat Menggunakan koneksi yang dibagikan dengan akun lain.

> **Note**
>
> Fitur ini tidak tersedia di Asia Pasifik (Hong Kong), Asia Pasifik (Hyderabad), Asia Pasifik (Jakarta), Asia Pasifik (Melbourne), Asia Pasifik (Osaka), Afrika (Cape Town), Timur Tengah (Bahrain), Timur Tengah (UEA), Eropa (Spanyol), Eropa (Zurich), Israel (Tel Aviv), atau (AS-Barat) Wilayah. AWS GovCloud Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline. Untuk pertimbangan dengan tindakan ini di Wilayah Eropa (Milan), lihat catatan diCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Untuk menambahkan tindakan sumber untuk repositori GitHub Enterprise Cloud Anda GitHub atau di CodePipeline, Anda dapat memilih salah satu untuk:

- Gunakan wizard Buat pipeline CodePipeline konsol atau halaman tindakan Edit untuk memilih opsi penyedia GitHub (melalui GitHub Aplikasi). Lihat Buat koneksi ke GitHub Enterprise Server (konsol) untuk menambahkan tindakan. Konsol membantu Anda membuat sumber daya koneksi.

  > **Note**
  >
  > Untuk tutorial yang memandu Anda melalui cara menambahkan GitHub koneksi dan menggunakan opsi klon penuh di pipeline Anda untuk mengkloning metadata, lihat. Tutorial: Gunakan klon lengkap dengan sumber GitHub pipa

- Gunakan CLI untuk menambahkan konfigurasi tindakan untuk tindakan dengan `GitHub` penyedia dengan langkah-langkah CLI yang ditunjukkan di. `CodeStarSourceConnection` Buat pipeline (CLI)

> **Note**
>
> Anda juga dapat membuat koneksi menggunakan konsol Alat Pengembang di bawah Pengaturan. Lihat Membuat Koneksi.

Sebelum Anda memulai:

- Anda harus telah membuat akun dengan GitHub.
- Anda harus sudah membuat repositori GitHub kode.
- Jika peran CodePipeline layanan Anda dibuat sebelum 18 Desember 2019, Anda mungkin perlu memperbarui izinnya untuk digunakan `codestar-connections:UseConnection` untuk AWS CodeStar koneksi. Untuk petunjuk, lihat Menambahkan izin ke peran CodePipeline layanan.

> **Note**
>
> Untuk membuat koneksi, Anda harus menjadi pemilik GitHub organisasi. Untuk repositori yang tidak berada di bawah organisasi, Anda harus menjadi pemilik repositori.

Topik

- Buat koneksi ke GitHub (konsol)
- Buat koneksi ke GitHub (CLI)

Buat koneksi ke GitHub (konsol)

Gunakan langkah-langkah ini untuk menggunakan CodePipeline konsol guna menambahkan tindakan koneksi untuk repositori Cloud GitHub atau GitHub Enterprise Cloud Anda.

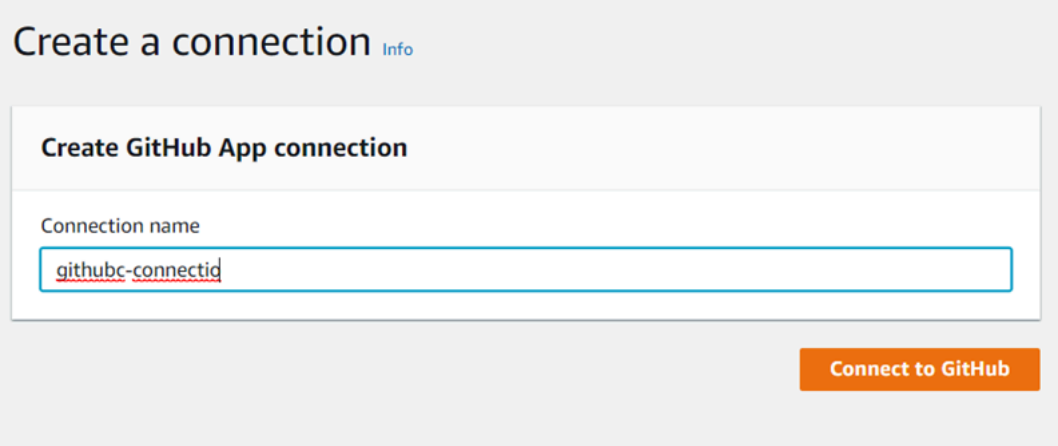
> **Note**
>
> Dalam langkah-langkah ini, Anda dapat memilih repositori tertentu di bawah Akses Repositori. Repositori apa pun yang tidak dipilih tidak akan dapat diakses atau dilihat oleh. CodePipeline

Langkah 1: Buat atau edit pipeline Anda

1. Masuk ke CodePipeline konsol.
2. Pilih salah satu dari berikut ini.
   - Pilih untuk membuat pipeline. Ikuti langkah-langkah di Create a Pipeline untuk menyelesaikan layar pertama dan pilih Berikutnya. Pada halaman Sumber, di bawah Penyedia Sumber, pilih GitHub (melalui GitHub Aplikasi).
   - Pilih untuk mengedit pipeline yang ada. Pilih Edit, lalu pilih Edit tahap. Pilih untuk menambahkan atau mengedit tindakan sumber Anda. Pada halaman Edit tindakan, di bawah Nama tindakan, masukkan nama untuk tindakan Anda. Di penyedia Tindakan, pilih GitHub (melalui GitHub Aplikasi).
3. Lakukan salah satu tindakan berikut:
   - Di bawah Koneksi, jika Anda belum membuat sambungan ke penyedia Anda, pilih Connect to GitHub. Lanjutkan ke Langkah 2: Buat Koneksi ke GitHub.
   - Di bawah Koneksi, jika Anda telah membuat koneksi ke penyedia Anda, pilih koneksi. Lanjutkan ke Langkah 3: Simpan tindakan sumber untuk koneksi Anda.

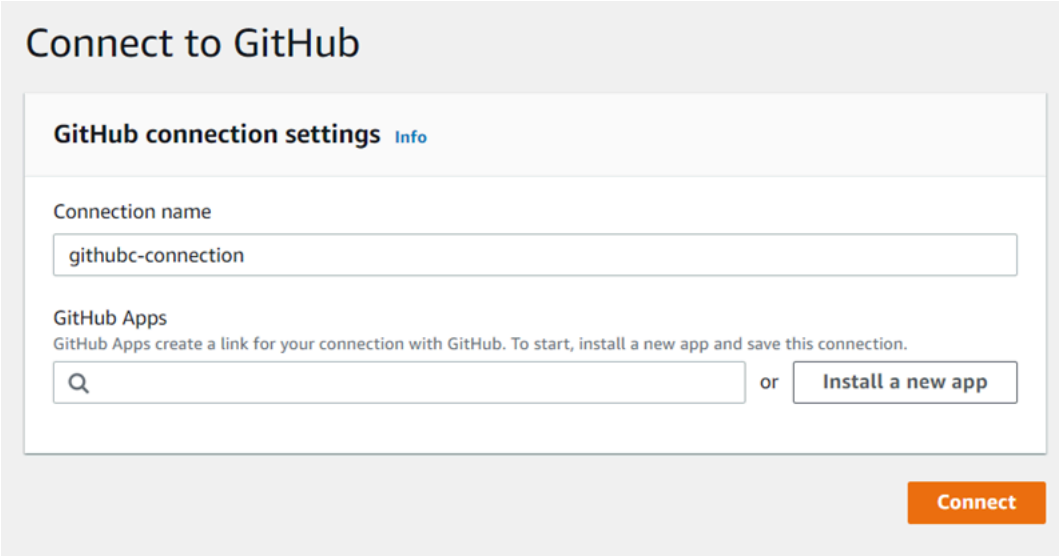
Langkah 2: Buat koneksi ke GitHub

Setelah Anda memilih untuk membuat koneksi, GitHub halaman Connect to akan muncul.



Untuk membuat koneksi ke GitHub

1. Di bawah pengaturan GitHub koneksi, nama koneksi Anda muncul di Nama koneksi. Pilih Connect to GitHub. Halaman permintaan akses muncul.
2. Pilih Otorisasi AWS Konektor untuk GitHub. Halaman koneksi menampilkan dan menampilkan bidang GitHub Aplikasi.



3. Di bawah GitHub Aplikasi, pilih penginstalan aplikasi atau pilih Instal aplikasi baru untuk membuatnya.

   Anda menginstal satu aplikasi untuk semua koneksi Anda ke penyedia tertentu. Jika Anda telah menginstal AWS Connector for GitHub app, pilih dan lewati langkah ini.

> **Note**
>
> Jika Anda ingin membuat token akses pengguna, pastikan Anda telah menginstal AWS Connector for GitHub app lalu biarkan bidang Instalasi aplikasi kosong. CodeConnections akan menggunakan token akses pengguna untuk koneksi.

4. Pada GitHub halaman Install AWS Connector for, pilih akun tempat Anda ingin menginstal aplikasi.

> **Note**
>
> Anda hanya menginstal aplikasi sekali untuk setiap GitHub akun. Jika sebelumnya Anda menginstal aplikasi, Anda dapat memilih Konfigurasi untuk melanjutkan ke halaman modifikasi untuk instalasi aplikasi Anda, atau Anda dapat menggunakan tombol kembali untuk kembali ke konsol.

5. Pada GitHub halaman Install AWS Connector for, tinggalkan default, dan pilih Install.
6. Pada GitHub halaman Connect to, ID koneksi untuk instalasi baru Anda muncul di GitHub Aplikasi. Pilih Hubungkan.

Langkah 3: Simpan tindakan GitHub sumber Anda

Gunakan langkah-langkah ini di halaman Edit tindakan untuk menyimpan tindakan sumber Anda dengan informasi koneksi Anda.

Untuk menyimpan tindakan GitHub sumber Anda

1. Dalam nama Repositori, pilih nama repositori pihak ketiga Anda.
2. Di bawah pemicu Pipeline, Anda dapat menambahkan pemicu jika tindakan Anda adalah CodeConnections tindakan. Untuk mengonfigurasi konfigurasi pemicu pipeline dan memfilter secara opsional dengan pemicu, lihat detail selengkapnya di. Tambahkan pemicu dengan tipe acara permintaan push atau pull kode
3. Dalam format artefak Output, Anda harus memilih format untuk artefak Anda.

   - Untuk menyimpan artefak keluaran dari GitHub tindakan menggunakan metode default, pilih CodePipeline default. Tindakan mengakses file dari GitHub repositori dan menyimpan artefak dalam file ZIP di toko artefak pipa.

- Untuk menyimpan file JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung, pilih klon Penuh. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.

  Jika Anda memilih opsi ini, Anda perlu memperbarui izin untuk peran layanan CodeBuild proyek Anda seperti yang ditunjukkan padaTambahkan CodeBuild GitClone izin untuk koneksi ke Bitbucket,, GitHub Enterprise Server GitHub, atau .com GitLab. Untuk tutorial yang menunjukkan cara menggunakan opsi klon Penuh, lihatTutorial: Gunakan klon lengkap dengan sumber GitHub pipa.

4. Pilih Berikutnya di wizard atau Simpan di halaman Edit tindakan.

Buat koneksi ke GitHub (CLI)

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk membuat koneksi.

Untuk melakukannya, gunakan perintah create-connection.

> **Important**
>
> Koneksi yang dibuat melalui AWS CLI atau AWS CloudFormation dalam `PENDING` status secara default. Setelah Anda membuat koneksi dengan CLI atau CloudFormation, gunakan konsol untuk mengedit koneksi untuk membuat statusnya. `AVAILABLE`

Untuk membuat koneksi

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows). Gunakan AWS CLI untuk menjalankan create-connection perintah, menentukan `--provider-type` dan `--connection-name` untuk koneksi Anda. Dalam contoh ini, nama penyedia pihak ketiga adalah `GitHub` dan nama koneksi yang ditentukan adalah `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type GitHub --connection-name
    MyConnection
   ```

   Jika berhasil, perintah ini mengembalikan informasi ARN koneksi seperti berikut ini.

   ```
   {
   ```

```
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Gunakan konsol untuk menyelesaikan koneksi. Untuk informasi selengkapnya, lihat Memperbarui sambungan yang tertunda.

3. Pipeline default untuk mendeteksi perubahan pada push kode ke repositori sumber koneksi. Untuk mengonfigurasi konfigurasi pemicu pipeline untuk rilis manual atau untuk tag Git, lakukan salah satu hal berikut:

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memulai dengan rilis manual saja, tambahkan baris berikut ke konfigurasi:

     ```
     "DetectChanges": "false",
     ```

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memfilter dengan pemicu, lihat detail selengkapnya diTambahkan pemicu dengan tipe acara permintaan push atau pull kode. Misalnya, berikut ini menambah tingkat pipa definisi JSON pipa. Dalam contoh ini, `release-v0` dan `release-v1` merupakan tag Git untuk disertakan, dan `release-v2` merupakan tag Git untuk dikecualikan.

     ```
     "triggers": [
                 {
                     "providerType": "CodeStarSourceConnection",
                     "gitConfiguration": {
                         "sourceActionName": "Source",
                         "push": [
                             {
                                 "tags": {
                                     "includes": [
                                         "release-v0", "release-v1"
                                     ],
                                     "excludes": [
                                         "release-v2"
                                     ]
                                 }
                             }
                         ]
                     }
                 }
             ]
     ```

GitHub Koneksi Enterprise Server

Koneksi memungkinkan Anda untuk mengotorisasi dan membuat konfigurasi yang mengaitkan penyedia pihak ketiga Anda dengan sumber daya Anda AWS . Untuk mengaitkan repositori pihak ketiga Anda sebagai sumber untuk pipeline Anda, Anda menggunakan koneksi.

> **Note**
>
> Alih-alih membuat atau menggunakan koneksi yang ada di akun Anda, Anda dapat menggunakan koneksi bersama antara yang lain Akun AWS. Lihat Menggunakan koneksi yang dibagikan dengan akun lain.

> **Note**
>
> Fitur ini tidak tersedia di Asia Pasifik (Hong Kong), Asia Pasifik (Hyderabad), Asia Pasifik (Jakarta), Asia Pasifik (Melbourne), Asia Pasifik (Osaka), Afrika (Cape Town), Timur Tengah (Bahrain), Timur Tengah (UEA), Eropa (Spanyol), Eropa (Zurich), Israel (Tel Aviv), atau (AS-Barat) Wilayah. AWS GovCloud Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline. Untuk pertimbangan dengan tindakan ini di Wilayah Eropa (Milan), lihat catatan diCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Untuk menambahkan tindakan sumber Server GitHub Perusahaan CodePipeline, Anda dapat memilih salah satu untuk:

- Gunakan wizard Buat pipeline CodePipeline konsol atau halaman tindakan Edit untuk memilih opsi penyedia Server GitHub Perusahaan. Lihat Buat koneksi ke GitHub Enterprise Server (konsol) untuk menambahkan tindakan. Konsol membantu Anda membuat sumber daya host dan sumber daya koneksi.
- Gunakan CLI untuk menambahkan konfigurasi tindakan untuk `CreateSourceConnection` tindakan dengan `GitHubEnterpriseServer` penyedia dan membuat sumber daya Anda:
  - Untuk membuat sumber daya koneksi Anda, lihat Buat host dan koneksi ke GitHub Enterprise Server (CLI) untuk membuat sumber daya host dan sumber daya koneksi dengan CLI.

- Gunakan `CreateSourceConnection` contoh konfigurasi tindakan CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri untuk menambahkan tindakan Anda seperti yang ditunjukkan padaBuat pipeline (CLI).

> **Note**
>
> Anda juga dapat membuat koneksi menggunakan konsol Alat Pengembang di bawah Pengaturan. Lihat Membuat Koneksi.

Sebelum Anda memulai:

- Anda harus telah membuat akun dengan GitHub Enterprise Server dan menginstal instance GitHub Enterprise Server pada infrastruktur Anda.

  > **Note**
  >
  > Setiap VPC hanya dapat dikaitkan dengan satu host (contoh Server GitHub Perusahaan) pada satu waktu.

- Anda harus sudah membuat repositori kode dengan GitHub Enterprise Server.

Topik

- Buat koneksi ke GitHub Enterprise Server (konsol)
- Buat host dan koneksi ke GitHub Enterprise Server (CLI)

Buat koneksi ke GitHub Enterprise Server (konsol)

Gunakan langkah-langkah ini untuk menggunakan CodePipeline konsol untuk menambahkan tindakan koneksi untuk repositori GitHub Enterprise Server Anda.

> **Note**
>
> GitHub Koneksi Enterprise Server hanya menyediakan akses ke repositori yang dimiliki oleh akun GitHub Enterprise Server yang digunakan untuk membuat koneksi.

Sebelum Anda mulai:

Untuk koneksi host ke GitHub Enterprise Server, Anda harus telah menyelesaikan langkah-langkah untuk membuat sumber daya host untuk koneksi Anda. Lihat Mengelola host untuk koneksi.

Langkah 1: Buat atau edit pipeline Anda

Untuk membuat atau mengedit pipeline

1. Masuk ke CodePipeline konsol.
2. Pilih salah satu dari berikut ini.
   - Pilih untuk membuat pipeline. Ikuti langkah-langkah di Create a Pipeline untuk menyelesaikan layar pertama dan pilih Berikutnya. Pada halaman Sumber, di bawah Penyedia sumber, pilih Server GitHub Perusahaan.
   - Pilih untuk mengedit pipeline yang ada. Pilih Edit, lalu pilih Edit tahap. Pilih untuk menambahkan atau mengedit tindakan sumber Anda. Pada halaman Edit tindakan, di bawah Nama tindakan, masukkan nama untuk tindakan Anda. Di penyedia Tindakan, pilih Server GitHub Perusahaan.
3. Lakukan salah satu tindakan berikut:
   - Di bawah Koneksi, jika Anda belum membuat sambungan ke penyedia Anda, pilih Connect to GitHub Enterprise Server. Lanjutkan ke Langkah 2: Buat Koneksi ke Server GitHub Perusahaan.
   - Di bawah Koneksi, jika Anda telah membuat koneksi ke penyedia Anda, pilih koneksi. Lanjutkan ke Langkah 3: Simpan Tindakan Sumber untuk Koneksi Anda.

Membuat koneksi ke GitHub Enterprise Server

Setelah Anda memilih untuk membuat koneksi, halaman Connect to GitHub Enterprise Server ditampilkan.

> **Important**
>
> AWS CodeConnections tidak mendukung GitHub Enterprise Server versi 2.22.0 karena masalah yang diketahui dalam rilis. Untuk menghubungkan, tingkatkan ke versi 2.22.1 atau versi terbaru yang tersedia.

Untuk terhubung ke Server GitHub Perusahaan

1. Di Nama koneksi, masukkan nama untuk koneksi Anda.
2. Di URL, masukkan titik akhir untuk server Anda.

> **Note**
>
> Jika URL yang disediakan telah digunakan untuk menyiapkan Server GitHub Perusahaan untuk koneksi, Anda akan diminta untuk memilih ARN sumber daya host yang dibuat sebelumnya untuk titik akhir tersebut.

3. Jika Anda telah meluncurkan server Anda ke Amazon VPC dan Anda ingin terhubung dengan VPC Anda, pilih Gunakan VPC dan selesaikan yang berikut ini.

   a. Di ID VPC, pilih ID VPC anda. Pastikan untuk memilih VPC untuk infrastruktur tempat instans Server GitHub Perusahaan Anda diinstal atau VPC dengan akses ke instance Server GitHub Perusahaan Anda melalui VPN atau Direct Connect.

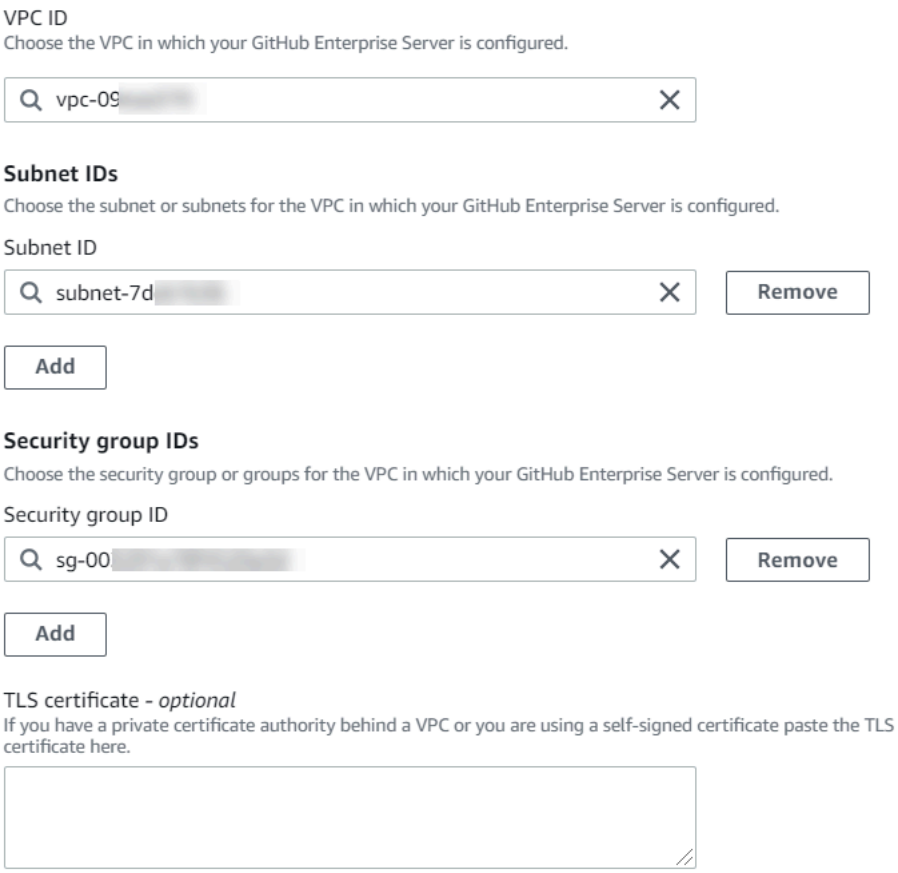
   b. Pada ID subnet, pilih Tambahkan. Di bidang, pilih ID subnet yang ingin Anda gunakan untuk host Anda. Anda dapat memilih hingga 10 subnet.

      Pastikan untuk memilih subnet untuk infrastruktur tempat instans GitHub Enterprise Server Anda diinstal atau subnet dengan akses ke instance GitHub Enterprise Server yang diinstal melalui VPN atau Direct Connect.

   c. Di bawah Grup keamanan IDs, pilih Tambah. Di bidang, pilih grup keamanan yang ingin Anda gunakan untuk host Anda. Anda dapat memilih hingga 10 grup keamanan.

      Pastikan untuk memilih grup keamanan untuk infrastruktur tempat instans Server GitHub Perusahaan Anda diinstal atau grup keamanan dengan akses ke instans Server GitHub Perusahaan yang diinstal melalui VPN atau Direct Connect.

   d. Jika Anda memiliki VPC pribadi yang dikonfigurasi, dan Anda telah mengonfigurasi instance Server GitHub Perusahaan Anda untuk melakukan validasi TLS menggunakan otoritas sertifikat non-publik, dalam sertifikat TLS, masukkan ID sertifikat Anda. Nilai Sertifikat TLS harus menjadi kunci publik sertifikat.



4. Pilih Connect to GitHub Enterprise Server. Koneksi yang dibuat ditampilkan dengan status Tertunda. Sumber daya host dibuat untuk koneksi dengan informasi server yang Anda berikan. Untuk nama host, URL digunakan.
5. Pilih Perbarui koneksi tertunda.
6. Jika diminta, pada halaman login GitHub Enterprise, masuk dengan kredensi GitHub Enterprise Anda.
7. Pada halaman Buat GitHub Aplikasi, pilih nama untuk aplikasi Anda.
8. <app-name>Pada halaman GitHub otorisasi, pilih Otorisasi.
9. Pada halaman instalasi aplikasi, sebuah pesan menunjukkan bahwa aplikasi konektor siap untuk diinstal. Jika Anda memiliki beberapa organisasi, Anda mungkin diminta untuk memilih organisasi tempat Anda ingin menginstal aplikasi.

   Pilih pengaturan repositori tempat Anda ingin menginstal aplikasi. Pilih Install.
10. Halaman koneksi yang menampilkan koneksi yang dibuat dalam status Tersedia.

Langkah 3: Simpan tindakan sumber Server GitHub Perusahaan Anda

Gunakan langkah-langkah ini di halaman wizard atau Edit tindakan untuk menyimpan tindakan sumber Anda dengan informasi koneksi Anda.

Untuk menyelesaikan dan menyimpan tindakan sumber Anda dengan koneksi Anda

1. Dalam nama Repositori, pilih nama repositori pihak ketiga Anda.
2. Di bawah pemicu Pipeline, Anda dapat menambahkan pemicu jika tindakan Anda adalah CodeConnections tindakan. Untuk mengonfigurasi konfigurasi pemicu pipeline dan memfilter secara opsional dengan pemicu, lihat detail selengkapnya di. Tambahkan pemicu dengan tipe acara permintaan push atau pull kode
3. Dalam format artefak Output, Anda harus memilih format untuk artefak Anda.
   - Untuk menyimpan artefak keluaran dari tindakan GitHub Enterprise Server menggunakan metode default, pilih CodePipelinedefault. Tindakan mengakses file dari repositori GitHub Enterprise Server dan menyimpan artefak dalam file ZIP di toko artefak pipeline.
   - Untuk menyimpan file JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung, pilih klon Penuh. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.
4. Pilih Berikutnya di wizard atau Simpan di halaman Edit tindakan.

Buat host dan koneksi ke GitHub Enterprise Server (CLI)

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk membuat koneksi.

Untuk melakukannya, gunakan perintah create-connection.

> **Important**
>
> Koneksi yang dibuat melalui AWS CLI atau AWS CloudFormation dalam `PENDING` status secara default. Setelah Anda membuat koneksi dengan CLI atau CloudFormation, gunakan konsol untuk mengedit koneksi untuk membuat statusnya. `AVAILABLE`

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk membuat host untuk koneksi terinstal.

> **Note**
>
> Anda hanya membuat host satu kali per akun GitHub Enterprise Server. Semua koneksi Anda ke akun GitHub Enterprise Server tertentu akan menggunakan host yang sama.

Anda menggunakan host untuk mewakili titik akhir infrastruktur tempat penyedia pihak ketiga Anda diinstal. Setelah Anda menyelesaikan pembuatan host dengan CLI, host dalam status Tertunda. Anda kemudian mengatur, atau mendaftarkan, host untuk memindahkannya ke status Tersedia. Setelah host tersedia, Anda menyelesaikan langkah-langkah untuk membuat koneksi.

Untuk melakukannya, gunakan perintah create-host.

> **Important**
>
> Host yang dibuat melalui status AWS CLI berada dalam `Pending` status secara default. Setelah Anda membuat host dengan CLI, gunakan konsol atau CLI untuk mengatur host untuk membuat statusnya. `Available`

Untuk membuat host

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows). Gunakan AWS CLI untuk menjalankan create-host perintah, menentukan, `--name``--provider-type`, dan `--provider-endpoint` untuk koneksi Anda. Dalam contoh ini, nama penyedia pihak ketiga adalah `GitHubEnterpriseServer` dan titik akhir adalah `my-instance.dev`.

   ```
   aws codestar-connections create-host --name MyHost --provider-type
    GitHubEnterpriseServer --provider-endpoint "https://my-instance.dev"
   ```

   Jika berhasil, perintah ini mengembalikan informasi Amazon Resource Name (ARN) host seperti berikut ini.

   ```
   {
       "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
   Host-28aef605"
   }
   ```

   Setelah langkah ini, host dalam status `PENDING`.

2. Gunakan konsol untuk menyelesaikan penyiapan host dan memindahkan host ke status `Available`.

Untuk membuat koneksi ke GitHub Enterprise Server

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows). Gunakan AWS CLI untuk menjalankan create-connection perintah, menentukan `--host-arn` dan `--connection-name` untuk koneksi Anda.

```
aws codestar-connections create-connection --host-arn arn:aws:codestar-
connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
 MyConnection
```

   Jika berhasil, perintah ini mengembalikan informasi ARN koneksi seperti berikut ini.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad"
}
```

2. Gunakan konsol untuk mengatur koneksi yang tertunda.
3. Pipeline default untuk mendeteksi perubahan pada push kode ke repositori sumber koneksi. Untuk mengonfigurasi konfigurasi pemicu pipeline untuk rilis manual atau untuk tag Git, lakukan salah satu hal berikut:

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memulai dengan rilis manual saja, tambahkan baris berikut ke konfigurasi:

```
"DetectChanges": "false",
```

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memfilter dengan pemicu, lihat detail selengkapnya di[Tambahkan pemicu dengan tipe acara permintaan push atau pull kode](). Misalnya, berikut ini menambah tingkat pipa definisi JSON pipa. Dalam contoh ini, `release-v0` dan `release-v1` merupakan tag Git untuk disertakan, dan `release-v2` merupakan tag Git untuk dikecualikan.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
```

```
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "tags": {
                                "includes": [
                                    "release-v0", "release-v1"
                                ],
                                "excludes": [
                                    "release-v2"
                                ]
                            }
                        }
                    ]
                }
            }
        ]
```

GitLabkoneksi .com

Koneksi memungkinkan Anda untuk mengotorisasi dan membuat konfigurasi yang mengaitkan penyedia pihak ketiga Anda dengan sumber daya Anda AWS . Untuk mengaitkan repositori pihak ketiga Anda sebagai sumber untuk pipeline Anda, Anda menggunakan koneksi.

> **Note**
>
> Alih-alih membuat atau menggunakan koneksi yang ada di akun Anda, Anda dapat menggunakan koneksi bersama antara yang lain Akun AWS. Lihat Menggunakan koneksi yang dibagikan dengan akun lain.

> **Note**
>
> Fitur ini tidak tersedia di Asia Pasifik (Hong Kong), Asia Pasifik (Hyderabad), Asia Pasifik (Jakarta), Asia Pasifik (Melbourne), Asia Pasifik (Osaka), Afrika (Cape Town), Timur Tengah (Bahrain), Timur Tengah (UEA), Eropa (Spanyol), Eropa (Zurich), Israel (Tel Aviv), atau (AS-Barat) Wilayah. AWS GovCloud Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline. Untuk pertimbangan dengan tindakan ini di Wilayah Eropa (Milan), lihat catatan diCodeStarSourceConnection untuk

Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Untuk menambahkan aksi GitLab sumber.com CodePipeline, Anda dapat memilih salah satu untuk:

- Gunakan wizard Buat pipeline CodePipeline konsol atau halaman tindakan Edit untuk memilih opsi GitLabpenyedia. Lihat Buat koneksi GitLab ke.com (konsol) untuk menambahkan tindakan. Konsol membantu Anda membuat sumber daya koneksi.
- Gunakan CLI untuk menambahkan konfigurasi tindakan untuk `CreateSourceConnection` tindakan dengan `GitLab` penyedia sebagai berikut:
  - Untuk membuat sumber daya koneksi Anda, lihat Buat koneksi GitLab ke.com (CLI) untuk membuat sumber daya koneksi dengan CLI.
  - Gunakan `CreateSourceConnection` contoh konfigurasi tindakan CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri untuk menambahkan tindakan Anda seperti yang ditunjukkan padaBuat pipeline (CLI).

> **Note**
>
> Anda juga dapat membuat koneksi menggunakan konsol Alat Pengembang di bawah Pengaturan. Lihat Membuat Koneksi.

> **Note**
>
> Dengan mengotorisasi instalasi koneksi ini di GitLab .com, Anda memberikan izin layanan kami untuk memproses data Anda dengan mengakses akun Anda, dan Anda dapat mencabut izin kapan saja dengan menghapus instalasi aplikasi.

Sebelum Anda memulai:

- Anda harus sudah membuat akun GitLab dengan.com.

> **Note**
>
> Koneksi hanya menyediakan akses ke repositori yang dimiliki oleh akun yang digunakan untuk membuat dan mengotorisasi koneksi.

> **Note**
>
> Anda dapat membuat koneksi ke repositori tempat Anda memiliki peran Pemilik GitLab, dan kemudian koneksi dapat digunakan dengan repositori dengan sumber daya seperti. CodePipeline Untuk repositori dalam grup, Anda tidak perlu menjadi pemilik grup.

- Untuk menentukan sumber untuk pipeline Anda, Anda harus sudah membuat repositori di gitlab.com.

Topik

- Buat koneksi GitLab ke.com (konsol)
- Buat koneksi GitLab ke.com (CLI)

Buat koneksi GitLab ke.com (konsol)

Gunakan langkah-langkah ini untuk menggunakan CodePipeline konsol untuk menambahkan tindakan koneksi untuk proyek Anda (repositori) di. GitLab

Untuk membuat atau mengedit pipeline

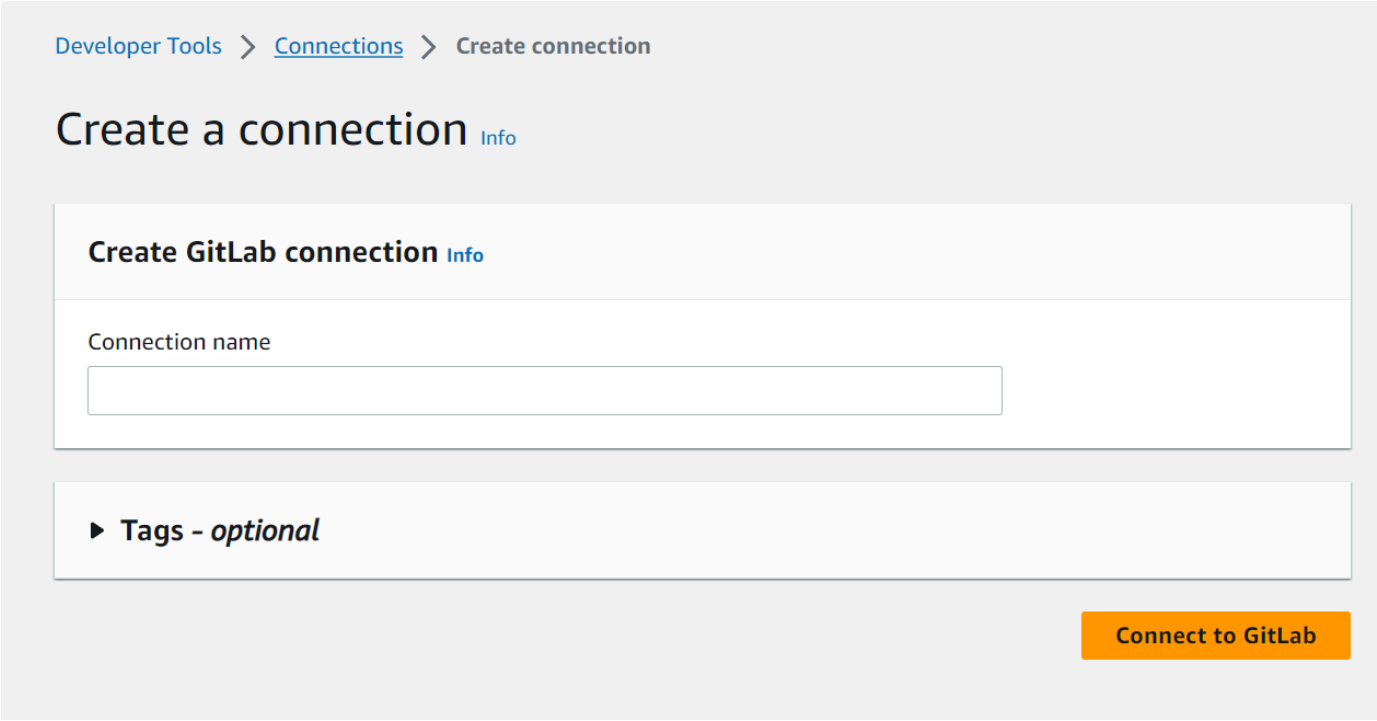
1. Masuk ke CodePipeline konsol.
2. Pilih salah satu dari berikut ini.
   - Pilih untuk membuat pipeline. Ikuti langkah-langkah di Create a Pipeline untuk menyelesaikan layar pertama dan pilih Berikutnya. Pada halaman Sumber, di bawah Penyedia Sumber, pilih GitLab.
   - Pilih untuk mengedit pipeline yang ada. Pilih Edit, lalu pilih Edit tahap. Pilih untuk menambahkan atau mengedit tindakan sumber Anda. Pada halaman Edit tindakan, di bawah Nama tindakan, masukkan nama untuk tindakan Anda. Di penyedia Action, pilih GitLab.

3. Lakukan salah satu tindakan berikut:

   - Di bawah Koneksi, jika Anda belum membuat sambungan ke penyedia Anda, pilih Connect to GitLab. Lanjutkan ke langkah 4 untuk membuat koneksi.
   - Di bawah Koneksi, jika Anda telah membuat koneksi ke penyedia Anda, pilih koneksi. Lanjutkan ke langkah 9.

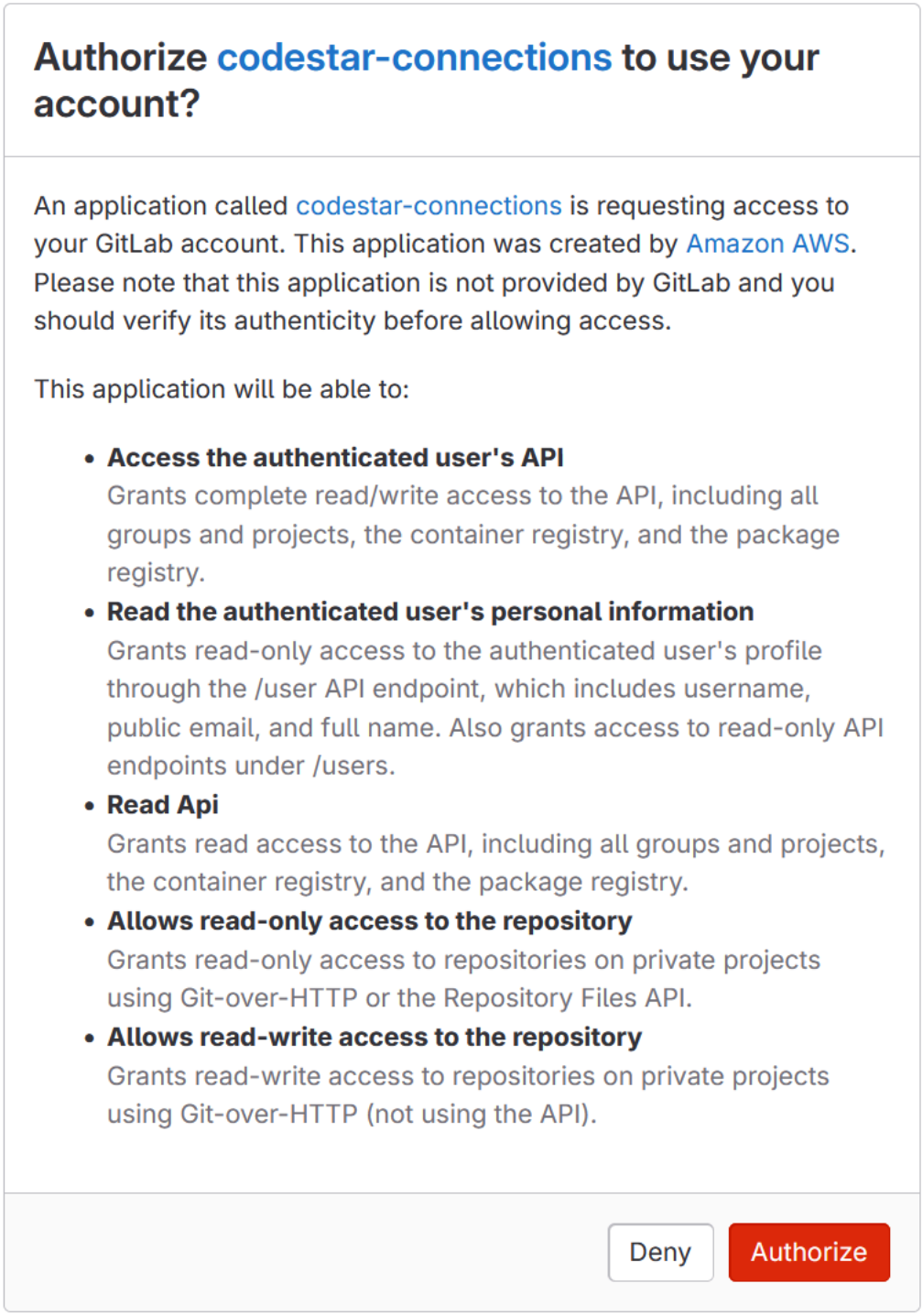
   > **Note**
   >
   > Jika Anda menutup jendela pop-up sebelum koneksi GitLab .com dibuat, Anda perlu me-refresh halaman.

4. Untuk membuat koneksi ke GitLab repositori.com, di bawah Pilih penyedia, pilih. GitLab Di Nama koneksi, masukkan nama untuk koneksi yang ingin Anda buat. Pilih Connect to GitLab.



5. Saat halaman masuk GitLab untuk.com ditampilkan, masuk dengan kredensialmu, lalu pilih Masuk.
6. Jika ini adalah pertama kalinya Anda mengotorisasi koneksi, halaman otorisasi ditampilkan dengan pesan yang meminta otorisasi untuk koneksi untuk mengakses akun.com Anda. GitLab

   Pilih Izinkan.



7. Browser kembali ke halaman konsol koneksi. Di bawah Buat GitLab koneksi, koneksi baru ditampilkan di Nama koneksi.
8. Pilih Connect to GitLab.

   Anda akan dikembalikan ke CodePipeline konsol.

> **Note**
>
> Setelah GitLab koneksi.com berhasil dibuat, spanduk sukses akan ditampilkan di jendela utama.
> Jika sebelumnya Anda belum masuk ke GitLab mesin saat ini, Anda harus menutup jendela pop-up secara manual.

9. Dalam nama Repositori, pilih nama proyek Anda GitLab dengan menentukan jalur proyek dengan namespace. Misalnya, untuk repositori tingkat grup, masukkan nama repositori dalam format berikut:. `group-name/repository-name` Untuk informasi selengkapnya tentang path dan namespace, lihat `path_with_namespace` bidang di https://docs.gitlab.com/ee/ api/ projects.html#. get-single-project Untuk informasi selengkapnya tentang namespace di GitLab, lihat https://docs.gitlab.com/ee/ user/namespace/.

> **Note**
>
> Untuk grup di GitLab, Anda harus secara manual menentukan jalur proyek dengan namespace. Misalnya, untuk repositori bernama `myrepo` dalam grup`mygroup`, masukkan yang berikut ini:. `mygroup/myrepo` Anda dapat menemukan jalur proyek dengan namespace di URL di. GitLab

10. Di bawah pemicu Pipeline, Anda dapat menambahkan pemicu jika tindakan Anda adalah CodeConnections tindakan. Untuk mengonfigurasi konfigurasi pemicu pipeline dan memfilter secara opsional dengan pemicu, lihat detail selengkapnya di. Tambahkan pemicu dengan tipe acara permintaan push atau pull kode
11. Di Nama cabang, pilih cabang tempat pipeline Anda ingin mendeteksi perubahan sumber.

> **Note**
>
> Jika nama cabang tidak terisi secara otomatis, maka Anda tidak memiliki akses Pemilik ke repositori. Entah nama proyek tidak valid, atau koneksi yang digunakan tidak memiliki akses ke proyek/repositori.

12. Dalam format artefak Output, Anda harus memilih format untuk artefak Anda.

- Untuk menyimpan artefak keluaran dari GitLab tindakan.com menggunakan metode default, pilih CodePipeline default. Tindakan mengakses file dari GitLab repositori.com dan menyimpan artefak dalam file ZIP di toko artefak pipeline.
- Untuk menyimpan file JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung, pilih klon Penuh. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.

  Jika Anda memilih opsi ini, Anda perlu memperbarui izin untuk peran layanan CodeBuild proyek Anda seperti yang ditunjukkan pada Tambahkan CodeBuild GitClone izin untuk koneksi ke Bitbucket,, GitHub Enterprise Server GitHub, atau .com GitLab. Untuk tutorial yang menunjukkan cara menggunakan opsi klon Penuh, lihat Tutorial: Gunakan klon lengkap dengan sumber GitHub pipa.

13. Pilih untuk menyimpan tindakan sumber dan lanjutkan.

Buat koneksi GitLab ke.com (CLI)

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk membuat koneksi.

Untuk melakukannya, gunakan perintah create-connection.

> **Important**
>
> Koneksi yang dibuat melalui AWS CLI atau AWS CloudFormation dalam `PENDING` status secara default. Setelah Anda membuat koneksi dengan CLI atau CloudFormation, gunakan konsol untuk mengedit koneksi untuk membuat statusnya. `AVAILABLE`

Untuk membuat koneksi

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows). Gunakan AWS CLI untuk menjalankan create-connection perintah, menentukan `--provider-type` dan `--connection-name` untuk koneksi Anda. Dalam contoh ini, nama penyedia pihak ketiga adalah `GitLab` dan nama koneksi yang ditentukan adalah `MyConnection`.

   ```
   aws codestar-connections create-connection --provider-type GitLab --connection-name
    MyConnection
   ```

Jika berhasil, perintah ini mengembalikan informasi ARN koneksi seperti berikut ini.

```
{
    "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
aEXAMPLE-8aad-4d5d-8878-dfcab0bc441f"
}
```

2. Gunakan konsol untuk menyelesaikan koneksi. Untuk informasi selengkapnya, lihat Memperbarui sambungan yang tertunda.
3. Pipeline default untuk mendeteksi perubahan pada push kode ke repositori sumber koneksi. Untuk mengonfigurasi konfigurasi pemicu pipeline untuk rilis manual atau untuk tag Git, lakukan salah satu hal berikut:

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memulai dengan rilis manual saja, tambahkan baris berikut ke konfigurasi:

     ```
     "DetectChanges": "false",
     ```

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memfilter dengan pemicu, lihat detail selengkapnya diTambahkan pemicu dengan tipe acara permintaan push atau pull kode. Misalnya, berikut ini menambah tingkat pipa definisi JSON pipa. Dalam contoh ini, `release-v0` dan `release-v1` merupakan tag Git untuk disertakan, dan `release-v2` merupakan tag Git untuk dikecualikan.

     ```
     "triggers": [
                {
                    "providerType": "CodeStarSourceConnection",
                    "gitConfiguration": {
                        "sourceActionName": "Source",
                        "push": [
                            {
                                "tags": {
                                    "includes": [
                                        "release-v0", "release-v1"
                                    ],
                                    "excludes": [
                                        "release-v2"
                                    ]
                                }
                            }
     ```

```
                ]
            }
        }
    ]
```

Koneksi untuk dikelola GitLab sendiri

Koneksi memungkinkan Anda untuk mengotorisasi dan membuat konfigurasi yang mengaitkan penyedia pihak ketiga Anda dengan sumber daya Anda AWS . Untuk mengaitkan repositori pihak ketiga Anda sebagai sumber untuk pipeline Anda, Anda menggunakan koneksi.

> **Note**
>
> Alih-alih membuat atau menggunakan koneksi yang ada di akun Anda, Anda dapat menggunakan koneksi bersama antara yang lain Akun AWS. Lihat Menggunakan koneksi yang dibagikan dengan akun lain.

> **Note**
>
> Fitur ini tidak tersedia di Asia Pasifik (Hong Kong), Asia Pasifik (Hyderabad), Asia Pasifik (Jakarta), Asia Pasifik (Melbourne), Asia Pasifik (Osaka), Afrika (Cape Town), Timur Tengah (Bahrain), Timur Tengah (UEA), Eropa (Spanyol), Eropa (Zurich), Israel (Tel Aviv), atau (AS-Barat) Wilayah. AWS GovCloud Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline. Untuk pertimbangan dengan tindakan ini di Wilayah Eropa (Milan), lihat catatan diCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Untuk menambahkan tindakan sumber yang GitLab dikelola sendiri CodePipeline, Anda dapat memilih salah satu untuk:

- Gunakan wizard Buat pipeline CodePipeline konsol atau halaman tindakan Edit untuk memilih opsi penyedia yang GitLabdikelola sendiri. Lihat Buat koneksi ke GitLab kelola sendiri (konsol) untuk menambahkan tindakan. Konsol membantu Anda membuat sumber daya host dan sumber daya koneksi.

- Gunakan CLI untuk menambahkan konfigurasi tindakan untuk `CreateSourceConnection` tindakan dengan `GitLabSelfManaged` penyedia dan membuat sumber daya Anda:
  - Untuk membuat sumber daya koneksi Anda, lihat Buat host dan koneksi ke GitLab self-managed (CLI) untuk membuat sumber daya host dan sumber daya koneksi dengan CLI.
  - Gunakan `CreateSourceConnection` contoh konfigurasi tindakan CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri untuk menambahkan tindakan Anda seperti yang ditunjukkan padaBuat pipeline (CLI).

> **Note**
>
> Anda juga dapat membuat koneksi menggunakan konsol Alat Pengembang di bawah Pengaturan. Lihat Membuat Koneksi.

Sebelum Anda memulai:

- Anda harus sudah membuat akun dengan GitLab dan memiliki GitLab Enterprise Edition atau GitLab Community Edition dengan instalasi yang dikelola sendiri. Untuk informasi selengkapnya, lihat https://docs.gitlab.com/ee/subscriptions/self_managed/.

  > **Note**
  >
  > Koneksi hanya menyediakan akses untuk akun yang digunakan untuk membuat dan mengotorisasi koneksi.

  > **Note**
  >
  > Anda dapat membuat koneksi ke repositori tempat Anda memiliki peran Pemilik GitLab, dan kemudian koneksi dapat digunakan dengan sumber daya seperti. CodePipeline Untuk repositori dalam grup, Anda tidak perlu menjadi pemilik grup.

- Anda harus sudah membuat token akses GitLab pribadi (PAT) dengan izin cakupan bawah berikut saja: api. Untuk informasi lebih lanjut, lihat https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html. Anda harus menjadi administrator untuk membuat dan menggunakan PAT.

> **Note**
>
> PAT Anda digunakan untuk mengotorisasi host dan tidak disimpan atau digunakan oleh koneksi. Untuk mengatur host, Anda dapat membuat PAT sementara dan kemudian setelah Anda mengatur host, Anda dapat menghapus PAT.

- Anda dapat memilih untuk mengatur host Anda sebelumnya. Anda dapat mengatur host dengan atau tanpa VPC. Untuk detail tentang konfigurasi VPC dan informasi tambahan tentang membuat host, lihat Membuat host.

Topik

- Buat koneksi ke GitLab kelola sendiri (konsol)
- Buat host dan koneksi ke GitLab self-managed (CLI)

Buat koneksi ke GitLab kelola sendiri (konsol)

Gunakan langkah-langkah ini untuk menggunakan CodePipeline konsol guna menambahkan tindakan koneksi untuk repositori GitLab self-managedr Anda.

> **Note**
>
> GitLab koneksi yang dikelola sendiri hanya menyediakan akses ke repositori yang dimiliki oleh akun yang GitLab dikelola sendiri yang digunakan untuk membuat koneksi.

Sebelum Anda mulai:

Untuk koneksi host ke GitLab self-managed, Anda harus telah menyelesaikan langkah-langkah untuk membuat sumber daya host untuk koneksi Anda. Lihat Mengelola host untuk koneksi.

Langkah 1: Buat atau edit pipeline Anda

Untuk membuat atau mengedit pipeline

1. Masuk ke CodePipeline konsol.
2. Pilih salah satu dari berikut ini.

- Pilih untuk membuat pipeline. Ikuti langkah-langkah di Create a Pipeline untuk menyelesaikan layar pertama dan pilih Berikutnya. Pada halaman Sumber, di bawah Penyedia sumber, pilih GitLab dikelola sendiri.
- Pilih untuk mengedit pipeline yang ada. Pilih Edit, lalu pilih Edit tahap. Pilih untuk menambahkan atau mengedit tindakan sumber Anda. Pada halaman Edit tindakan, di bawah Nama tindakan, masukkan nama untuk tindakan Anda. Di penyedia Action, pilih yang GitLab dikelola sendiri.

3. Lakukan salah satu tindakan berikut:

   - Di bawah Koneksi, jika Anda belum membuat sambungan ke penyedia Anda, pilih Connect to GitLab self-managed. Lanjutkan ke Langkah 2: Buat Koneksi untuk GitLab dikelola sendiri.
   - Di bawah Koneksi, jika Anda telah membuat koneksi ke penyedia Anda, pilih koneksi, lalu lanjutkan ke Langkah 3: Simpan tindakan sumber yang GitLab dikelola sendiri.

Langkah 2: Buat koneksi ke GitLab self-managed

Setelah Anda memilih untuk membuat koneksi, halaman Connect to GitLab self-managed akan ditampilkan.

Untuk terhubung ke yang dikelola GitLab sendiri

1. Di Nama koneksi, masukkan nama untuk koneksi Anda.
2. Di URL, masukkan titik akhir untuk server Anda.

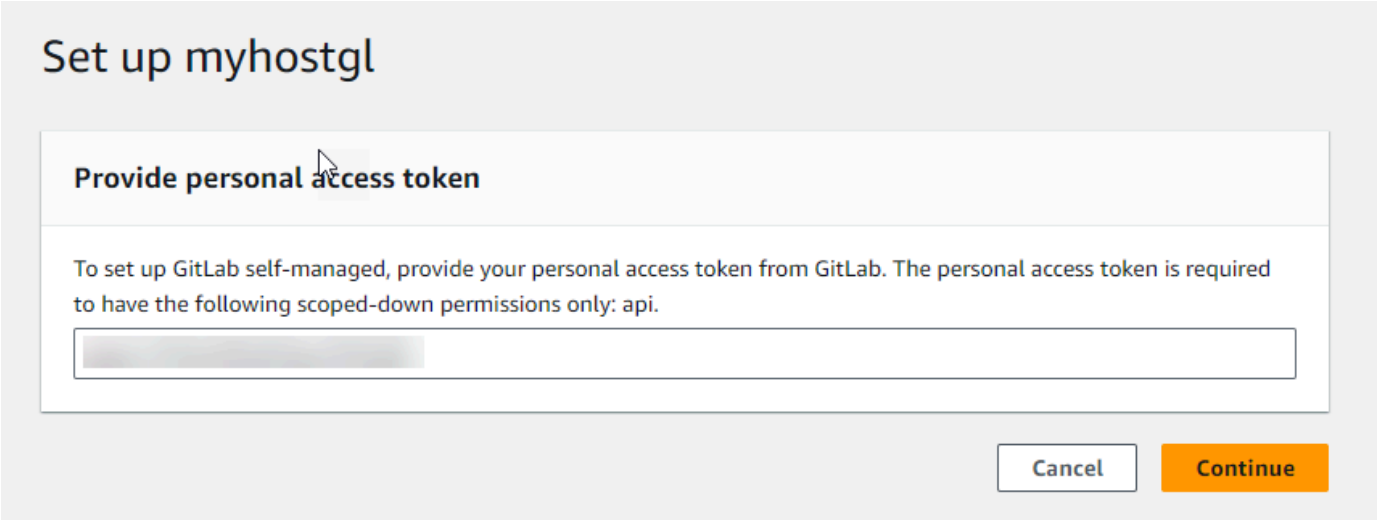
   > **Note**
   >
   > Jika URL yang disediakan telah digunakan untuk menyiapkan host untuk koneksi, Anda akan diminta untuk memilih ARN sumber daya host yang dibuat sebelumnya untuk titik akhir tersebut.

3. Jika Anda telah meluncurkan server Anda ke VPC Amazon dan Anda ingin terhubung dengan VPC Anda, pilih Gunakan VPC dan lengkapi informasi untuk VPC.
4. Pilih Connect to GitLab self-managed. Koneksi yang dibuat ditampilkan dengan status Tertunda. Sumber daya host dibuat untuk koneksi dengan informasi server yang Anda berikan. Untuk nama host, URL digunakan.
5. Pilih Perbarui koneksi tertunda.

6. Jika halaman terbuka dengan pesan pengalihan yang mengonfirmasi bahwa Anda ingin melanjutkan ke penyedia, pilih Lanjutkan. Masukkan otorisasi untuk penyedia.
7. A Menetapkan tampilan *host_name* halaman. Di Berikan token akses pribadi, berikan GitLab PAT Anda hanya izin cakupan bawah berikut:. `api`

   > **Note**
   >
   > Hanya administrator yang dapat membuat dan menggunakan PAT.

   Pilih Lanjutkan.



8. Halaman koneksi yang menampilkan koneksi yang dibuat dalam status Tersedia.

Langkah 3: Simpan tindakan sumber yang GitLab dikelola sendiri

Gunakan langkah-langkah ini di halaman wizard atau Edit tindakan untuk menyimpan tindakan sumber Anda dengan informasi koneksi Anda.

Untuk menyelesaikan dan menyimpan tindakan sumber Anda dengan koneksi Anda

1. Dalam nama Repositori, pilih nama repositori pihak ketiga Anda.
2. Di bawah pemicu Pipeline, Anda dapat menambahkan pemicu jika tindakan Anda adalah CodeConnections tindakan. Untuk mengonfigurasi konfigurasi pemicu pipeline dan memfilter secara opsional dengan pemicu, lihat detail selengkapnya di. Tambahkan pemicu dengan tipe acara permintaan push atau pull kode

3. Dalam format artefak Output, Anda harus memilih format untuk artefak Anda.

   - Untuk menyimpan artefak keluaran dari tindakan yang GitLab dikelola sendiri menggunakan metode default, pilih CodePipeline default. Tindakan mengakses file dari repositori dan menyimpan artefak dalam file ZIP di toko artefak pipa.
   - Untuk menyimpan file JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung, pilih klon Penuh. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.

4. Pilih Berikutnya di wizard atau Simpan di halaman Edit tindakan.

Buat host dan koneksi ke GitLab self-managed (CLI)

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk membuat koneksi.

Untuk melakukannya, gunakan perintah create-connection.

> **Important**
>
> Koneksi yang dibuat melalui AWS CLI atau AWS CloudFormation dalam `PENDING` status secara default. Setelah Anda membuat koneksi dengan CLI atau CloudFormation, gunakan konsol untuk mengedit koneksi untuk membuat statusnya. `AVAILABLE`

Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk membuat host untuk koneksi terinstal.

Anda menggunakan host untuk mewakili titik akhir infrastruktur tempat penyedia pihak ketiga Anda diinstal. Setelah Anda menyelesaikan pembuatan host dengan CLI, host dalam status Tertunda. Anda kemudian mengatur, atau mendaftarkan, host untuk memindahkannya ke status Tersedia. Setelah host tersedia, Anda menyelesaikan langkah-langkah untuk membuat koneksi.

Untuk melakukannya, gunakan perintah create-host.

> **Important**
>
> Host yang dibuat melalui status AWS CLI berada dalam `Pending` status secara default. Setelah Anda membuat host dengan CLI, gunakan konsol atau CLI untuk mengatur host untuk membuat statusnya. `Available`

Untuk membuat host

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows). Gunakan AWS CLI untuk menjalankan create-host perintah, menentukan, `--name--provider-type`, dan `--provider-endpoint` untuk koneksi Anda. Dalam contoh ini, nama penyedia pihak ketiga adalah `GitLabSelfManaged` dan titik akhir adalah `my-instance.dev`.

   ```
   aws codestar-connections create-host --name MyHost --provider-type
    GitLabSelfManaged --provider-endpoint "https://my-instance.dev"
   ```

   Jika berhasil, perintah ini mengembalikan informasi Amazon Resource Name (ARN) host seperti berikut ini.

   ```
   {
       "HostArn": "arn:aws:codestar-connections:us-west-2:account_id:host/My-
   Host-28aef605"
   }
   ```

   Setelah langkah ini, host dalam status `PENDING`.

2. Gunakan konsol untuk menyelesaikan penyiapan host dan memindahkan host ke status `Available`.

Untuk membuat koneksi ke GitLab self-managed

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows). Gunakan AWS CLI untuk menjalankan create-connection perintah, menentukan `--host-arn` dan `--connection-name` untuk koneksi Anda.

   ```
   aws codestar-connections create-connection --host-arn arn:aws:codestar-
   connections:us-west-2:account_id:host/MyHost-234EXAMPLE --connection-name
    MyConnection
   ```

   Jika berhasil, perintah ini mengembalikan informasi ARN koneksi seperti berikut ini.

   ```
   {
       "ConnectionArn": "arn:aws:codestar-connections:us-west-2:account_id:connection/
   aEXAMPLE-8aad"
   }
   ```

2. Gunakan konsol untuk mengatur koneksi yang tertunda.
3. Pipeline default untuk mendeteksi perubahan pada push kode ke repositori sumber koneksi. Untuk mengonfigurasi konfigurasi pemicu pipeline untuk rilis manual atau untuk tag Git, lakukan salah satu hal berikut:

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memulai dengan rilis manual saja, tambahkan baris berikut ke konfigurasi:

     ```
     "DetectChanges": "false",
     ```

   - Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memfilter dengan pemicu, lihat detail selengkapnya di[Tambahkan pemicu dengan tipe acara permintaan push atau pull kode]. Misalnya, berikut ini menambah tingkat pipa definisi JSON pipa. Dalam contoh ini, `release-v0` dan `release-v1` merupakan tag Git untuk disertakan, dan `release-v2` merupakan tag Git untuk dikecualikan.

     ```
     "triggers": [
                 {
                     "providerType": "CodeStarSourceConnection",
                     "gitConfiguration": {
                         "sourceActionName": "Source",
                         "push": [
                             {
                                 "tags": {
                                     "includes": [
                                         "release-v0", "release-v1"
                                     ],
                                     "excludes": [
                                         "release-v2"
                                     ]
                                 }
                             }
                         ]
                     }
                 }
             ]
     ```

Menggunakan koneksi yang dibagikan dengan akun lain

Anda dapat membuat dan mengelola koneksi bersama menggunakan AWS RAM. Hal ini memungkinkan koneksi untuk dibagi antara Akun AWS untuk akses ke repositori pihak ketiga. Hal ini memungkinkan koneksi tunggal untuk digunakan dalam CodePipeline pipeline di seluruh akun sekaligus mengurangi kebutuhan pengguna untuk mengelola dan mengelola koneksi terpisah di setiap akun.

Untuk menggunakan koneksi bersama CodePipeline, lakukan hal berikut.

- Buat koneksi menggunakan konsol Alat Pengembang di bawah Pengaturan. Lihat Membuat Koneksi.
- Siapkan pembagian sumber daya menggunakan AWS RAM. Lihat Berbagi koneksi dengan Akun AWS.
- Saat Anda menggunakan wizard Buat pipeline CodePipeline konsol atau halaman tindakan Edit untuk memilih penyedia koneksi, seperti opsi penyedia Bitbucket, Anda dapat memilih koneksi yang telah dibagikan dengan akun target.

Otomatiskan saluran pipa awal menggunakan pemicu dan penyaringan

Pemicu memungkinkan Anda mengonfigurasi pipeline untuk memulai pada jenis peristiwa tertentu atau jenis peristiwa yang difilter, seperti saat perubahan pada cabang tertentu atau permintaan tarik terdeteksi. Pemicu dapat dikonfigurasi untuk tindakan sumber dengan koneksi yang menggunakan `CodeStarSourceConnection` tindakan di CodePipeline, seperti, Bitbucket GitHub, dan. GitLab Untuk informasi selengkapnya tentang tindakan sumber yang menggunakan koneksi, lihatTambahkan penyedia sumber pihak ketiga ke saluran pipa menggunakan CodeConnections.

Tindakan sumber, seperti CodeCommit dan S3, menggunakan deteksi perubahan otomatis untuk memulai saluran pipa saat perubahan dilakukan. Untuk informasi selengkapnya, lihat CodeCommit tindakan sumber dan EventBridge.

Anda menentukan pemicu menggunakan konsol atau CLI.

Anda menentukan jenis filter sebagai berikut:

- Tidak ada filter

  Konfigurasi pemicu ini memulai pipeline Anda pada setiap push ke cabang default yang ditentukan sebagai bagian dari konfigurasi tindakan.
- Tentukan filter

  Anda menambahkan filter yang memulai pipeline Anda pada filter tertentu, seperti pada nama cabang untuk push kode, dan mengambil komit yang tepat. Ini juga mengonfigurasi pipeline untuk tidak memulai secara otomatis pada perubahan apa pun.
  - Dorong
    - Kombinasi filter yang valid adalah:
      - Tag Git

        Sertakan atau kecualikan
      - cabang

        Sertakan atau kecualikan
      - cabang+jalur file

Sertakan atau kecualikan

- Permintaan tarik
  - Kombinasi filter yang valid adalah:
    - cabang

      Sertakan atau kecualikan
    - cabang+jalur file

      Sertakan atau kecualikan
- Jangan mendeteksi perubahan

  Ini tidak menambahkan pemicu dan pipa tidak dimulai secara otomatis pada perubahan apa pun.

Tabel berikut menyediakan opsi filter yang valid untuk setiap jenis acara. Tabel juga menunjukkan konfigurasi pemicu mana yang default ke true atau false untuk deteksi perubahan otomatis dalam konfigurasi tindakan.

Konfigurasi pemicu	Tipe peristiwa	Opsi filter	Mendeteksi perubahan
Tambahkan pemicu - tanpa filter	none	none	true
Tambahkan pemicu — filter pada push kode	acara push	Git tag, cabang, jalur berkas	false
Tambahkan pemicu — filter untuk permintaan tarik	permintaan tarik	cabang, jalur file	false
Tidak ada pemicu - jangan mendeteksi	none	none	false

> **Note**
>
> Jenis pemicu ini menggunakan deteksi perubahan otomatis (sebagai tipe `Webhook` pemicu). Penyedia tindakan sumber yang menggunakan tipe pemicu ini adalah koneksi yang dikonfigurasi untuk push kode (Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab .com, dan GitLab dikelola sendiri).

Untuk definisi bidang dan referensi lebih lanjut untuk pemicu, lihat

Untuk daftar definisi bidang dalam struktur JSON, lihat`triggers`.

Untuk pemfilteran, pola ekspresi reguler dalam format glob didukung seperti yang dijelaskan dalam. Bekerja dengan pola glob dalam sintaks

> **Note**
>
> Dalam kasus tertentu, untuk pipeline dengan pemicu yang difilter pada jalur file, pipeline mungkin tidak dimulai saat cabang dengan filter jalur file pertama kali dibuat. Untuk informasi selengkapnya, lihat Saluran pipa dengan koneksi yang menggunakan pemfilteran pemicu berdasarkan jalur file mungkin tidak dimulai pada pembuatan cabang.

Pertimbangan untuk filter pemicu

Pertimbangan berikut berlaku saat menggunakan pemicu.

- Anda tidak dapat menambahkan lebih dari satu pemicu per tindakan sumber.
- Anda dapat menambahkan beberapa jenis filter ke pemicu. Sebagai contoh, lihat 4: Pemicu dengan dua jenis filter push dengan termasuk dan pengecualian yang saling bertentangan.
- Untuk pemicu dengan filter cabang dan jalur file, saat mendorong cabang untuk pertama kalinya, pipeline tidak akan berjalan karena tidak ada akses ke daftar file yang diubah untuk cabang yang baru dibuat.
- Menggabungkan permintaan tarik dapat memicu dua eksekusi pipeline jika konfigurasi pemicu push (branch filter) dan pull request (branch filter) memicu berpotongan.
- Untuk filter yang memicu pipeline Anda pada peristiwa permintaan tarik, untuk jenis peristiwa permintaan tarik Tertutup, penyedia repositori pihak ketiga untuk koneksi Anda mungkin

memiliki status terpisah untuk peristiwa gabungan. Misalnya, di Bitbucket, peristiwa Git untuk penggabungan bukanlah peristiwa penutupan permintaan tarik. Namun, dalam GitHub, menggabungkan permintaan tarik adalah peristiwa penutupan. Untuk informasi selengkapnya, lihat Peristiwa permintaan tarik untuk pemicu oleh penyedia.

Peristiwa permintaan tarik untuk pemicu oleh penyedia

Tabel berikut menyediakan ringkasan peristiwa Git, seperti untuk penutupan permintaan tarik, yang menghasilkan jenis peristiwa permintaan tarik menurut penyedia.

	Penyedia repositori untuk koneksi Anda			
Acara PR untuk pemicu	Bitbucket	GitHub	GHES	GitLab
Buka - Opsi ini memicu pipeline saat permintaan tarik dibuat untuk jalur cabang/file.	Membuat permintaan tarik menghasilkan peristiwa Git Terbuka.	Membuat permintaan tarik menghasilkan peristiwa Git Terbuka.	Membuat permintaan tarik menghasilkan peristiwa Git Terbuka.	Membuat permintaan tarik menghasilkan peristiwa Git Terbuka.
Pembaruan - Opsi ini memicu pipeline saat revisi permintaan tarik diterbitkan untuk jalur cabang/file.	Menerbitkan hasil pembaruan dalam peristiwa Git yang Diperbarui.	Menerbitkan hasil pembaruan dalam peristiwa Git yang Diperbarui.	Menerbitkan hasil pembaruan dalam peristiwa Git yang Diperbarui.	Menerbitkan hasil pembaruan dalam peristiwa Git yang Diperbarui.
Ditutup - Opsi ini memicu pipeline saat permintaan tarik ditutup untuk branch/file jalur.	Menggabungkan permintaan tarik di Bitbucket menghasilkan peristiwa Git Tertutup. Penting: Menutup	Menggabungkan atau menutup permintaan tarik secara manual menghasilkan peristiwa Git Tertutup.	Menggabungkan atau menutup permintaan tarik secara manual menghasilkan peristiwa Git Tertutup.	Menggabungkan atau menutup permintaan tarik secara manual menghasilkan peristiwa Git Tertutup.

	Penyedia repositori untuk koneksi Anda			
Acara PR untuk pemicu	Bitbucket	GitHub	GHES	GitLab
	permintaan tarik secara manual di Bitbucket tanpa penggabungan tidak menghasil kan peristiwa Git Tertutup.			

Contoh untuk filter pemicu

Untuk konfigurasi Git dengan filter untuk jenis peristiwa permintaan push dan pull, filter yang ditentukan mungkin bertentangan satu sama lain. Berikut ini adalah contoh kombinasi filter yang valid untuk acara permintaan push dan pull. Pemicu dapat berisi beberapa jenis filter, seperti dua jenis filter push dalam konfigurasi pemicu, dan jenis filter permintaan push dan pull akan menggunakan operasi OR di antara mereka, yang berarti setiap kecocokan akan memulai pipeline. Demikian pula, setiap jenis filter dapat menyertakan beberapa filter seperti FilePaths dan cabang; filter ini akan menggunakan operasi AND, yang berarti hanya kecocokan penuh yang akan memulai pipeline. Setiap jenis filter dapat berisi termasuk dan tidak termasuk, dan ini akan menggunakan operasi AND di antara mereka, yang berarti hanya kecocokan penuh yang akan memulai pipeline. Nama di dalam include/exclude, seperti nama cabang, menggunakan operasi OR. Jika ada konflik seperti antara dua filter Push seperti di mana satu menyertakan `main` cabang dan satu mengecualikannya, maka defaultnya adalah mengecualikan. Daftar berikut merangkum operasi untuk setiap bagian dari objek konfigurasi Git.

Untuk daftar definisi bidang dalam struktur JSON dan referensi terperinci untuk menyertakan dan mengecualikan, lihat. `triggers`

Example 1: Jenis filter dengan filter untuk cabang dan jalur file (DAN operasi)

Untuk jenis filter tunggal seperti permintaan tarik, Anda dapat menggabungkan filter, dan filter ini akan menggunakan operasi AND, yang berarti hanya kecocokan penuh yang akan memulai pipeline. Contoh berikut menunjukkan konfigurasi Git untuk jenis acara push dengan dua filter berbeda (`filePathsdanbranches`). Dalam contoh berikut, `filePaths` akan dan'ed dengan: `branches`

```
{
  "filePaths": {
    "includes": ["common/**/*.js"]
  },
  "branches": {
    "includes": ["feature/**"]
  }
}
```

Dengan konfigurasi Git di atas, contoh ini menunjukkan peristiwa yang akan memulai eksekusi pipeline karena operasi AND berhasil. Dengan kata lain, jalur `common/app.js` file disertakan untuk filter, yang memulai pipeline sebagai AND bahkan jika cabang `refs/heads/feature/triggers` yang ditentukan tidak berdampak.

```
{
  "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "common/app.js"
      ]
      ...
    }
  ]
}
```

Contoh berikut menunjukkan peristiwa untuk pemicu dengan konfigurasi di atas yang tidak akan memulai eksekusi pipeline karena cabang dapat memfilter, tetapi jalur file tidak.

```
{
   "ref": "refs/heads/feature/triggers",
  ...
  "commits": [
    {
      ...
      "modified": [
        "src/Main.java"
      ]
      ...
    }
```

```
  ]
}
```

Example 2: Termasuk dan tidak termasuk menggunakan operasi AND di antara mereka

Filter pemicu, seperti cabang dalam satu jenis peristiwa permintaan tarik, menggunakan operasi AND antara include dan excludes. Ini memungkinkan Anda mengonfigurasi beberapa pemicu untuk memulai eksekusi untuk pipeline yang sama. Contoh berikut menunjukkan konfigurasi pemicu dengan tipe filter tunggal (`branches`) di objek konfigurasi untuk acara push. `Excludes`Operasi `Includes` dan akan dan'ed, artinya jika cabang cocok dengan pola pengecualian (seperti `feature-branch` dalam contoh), pipeline tidak akan dipicu kecuali include juga cocok. Jika pola include cocok, seperti untuk `main` cabang, maka pipeline akan dipicu.

Untuk contoh berikut JSON:

- Mendorong komit ke `main` cabang akan memicu pipa
- Mendorong komit ke `feature-branch` cabang tidak akan memicu pipa.

```
{
  "branches": {
      "Includes": [
          "main"
      ],
      "Excludes": [
          "feature-branch"
      ]
   }
```

Example 3: Pemicu dengan jenis filter permintaan push dan pull (operasi OR), filter untuk jalur file dan cabang (DAN operasi), dan includes/excludes (operasi AND)

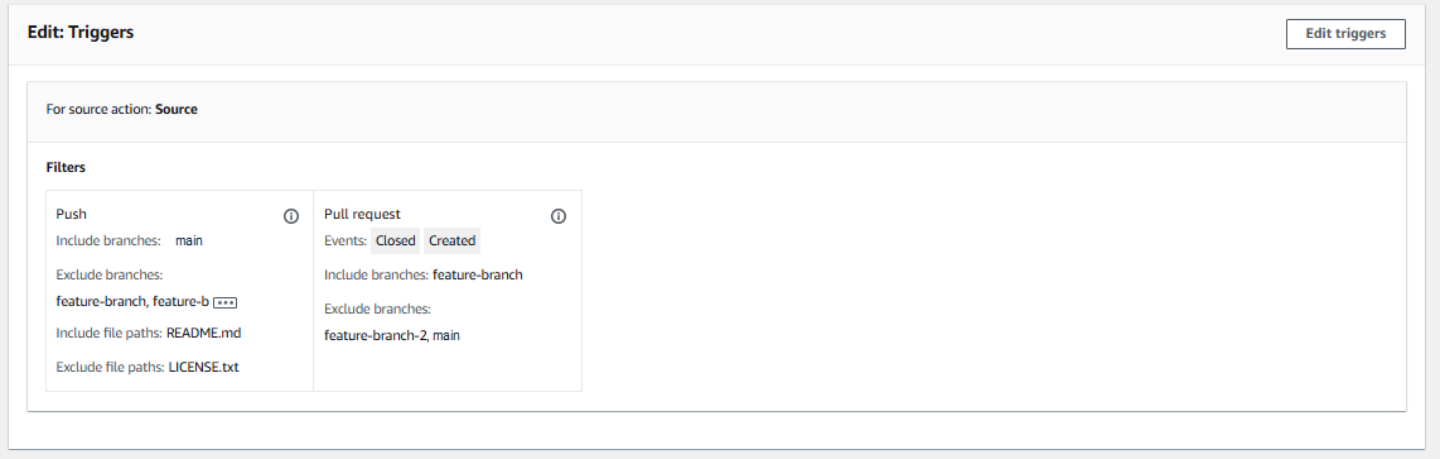
Objek konfigurasi pemicu, seperti pemicu yang berisi jenis peristiwa push dan jenis peristiwa permintaan tarik, menggunakan operasi OR antara dua jenis peristiwa. Contoh berikut menunjukkan konfigurasi pemicu dengan jenis acara push dengan `main` cabang yang disertakan dan satu jenis peristiwa permintaan tarik dengan cabang yang sama `main` dikecualikan. Selain itu, jenis acara push memiliki satu jalur file yang `LICENSE.txt` dikecualikan dan satu jalur file `README.MD` disertakan. Untuk jenis peristiwa kedua, permintaan tarik yang ada `Closed` atau `Created` di `feature-branch` cabang (termasuk) memulai pipeline, dan pipeline tidak dimulai saat membuat atau menutup permintaan tarik pada `main` cabang

`feature-branch-2` atau (dikecualikan) `Includes` .Dan `Excludes` operasinya akan DAN'd, dengan konflik default ke pengecualian. Misalnya, untuk acara permintaan tarik di `feature-branch` cabang (disertakan untuk permintaan tarik) sementara `feature-branch` cabang dikecualikan untuk jenis acara push, jadi defaultnya adalah mengecualikan.

Untuk contoh berikut,

- Mendorong komit ke `main` cabang (termasuk) untuk jalur `README.MD` file (termasuk) akan memicu pipeline.
- Di `feature-branch` cabang (dikecualikan), mendorong komit tidak akan memicu pipeline.
- Pada cabang yang disertakan, mengedit jalur `README.MD` file (termasuk) memicu pipeline.
- Pada cabang yang disertakan, mengedit jalur `LICENSE.TXT` file (dikecualikan) tidak memicu pipeline.
- Di `feature-branch` cabang, menutup permintaan tarik untuk jalur `README.MD` file (disertakan untuk acara push) tidak akan memicu pipeline karena jenis acara push menentukan `feature-branch` cabang sebagai dikecualikan sehingga konflik default ke pengecualian.

Gambar berikut menunjukkan konfigurasi.



Berikut ini adalah contoh JSON untuk konfigurasi.

```
"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"
                            ]
                        }
                    }
                ]
            }
        }
    ]
},
```

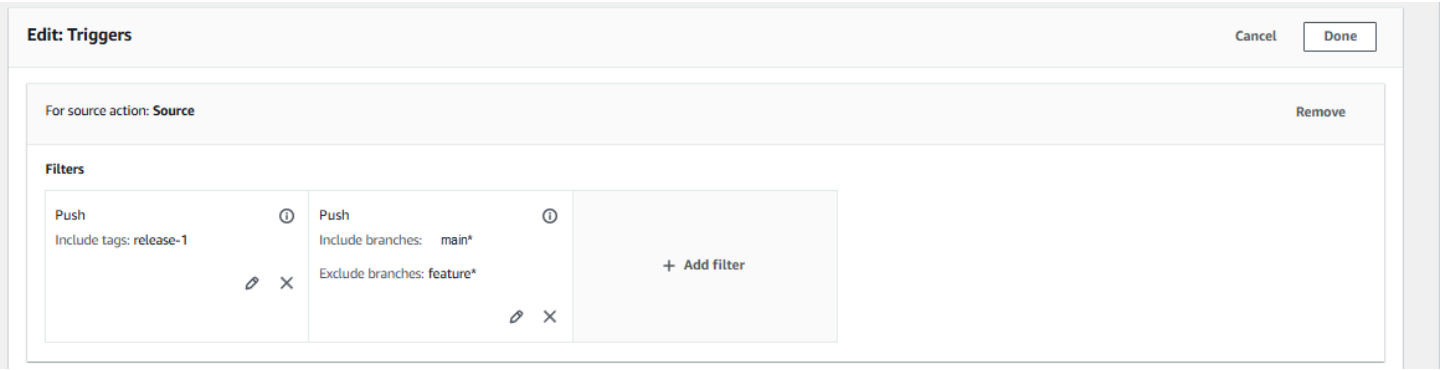
Example 4: Pemicu dengan dua jenis filter push dengan termasuk dan pengecualian yang saling bertentangan

Gambar berikut menunjukkan jenis filter push yang menentukan untuk memfilter pada tag `release-1` (termasuk). Jenis filter dorong kedua ditambahkan yang menentukan untuk memfilter pada cabang `main` (termasuk) dan untuk tidak memulai dorongan ke `feature*` cabang (dikecualikan).

Untuk contoh berikut:

- Mendorong rilis dari tag `release-1` (disertakan untuk filter push pertama) di `feature-branch` cabang (dikecualikan `feature*` untuk filter push kedua) tidak akan memicu pipeline karena dua jenis acara akan DAN'd.
- Mendorong pelepasan dari `main` cabang (termasuk untuk filter Push kedua) akan memulai pipa.

Contoh halaman Edit berikut menunjukkan dua jenis filter Push dan konfigurasinya untuk menyertakan dan mengecualikan.



Berikut ini adalah contoh JSON untuk konfigurasi.

```
"triggers": [
            {
                "providerType": "CodeStarSourceConnection",
                "gitConfiguration": {
                    "sourceActionName": "Source",
                    "push": [
                        {
                            "tags": {
                                "includes": [
                                    "release-1"
                                ]
```

```
                        }
                    },
                    {
                        "branches": {
                            "includes": [
                                "main*"
                            ],
                            "excludes": [
                                "feature*"
                            ]
                        }
                    }
                ]
            }
        }
    ]
},
```

Example 5: Pemicu dikonfigurasi sementara konfigurasi tindakan default BranchName digunakan untuk memulai manual

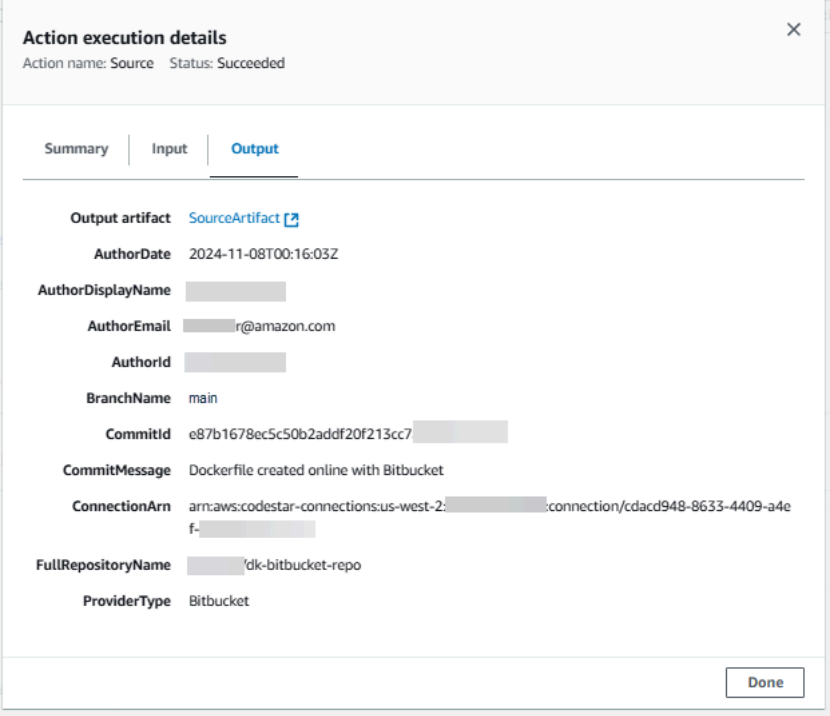
`BranchName`Bidang default konfigurasi tindakan mendefinisikan satu cabang yang akan digunakan saat pipeline dimulai secara manual, sementara pemicu dengan filter dapat digunakan untuk cabang atau cabang apa pun yang Anda tentukan.

Berikut ini adalah contoh JSON untuk konfigurasi tindakan yang menunjukkan `BranchName` bidang.

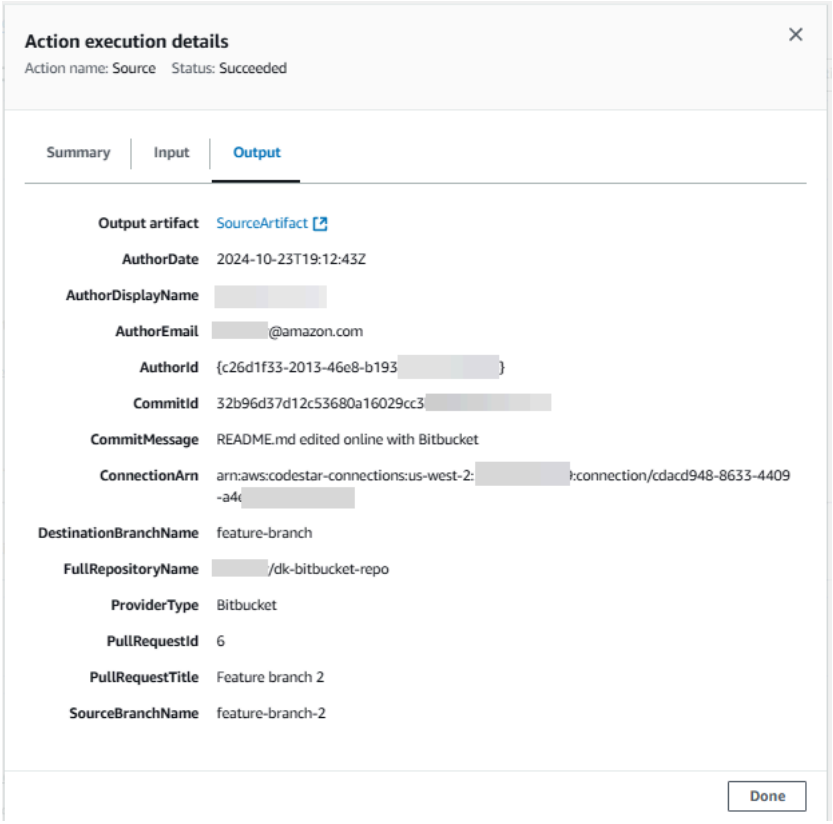
```
{
            "name": "Source",
            "actions": [
                {
                    "name": "Source",
                    "actionTypeId": {
                        "category": "Source",
                        "owner": "AWS",
                        "provider": "CodeStarSourceConnection",
                        "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"
        }
    ],
```

Contoh keluaran tindakan berikut menunjukkan cabang utama default digunakan saat pipeline dimulai secara manual.



Contoh keluaran tindakan berikut menunjukkan permintaan tarik dan cabang yang digunakan untuk pemicu saat difilter oleh permintaan tarik.



Tambahkan pemicu pada push kode tanpa filter

Pemicu memungkinkan Anda mengonfigurasi pipeline untuk memulai pada jenis peristiwa tertentu, seperti permintaan push atau pull kode. Pemicu dapat dikonfigurasi untuk tindakan sumber dengan koneksi yang menggunakan CodeStarSourceConnection tindakan di CodePipeline, seperti, Bitbucket GitHub, dan. GitLab

Menambahkan pemicu (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda edit. Jika tidak, gunakan langkah-langkah ini pada wizard pembuatan pipeline.
3. Pada halaman detail pipeline, pilih Edit.
4. Pada halaman Edit, pilih tindakan sumber yang ingin Anda edit. Pilih Edit pemicu. Pilih untuk menambahkan pemicu.

5. Di Jenis pemicu, pilih Tidak ada filter.

Tambahkan pemicu dengan tipe acara permintaan push atau pull kode

Anda dapat mengonfigurasi filter untuk pemicu pipeline agar eksekusi pipeline dimulai untuk berbagai peristiwa Git, seperti push tag atau cabang, perubahan pada jalur file tertentu, permintaan tarik dibuka ke cabang tertentu, dan sebagainya. Anda dapat menggunakan AWS CodePipeline konsol atau update-pipeline perintah create-pipeline dan di AWS CLI untuk mengkonfigurasi filter pemicu.

> **Note**
>
> `BranchName`Bidang konfigurasi tindakan mendefinisikan satu cabang, sementara pemicu dengan filter dapat digunakan untuk cabang atau cabang apa pun yang Anda tentukan. Untuk pipeline di mana pemicu digunakan untuk memfilter cabang dengan permintaan push atau pull, pipeline tidak akan menggunakan cabang `BranchName` bidang default dalam konfigurasi tindakan. Namun, cabang di `BranchName` bidang dalam konfigurasi tindakan adalah default saat pipeline dimulai secara manual. Sebagai contoh, lihat 5: Pemicu dikonfigurasi sementara konfigurasi tindakan default BranchName digunakan untuk memulai manual.

Anda dapat menentukan filter untuk jenis pemicu berikut:

- Dorong

  Pemicu push memulai pipeline saat perubahan didorong ke repositori sumber Anda. Eksekusi akan menggunakan komit dari cabang yang Anda dorong (yaitu, cabang tujuan). Anda dapat memfilter pemicu push pada cabang, jalur file, atau tag Git.

- Permintaan tarik

  Pemicu permintaan tarik memulai pipeline saat permintaan tarik dibuka, diperbarui, atau ditutup di repositori sumber Anda. Eksekusi akan menggunakan komit dari cabang sumber yang Anda tarik (yaitu, cabang sumber). Anda dapat memfilter pemicu permintaan tarik pada cabang dan jalur file.

  Jenis acara yang didukung untuk permintaan tarik adalah sebagai berikut. Semua peristiwa permintaan tarik lainnya diabaikan.

- Dibuka
- Diperbarui
- Ditutup (digabung)

> **Note**
>
> Perilaku peristiwa permintaan tarik tertentu dapat berbeda menurut penyedia. Lihat perinciannya di Peristiwa permintaan tarik untuk pemicu oleh penyedia.

Definisi pipeline memungkinkan Anda untuk menggabungkan filter yang berbeda dalam konfigurasi pemicu push yang sama. Untuk detail tentang definisi pipeline, lihatTambahkan filter untuk jenis acara permintaan push dan pull (CLI). Untuk daftar definisi bidang, lihat pemicu dalam referensi struktur Pipeline dalam panduan ini.

Topik

- Tambahkan filter untuk jenis acara permintaan push dan pull (konsol)
- Tambahkan filter untuk jenis acara permintaan push dan pull (CLI)
- Tambahkan filter untuk jenis acara permintaan push dan pull (CloudFormation template)

Tambahkan filter untuk jenis acara permintaan push dan pull (konsol)

Anda dapat menggunakan konsol untuk menambahkan filter untuk acara push dan menyertakan atau mengecualikan cabang atau jalur file.

Tambahkan filter (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua pipeline yang terkait dengan AWS akun Anda akan ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda edit. Jika tidak, gunakan langkah-langkah ini pada wizard pembuatan pipeline.
3. Pada halaman detail pipeline, pilih Edit.
4. Pada halaman Edit, pilih tindakan sumber yang ingin Anda edit. Pilih Edit pemicu. Pilih Tentukan filter.

5. Dalam jenis Acara, pilih Push dari opsi berikut.

   - Pilih Push untuk memulai pipeline saat perubahan didorong ke repositori sumber Anda. Memilih ini memungkinkan bidang untuk menentukan filter untuk cabang dan jalur file atau tag Git.
   - Pilih Permintaan tarik untuk memulai pipeline saat permintaan tarik dibuka, diperbarui, atau ditutup di repositori sumber Anda. Memilih ini memungkinkan bidang untuk menentukan filter untuk cabang tujuan dan jalur file.

6. Di bawah Push, dalam jenis Filter, pilih salah satu opsi berikut.

   - Pilih Cabang untuk menentukan cabang di repositori sumber Anda yang dipantau pemicu untuk mengetahui kapan harus memulai alur kerja. Di Sertakan, masukkan pola untuk nama cabang dalam format glob yang ingin Anda tentukan untuk konfigurasi pemicu untuk memulai pipeline Anda pada perubahan di cabang yang ditentukan. Di Kecualikan, masukkan pola regex untuk nama cabang dalam format glob yang ingin Anda tentukan agar konfigurasi pemicu diabaikan dan untuk tidak memulai pipeline Anda pada perubahan di cabang yang ditentukan. Untuk informasi selengkapnya, lihat Bekerja dengan pola glob dalam sintaks.

     > **Note**
     >
     > Jika include dan exclude keduanya memiliki pola yang sama, maka defaultnya adalah mengecualikan pola.

     Anda dapat menggunakan pola glob untuk menentukan nama cabang Anda. Misalnya, gunakan `main*` untuk mencocokkan semua cabang yang dimulai dengan`main`. Untuk informasi selengkapnya, lihat Bekerja dengan pola glob dalam sintaks.

     Untuk pemicu dorong, tentukan cabang yang Anda dorong, yaitu cabang tujuan. Untuk pemicu permintaan tarik, tentukan cabang tujuan tempat Anda membuka permintaan tarik.

   - (Opsional) Di bawah Jalur file, tentukan jalur file untuk pemicu Anda. Masukkan nama di Sertakan dan Kecualikan sebagaimana mestinya.

     Anda dapat menggunakan pola glob untuk menentukan nama jalur file Anda. Misalnya, gunakan `prod*` untuk mencocokkan semua jalur file yang dimulai dengan`prod`. Untuk informasi selengkapnya, lihat Bekerja dengan pola glob dalam sintaks.

- Pilih Tag untuk mengonfigurasi konfigurasi pemicu pipeline untuk memulai dengan tag Git. Di Sertakan, masukkan pola untuk nama tag dalam format glob yang ingin Anda tentukan untuk konfigurasi pemicu untuk memulai pipeline Anda saat rilis tag atau tag yang ditentukan. Di Kecualikan, masukkan pola regex untuk nama tag dalam format glob yang ingin Anda tentukan agar konfigurasi pemicu diabaikan dan untuk tidak memulai pipeline Anda saat rilis tag atau tag yang ditentukan. Jika include dan exclude keduanya memiliki pola tag yang sama, maka defaultnya adalah mengecualikan pola tag.

7. Di bawah Push, dalam jenis Filter, pilih salah satu opsi berikut.

- Pilih Cabang untuk menentukan cabang di repositori sumber Anda yang dipantau pemicu untuk mengetahui kapan harus memulai alur kerja. Di Sertakan, masukkan pola untuk nama cabang dalam format glob yang ingin Anda tentukan untuk konfigurasi pemicu untuk memulai pipeline Anda pada perubahan di cabang yang ditentukan. Di Kecualikan, masukkan pola regex untuk nama cabang dalam format glob yang ingin Anda tentukan agar konfigurasi pemicu diabaikan dan untuk tidak memulai pipeline Anda pada perubahan di cabang yang ditentukan. Untuk informasi selengkapnya, lihat Bekerja dengan pola glob dalam sintaks.

  > **Note**
  >
  > Jika include dan exclude keduanya memiliki pola yang sama, maka defaultnya adalah mengecualikan pola.

  Anda dapat menggunakan pola glob untuk menentukan nama cabang Anda. Misalnya, gunakan `main*` untuk mencocokkan semua cabang yang dimulai dengan`main`. Untuk informasi selengkapnya, lihat Bekerja dengan pola glob dalam sintaks.

  Untuk pemicu dorong, tentukan cabang yang Anda dorong, yaitu cabang tujuan. Untuk pemicu permintaan tarik, tentukan cabang tujuan tempat Anda membuka permintaan tarik.

- (Opsional) Di bawah Jalur file, tentukan jalur file untuk pemicu Anda. Masukkan nama di Sertakan dan Kecualikan sebagaimana mestinya.

  Anda dapat menggunakan pola glob untuk menentukan nama jalur file Anda. Misalnya, gunakan `prod*` untuk mencocokkan semua jalur file yang dimulai dengan`prod`. Untuk informasi selengkapnya, lihat Bekerja dengan pola glob dalam sintaks.

- Pilih Tarik permintaan untuk mengonfigurasi konfigurasi pemicu pipeline untuk memulai dengan peristiwa permintaan tarik yang Anda tentukan.

Tambahkan filter untuk jenis acara permintaan push dan pull (CLI)

Anda dapat memperbarui pipeline JSON untuk menambahkan filter untuk pemicu.

Untuk menggunakan AWS CLI untuk membuat atau memperbarui pipeline Anda, gunakan `update-pipeline` perintah `create-pipeline` or.

Contoh berikut struktur JSON memberikan referensi untuk definisi bidang di bawah`create-pipeline`.

Untuk daftar definisi bidang, lihat pemicu dalam referensi struktur Pipeline dalam panduan ini.

```
{
    "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"
                        }
                    }
                ]
            }
        ]
    }
}
```

Tambahkan filter untuk jenis acara permintaan push dan pull (CloudFormation template)

Anda dapat memperbarui sumber daya pipeline CloudFormation untuk menambahkan pemfilteran pemicu.

Contoh cuplikan template berikut menyediakan referensi YAMB untuk memicu definisi bidang. Untuk daftar definisi bidang, lihat pemicu dalam referensi struktur Pipeline dalam panduan ini.

Untuk contoh templat lengkap untuk sumber koneksi dan konfigurasi filter pemicu, lihat Pipeline dengan dua tahap dan konfigurasi pemicu di Panduan CloudFormation Pengguna.

```
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
```

Tambahkan pemicu untuk mematikan deteksi perubahan

Pemicu memungkinkan Anda mengonfigurasi pipeline untuk memulai pada jenis peristiwa tertentu, seperti permintaan push atau pull kode. Pemicu dapat dikonfigurasi untuk tindakan sumber dengan koneksi yang menggunakan `CodeStarSourceConnection` tindakan di CodePipeline, seperti, Bitbucket GitHub, dan. GitLab

Menambahkan pemicu untuk mematikan deteksi perubahan (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda edit. Jika tidak, gunakan langkah-langkah ini pada wizard pembuatan pipeline.
3. Pada halaman detail pipeline, pilih Edit.
4. Pada halaman Edit, pilih tindakan sumber yang ingin Anda edit. Pilih Edit pemicu. Pilih untuk menambahkan pemicu.
5. Di Jenis pemicu, pilih Jangan mendeteksi perubahan.

Mulai dan hentikan jaringan pipa secara manual

Anda dapat menggunakan AWS CodePipeline konsol atau AWS CLI untuk memulai dan menghentikan saluran pipa secara manual. Ada metode otomatis untuk memulai saluran pipa dengan deteksi perubahan berbasis peristiwa seperti yang dijelaskan dalam, atau konfigurasi pemicu seperti yang dijelaskan dalam.

Topik

- Mulai pipa di CodePipeline
- Hentikan eksekusi pipeline di CodePipeline

Mulai pipa di CodePipeline

Setiap eksekusi pipeline dapat dimulai berdasarkan pemicu yang berbeda. Setiap eksekusi pipeline dapat memiliki jenis pemicu yang berbeda, tergantung pada bagaimana pipeline dimulai. Jenis pemicu untuk setiap eksekusi ditampilkan dalam riwayat eksekusi untuk pipeline. Jenis pemicu dapat bergantung pada penyedia tindakan sumber sebagai berikut:

> **Note**
>
> Anda tidak dapat menentukan lebih dari satu pemicu per tindakan sumber.

- Pembuatan pipa: Saat pipeline dibuat, eksekusi pipeline dimulai secara otomatis. Ini adalah tipe `CreatePipeline` pemicu dalam riwayat Eksekusi.
- Perubahan pada objek yang direvisi: Kategori ini mewakili jenis `PutActionRevision` pemicu dalam riwayat Eksekusi.
- Ubah deteksi pada cabang dan komit untuk push kode: Kategori ini mewakili tipe `CloudWatchEvent` pemicu dalam riwayat Eksekusi. Ketika perubahan terdeteksi ke komit sumber dan cabang di repositori sumber, pipeline Anda dimulai. Jenis pemicu ini menggunakan deteksi perubahan otomatis. Penyedia aksi sumber yang menggunakan tipe pemicu ini adalah S3 dan CodeCommit. Jenis ini juga digunakan untuk jadwal yang memulai pipeline Anda. Lihat Mulai pipeline sesuai jadwal.
- Polling untuk perubahan sumber: Kategori ini mewakili jenis `PollForSourceChanges` pemicu dalam riwayat Eksekusi. Ketika perubahan terdeteksi ke komit sumber dan cabang di repositori

sumber melalui polling, pipeline Anda dimulai. Jenis pemicu ini tidak disarankan dan harus dimigrasikan untuk menggunakan deteksi perubahan otomatis. Penyedia aksi sumber yang menggunakan tipe pemicu ini adalah S3 dan CodeCommit.

- Peristiwa webhook untuk sumber pihak ketiga: Kategori ini mewakili jenis `Webhook` pemicu dalam riwayat Eksekusi. Ketika perubahan terdeteksi oleh peristiwa webhook, pipeline Anda dimulai. Jenis pemicu ini menggunakan deteksi perubahan otomatis. Penyedia tindakan sumber yang menggunakan tipe pemicu ini adalah koneksi yang dikonfigurasi untuk push kode (Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab .com, dan GitLab dikelola sendiri).
- Peristiwa WebHookv2 untuk sumber pihak ketiga: Kategori ini mewakili jenis **`WebhookV2`** pemicu dalam riwayat Eksekusi. Tipe ini untuk eksekusi yang dipicu berdasarkan pemicu yang ditentukan dalam definisi pipeline. Ketika rilis dengan tag Git tertentu terdeteksi, pipeline Anda dimulai. Anda dapat menggunakan tag Git untuk menandai komit dengan nama atau pengenal lain yang membantu pengguna repositori lain memahami pentingnya komit tersebut. Anda juga dapat menggunakan tag Git untuk mengidentifikasi komit tertentu dalam riwayat repositori. Jenis pemicu ini menonaktifkan deteksi perubahan otomatis. Penyedia aksi sumber yang menggunakan tipe pemicu ini adalah koneksi yang dikonfigurasi untuk tag Git (Bitbucket Cloud, GitHub, GitHub Enterprise Server, GitLab dan.com).
- Memulai pipeline secara manual: Kategori ini mewakili tipe `StartPipelineExecution` pemicu dalam riwayat Eksekusi. Anda dapat menggunakan konsol atau AWS CLI untuk memulai pipeline secara manual. Untuk informasi, lihat Mulai pipa secara manual.
- RollbackStage: Kategori ini mewakili jenis `RollbackStage` pemicu dalam riwayat Eksekusi. Anda dapat menggunakan konsol atau AWS CLI untuk memutar kembali panggung secara manual atau otomatis. Untuk informasi, lihat Mengkonfigurasi rollback panggung.

Saat Anda menambahkan tindakan sumber ke pipeline yang menggunakan jenis pemicu deteksi perubahan otomatis, tindakan akan berfungsi dengan sumber daya tambahan. Membuat setiap tindakan sumber dirinci dalam bagian terpisah karena sumber daya tambahan ini untuk deteksi perubahan. Untuk detail tentang setiap penyedia sumber dan metode deteksi perubahan yang diperlukan untuk deteksi perubahan otomatis, lihatTindakan sumber dan metode deteksi perubahan.

Topik

- Mulai pipa secara manual
- Mulai pipeline sesuai jadwal
- Mulai pipeline dengan penggantian revisi sumber

Mulai pipa secara manual

Secara default, pipeline dimulai secara otomatis ketika dibuat dan setiap kali perubahan dibuat di repositori sumber. Namun, Anda mungkin ingin menjalankan kembali revisi terbaru melalui pipeline untuk kedua kalinya. Anda dapat menggunakan CodePipeline konsol atau start-pipeline-execution perintah AWS CLI and untuk menjalankan kembali revisi terbaru secara manual melalui pipeline Anda.

Topik

- Mulai pipeline secara manual (konsol)
- Mulai pipa secara manual (CLI)

Mulai pipeline secara manual (konsol)

Untuk memulai pipeline secara manual dan menjalankan revisi terbaru melalui pipeline

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Di Nama, pilih nama pipeline yang ingin Anda mulai.
3. Pada halaman detail pipeline, pilih Rilis perubahan. Jika pipeline dikonfigurasi untuk meneruskan parameter (variabel pipeline), maka memilih Release change membuka jendela Release change. Dalam variabel Pipeline, di bidang atau bidang untuk variabel di tingkat pipeline, masukkan nilai atau nilai yang ingin Anda lewati untuk eksekusi pipeline ini. Untuk informasi selengkapnya, lihat Referensi variabel.

   Ini memulai revisi terbaru yang tersedia di setiap lokasi sumber yang ditentukan dalam aksi sumber melalui pipeline.

Mulai pipa secara manual (CLI)

Untuk memulai pipeline secara manual dan menjalankan versi artefak terbaru melalui pipa

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan start-pipeline-execution perintah, dengan menentukan nama pipeline yang ingin Anda mulai. Misalnya, untuk mulai menjalankan perubahan terakhir melalui pipeline bernama*MyFirstPipeline*:

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

Untuk memulai pipeline di mana variabel dikonfigurasi pada tingkat pipeline, gunakan start-pipeline-execution perintah dengan --variables argumen opsional untuk memulai pipeline dan menambahkan variabel yang akan digunakan dalam eksekusi. Misalnya, untuk menambahkan variabel `var1` dengan nilai1, gunakan perintah berikut:

```
aws codepipeline start-pipeline-execution --name MyFirstPipeline --variables
 name=var1,value=1
```

2. Untuk memverifikasi keberhasilan, lihat objek yang dikembalikan. Perintah ini mengembalikan ID eksekusi, mirip dengan berikut ini:

```
{
    "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
}
```

> **Note**
>
> Setelah Anda memulai pipeline, Anda dapat memantau kemajuannya di CodePipeline konsol atau dengan menjalankan get-pipeline-state perintah. Untuk informasi selengkapnya, lihat Lihat saluran pipa (konsol) dan Lihat detail dan riwayat saluran pipa (CLI).

Mulai pipeline sesuai jadwal

Anda dapat mengatur aturan EventBridge untuk memulai pipeline sesuai jadwal.

Buat EventBridge aturan yang menjadwalkan pipeline Anda untuk memulai (konsol)

Untuk membuat EventBridge aturan dengan jadwal sebagai sumber acara

1. Buka EventBridge konsol Amazon di https://console.aws.amazon.com/events/.
2. Di panel navigasi, pilih Aturan.
3. Pilih Buat aturan, lalu di bawah Rincian aturan, pilih Jadwal.

4. Atur jadwal menggunakan tingkat atau ekspresi tetap. Untuk selengkapnya, lihat Menjadwalkan Ekspresi untuk Aturan.
5. Di Target, pilih CodePipeline.
6. Masukkan ARN pipa untuk eksekusi pipa untuk jadwal ini.

   > **Note**
   >
   > Anda dapat menemukan pipa ARN di bawah Pengaturan di konsol. Lihat Lihat ARN pipeline dan peran layanan ARN (konsol).

7. Pilih salah satu dari berikut ini untuk membuat atau menentukan peran layanan IAM yang memberikan EventBridge izin untuk memanggil target yang terkait dengan EventBridge aturan Anda (dalam hal ini, targetnya adalah). CodePipeline

   - Pilih Buat peran baru untuk sumber daya khusus ini untuk membuat peran layanan yang memberikan EventBridge izin untuk memulai eksekusi pipeline Anda.
   - Pilih Gunakan peran yang ada untuk memasukkan peran layanan yang memberikan EventBridge izin untuk memulai eksekusi pipeline Anda.
8. Pilih Konfigurasikan detail.
9. Pada halaman Konfigurasi detail aturan, masukkan nama dan deskripsi untuk aturan, lalu pilih Status untuk mengaktifkan aturan.
10. Jika Anda puas dengan aturan, pilih Create rule (Buat aturan).

Buat EventBridge aturan yang menjadwalkan pipeline Anda untuk memulai (CLI)

Untuk menggunakan AWS CLI untuk membuat aturan, panggil put-rule perintah, tentukan:

- Nama yang secara unik mengidentifikasi aturan yang Anda buat. Nama ini harus unik di semua pipeline yang Anda buat CodePipeline terkait dengan AWS akun Anda.
- Ekspresi jadwal untuk aturan.

Untuk membuat EventBridge aturan dengan jadwal sebagai sumber acara

1. Panggil put-rule perintah dan sertakan `--schedule-expression` parameter `--name` dan.

   Contoh:

Contoh perintah berikut digunakan --schedule-expression untuk membuat aturan `MyRule2` yang disebut filter EventBridge pada jadwal.

```
aws events put-rule --schedule-expression 'cron(15 10 ? * 6L 2002-2005)' --name
 MyRule2
```

2. Untuk menambahkan CodePipeline sebagai target, panggil put-targets perintah dan sertakan parameter berikut:

   - `--rule`Parameter digunakan dengan yang `rule_name` Anda buat dengan menggunakanput-rule.
   - `--targets`Parameter digunakan dengan `Id` daftar target dalam daftar target dan pipa target. ARN

   Contoh perintah berikut menentukan bahwa untuk aturan yang dipanggil`MyCodeCommitRepoRule`, target `Id` terdiri dari nomor satu, menunjukkan bahwa dalam daftar target untuk aturan, ini adalah target 1. Perintah sample juga menentukan contoh ARN untuk pipeline. Pipeline dimulai ketika sesuatu berubah di repositori.

   ```
   aws events put-targets --rule MyCodeCommitRepoRule --targets
    Id=1,Arn=arn:aws:codepipeline:us-west-2:80398EXAMPLE:TestPipeline
   ```

3. Berikan izin EventBridge untuk digunakan CodePipeline untuk menjalankan aturan. Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis sumber daya untuk Amazon. EventBridge

   a. Gunakan contoh berikut untuk membuat kebijakan kepercayaan agar memungkinkan EventBridge untuk mengambil peran layanan. Nama itu`trustpolicyforEB.json`.

      JSON

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Principal": {
                      "Service": "events.amazonaws.com"
      ```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Gunakan perintah berikut untuk membuat `Role-for-MyRule` peran dan melampirkan kebijakan kepercayaan.

```
aws iam create-role --role-name Role-for-MyRule --assume-role-policy-document
 file://trustpolicyforEB.json
```

c. Buat kebijakan izin JSON seperti yang ditunjukkan dalam contoh ini untuk pipeline bernama. `MyFirstPipeline` Beri nama kebijakan `permissionspolicyforEB.json` izin.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:StartPipelineExecution"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111122223333:MyFirstPipeline"
            ]
        }
    ]
}
```

d. Gunakan perintah berikut untuk melampirkan kebijakan `CodePipeline-Permissions-Policy-for-EB` izin baru ke `Role-for-MyRule` peran yang Anda buat.

```
aws iam put-role-policy --role-name Role-for-MyRule --policy-name CodePipeline-
Permissions-Policy-For-EB --policy-document file://permissionspolicyforCWE.json
```

Mulai pipeline dengan penggantian revisi sumber

Anda dapat menggunakan penggantian untuk memulai pipeline dengan ID revisi sumber tertentu yang Anda berikan untuk eksekusi pipeline. Misalnya, jika Anda ingin memulai pipeline yang akan memproses ID komit tertentu dari CodeCommit sumber Anda, Anda dapat menambahkan ID komit sebagai penggantian saat memulai pipeline.

> **Note**
>
> Anda juga dapat membuat penggantian sumber menggunakan entri transformasi input untuk menggunakan `revisionValue` in EventBridge untuk acara pipeline Anda, di mana turunan dari variabel peristiwa sumber untuk kunci objek, komit, atau ID gambar Anda. `revisionValue` Untuk informasi selengkapnya, lihat langkah opsional untuk entri transformasi masukan yang disertakan dalam prosedur di bawahTindakan sumber dan sumber daya Amazon ECR EventBridge,Menyambungkan ke tindakan sumber Amazon S3 dengan sumber yang diaktifkan untuk acara, atauCodeCommit tindakan sumber dan EventBridge.

Ada empat jenis revisi sumber untuk`revisionType`:

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

> **Note**
>
> Untuk `COMMIT_ID` dan `IMAGE_DIGEST` jenis revisi sumber, ID revisi sumber berlaku untuk semua konten dalam repositori, di semua cabang.

> **Note**
>
> Untuk `S3_OBJECT_VERSION_ID` dan `S3_OBJECT_KEY` jenis revisi sumber, salah satu tipe dapat digunakan secara independen, atau dapat digunakan bersama untuk mengganti

sumber dengan spesifik ObjectKey dan versionId. Untuk`S3_OBJECT_KEY`, parameter konfigurasi `AllowOverrideForS3ObjectKey` perlu diatur ke`true`. Untuk informasi selengkapnya tentang parameter konfigurasi sumber S3, lihatParameter konfigurasi.

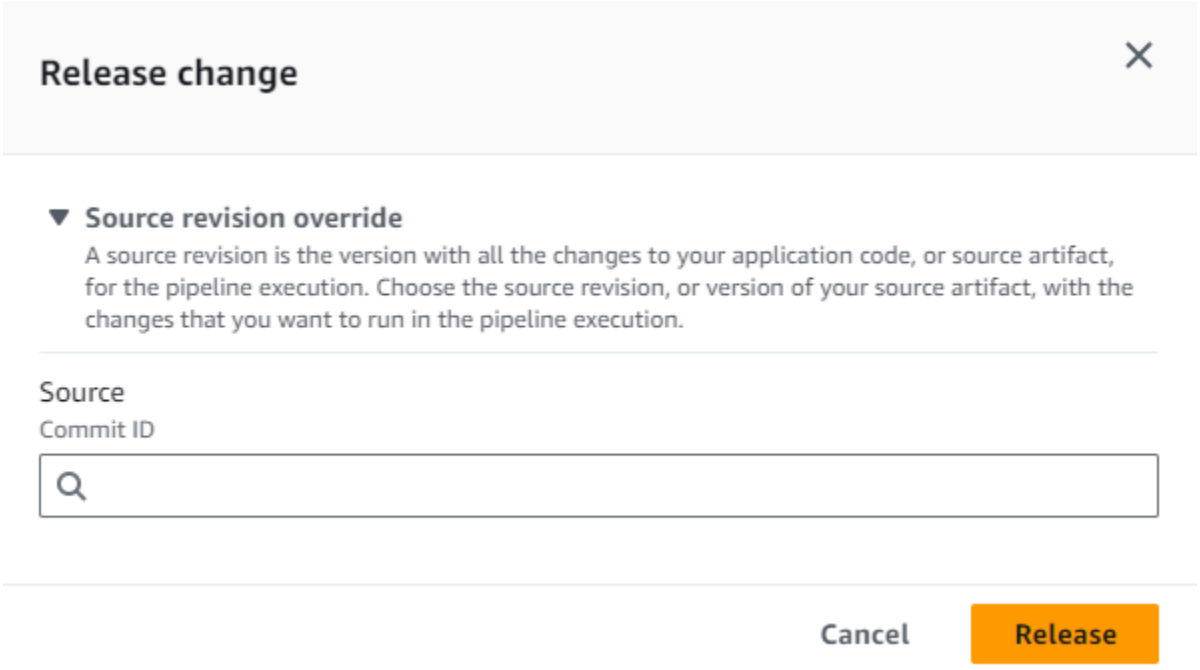
Topik

- Mulai pipeline dengan penggantian revisi sumber (konsol)
- Mulai pipeline dengan penggantian revisi sumber (CLI)

Mulai pipeline dengan penggantian revisi sumber (konsol)

Untuk memulai pipeline secara manual dan menjalankan revisi terbaru melalui pipeline

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Di Nama, pilih nama pipeline yang ingin Anda mulai.
3. Pada halaman detail pipeline, pilih Rilis perubahan. Memilih Perubahan rilis membuka jendela Rilis perubahan. Untuk penggantian revisi Sumber, pilih panah untuk memperluas bidang. Di Sumber, masukkan ID revisi sumber. Misalnya, jika pipeline Anda memiliki CodeCommit sumber, pilih ID komit dari bidang yang ingin Anda gunakan.



Mulai pipeline dengan penggantian revisi sumber (CLI)

Untuk memulai pipeline secara manual dan menjalankan ID revisi sumber yang ditentukan untuk artefak melalui pipa

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan start-pipeline-execution perintah, dengan menentukan nama pipeline yang ingin Anda mulai. Anda juga menggunakan --source-revisions argumen untuk memberikan ID revisi sumber. Revisi sumber terdiri dari ActionName, revisionType, dan revisionValue. Nilai RevisionType yang valid adalah. `COMMIT_ID | IMAGE_DIGEST | S3_OBJECT_VERSION_ID | S3_OBJECT_KEY`

   Dalam contoh berikut, untuk mulai menjalankan perubahan yang ditentukan melalui pipeline bernamacodecommit-pipeline, perintah berikut menspesifikasikan nama tindakan sumber Sumber, jenis revisi`COMMIT_ID`, dan ID komit dari`78a25c18755ccac3f2a9eec099dEXAMPLE`.

   ```
   aws codepipeline start-pipeline-execution --name codecommit-pipeline --source-
   revisions
    actionName=Source,revisionType=COMMIT_ID,revisionValue=78a25c18755ccac3f2a9eec099dEXAMPLE
    --region us-west-1
   ```

2. Untuk memverifikasi keberhasilan, lihat objek yang dikembalikan. Perintah ini mengembalikan ID eksekusi, mirip dengan berikut ini:

   ```
   {
       "pipelineExecutionId": "c53dbd42-This-Is-An-Example"
   }
   ```

   > **Note**
   >
   > Setelah Anda memulai pipeline, Anda dapat memantau kemajuannya di CodePipeline konsol atau dengan menjalankan get-pipeline-state perintah. Lihat informasi yang lebih lengkap di Lihat saluran pipa (konsol) dan Lihat detail dan riwayat saluran pipa (CLI).

Hentikan eksekusi pipeline di CodePipeline

Ketika eksekusi pipeline mulai berjalan melalui pipeline, ia memasuki satu tahap pada satu waktu dan mengunci panggung sementara semua eksekusi aksi di panggung sedang berjalan. Tindakan

yang sedang berlangsung ini harus ditangani sedemikian rupa sehingga, ketika eksekusi pipeline dihentikan, tindakan tersebut diizinkan untuk diselesaikan atau ditinggalkan.

Ada dua cara untuk menghentikan eksekusi pipeline:

- Berhenti dan tunggu: AWS CodePipeline menunggu untuk menghentikan eksekusi sampai semua tindakan yang sedang berlangsung selesai (yaitu, tindakan memiliki `Failed` status `Succeeded` atau). Opsi ini mempertahankan tindakan yang sedang berlangsung. Eksekusi dalam `Stopping` keadaan sampai tindakan yang sedang berlangsung selesai. Kemudian eksekusi dalam `Stopped` keadaan. Panggung terbuka setelah tindakan selesai.

  Jika Anda memilih untuk berhenti dan menunggu, dan Anda berubah pikiran saat eksekusi Anda masih dalam `Stopping` keadaan, Anda kemudian dapat memilih untuk meninggalkannya.
- Berhenti dan tinggalkan: AWS CodePipeline menghentikan eksekusi tanpa menunggu tindakan yang sedang berlangsung selesai. Eksekusi dalam `Stopping` keadaan untuk waktu yang sangat singkat sementara tindakan yang sedang berlangsung ditinggalkan. Setelah eksekusi dihentikan, eksekusi tindakan dalam `Abandoned` keadaan sementara eksekusi pipeline dalam `Stopped` keadaan. Panggung terbuka.

  Untuk eksekusi pipeline dalam `Stopped` keadaan, tindakan pada tahap di mana eksekusi dihentikan dapat dicoba lagi.

  > **Warning**
  >
  > Opsi ini dapat menyebabkan tugas gagal atau tugas di luar urutan.

Topik

- Hentikan eksekusi pipeline (konsol)
- Menghentikan Eksekusi Masuk (Konsol)
- Hentikan eksekusi pipeline (CLI)
- Hentikan Eksekusi Masuk (CLI)

Hentikan eksekusi pipeline (konsol)

Anda dapat menggunakan konsol untuk menghentikan eksekusi pipeline. Pilih eksekusi, lalu pilih metode untuk menghentikan eksekusi pipeline.

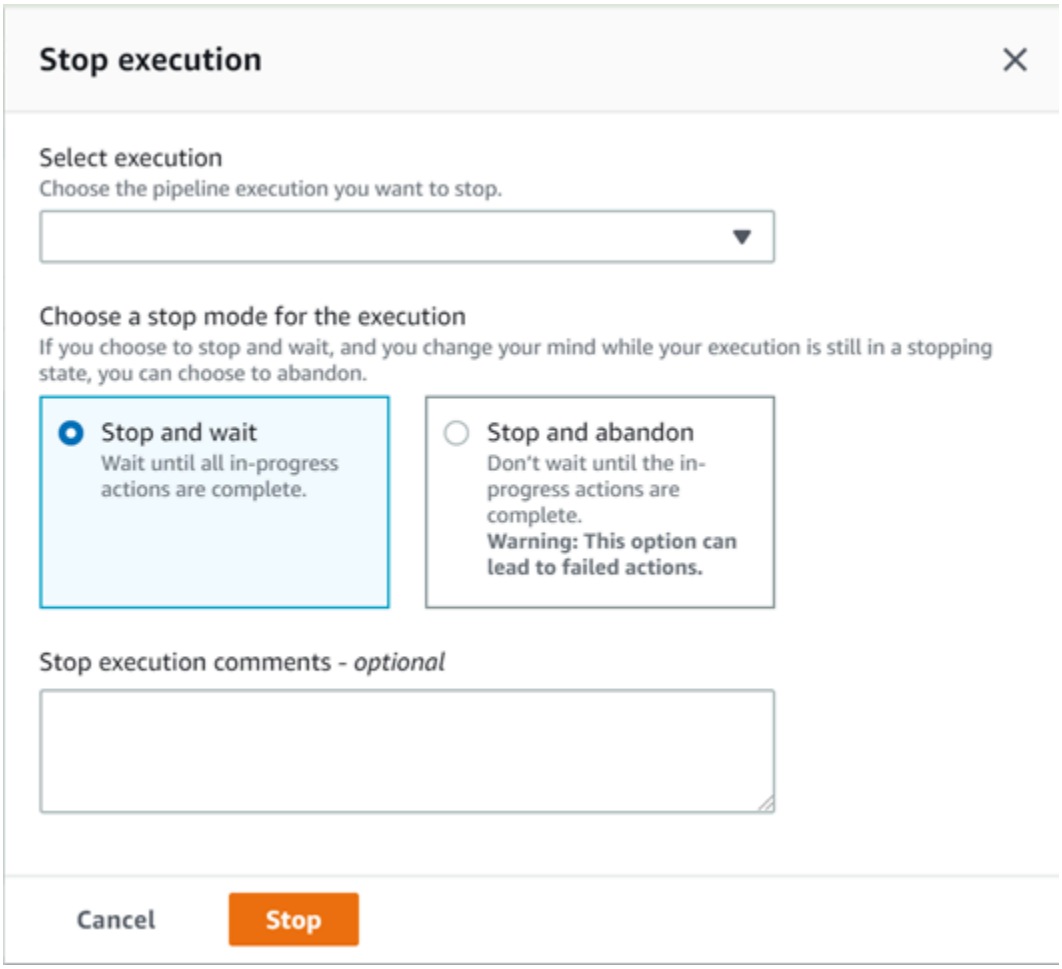
> **Note**
>
> Anda juga dapat menghentikan eksekusi pipeline yang merupakan eksekusi masuk. Untuk mempelajari lebih lanjut tentang menghentikan eksekusi masuk, lihatMenghentikan Eksekusi Masuk (Konsol).

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Lakukan salah satu tindakan berikut:

   > **Note**
   >
   > Sebelum Anda menghentikan eksekusi, kami sarankan Anda menonaktifkan transisi di depan panggung. Dengan cara ini, ketika panggung terbuka karena eksekusi yang dihentikan, tahap tidak menerima eksekusi pipa berikutnya.

   - Di Nama, pilih nama pipeline dengan eksekusi yang ingin Anda hentikan. Pada halaman detail pipeline, pilih Hentikan eksekusi.
   - Pilih Lihat riwayat. Pada halaman riwayat, pilih Hentikan eksekusi.
3. Pada halaman Stop execution, di bawah Pilih eksekusi, pilih eksekusi yang ingin Anda hentikan.

   > **Note**
   >
   > Eksekusi ditampilkan hanya jika masih dalam proses. Eksekusi yang sudah selesai tidak ditampilkan.



4. Di bawah Pilih tindakan yang akan diterapkan pada eksekusi, pilih salah satu dari berikut ini:

   - Untuk memastikan eksekusi tidak berhenti sampai semua tindakan yang sedang berlangsung selesai, pilih Berhenti dan tunggu.

     > **Note**
     >
     > Anda tidak dapat memilih untuk berhenti dan menunggu jika eksekusi sudah dalam keadaan berhenti, tetapi Anda dapat memilih untuk berhenti dan meninggalkan.

   - Untuk berhenti tanpa menunggu tindakan yang sedang berlangsung selesai, pilih Berhenti dan tinggalkan.

> **Warning**
>
> Opsi ini dapat menyebabkan tugas gagal atau tugas di luar urutan.

5. (Opsional) Masukkan komentar. Komentar ini, bersama dengan status eksekusi, ditampilkan di halaman riwayat untuk eksekusi.
6. Pilih Berhenti.

> **Important**
>
> Tindakan ini tidak dapat dibatalkan.

7. Lihat status eksekusi dalam visualisasi pipeline sebagai berikut:

   - Jika Anda memilih untuk berhenti dan menunggu, eksekusi yang dipilih akan berlanjut hingga tindakan yang sedang berlangsung selesai.
     - Pesan spanduk sukses ditampilkan di bagian atas konsol.
     - Pada tahap saat ini, tindakan yang sedang berlangsung berlanjut dalam suatu `InProgress` keadaan. Sementara tindakan sedang berlangsung, eksekusi pipeline dalam `Stopping` keadaan.

       Setelah tindakan selesai (yaitu, tindakan gagal atau berhasil), eksekusi pipeline berubah menjadi `Stopped` status dan tindakan berubah menjadi `Succeeded` status `Failed` atau. Anda juga dapat melihat status tindakan pada halaman detail eksekusi. Anda dapat melihat status eksekusi pada halaman riwayat eksekusi atau halaman detail eksekusi.
     - Eksekusi pipeline berubah menjadi `Stopping` status secara singkat, dan kemudian berubah menjadi `Stopped` keadaan. Anda dapat melihat status eksekusi pada halaman riwayat eksekusi atau halaman detail eksekusi.
   - Jika Anda memilih untuk berhenti dan meninggalkan, eksekusi tidak menunggu tindakan yang sedang berlangsung selesai.
     - Pesan spanduk sukses ditampilkan di bagian atas konsol.
     - Pada tahap saat ini, tindakan dalam proses berubah menjadi status. `Abandoned` Anda juga dapat melihat status tindakan di halaman detail eksekusi.

- Eksekusi pipeline berubah menjadi `Stopping` status secara singkat, dan kemudian berubah menjadi `Stopped` keadaan. Anda dapat melihat status eksekusi pada halaman riwayat eksekusi atau halaman detail eksekusi.

Anda dapat melihat status eksekusi pipeline dalam tampilan riwayat eksekusi dan tampilan riwayat terperinci.

Menghentikan Eksekusi Masuk (Konsol)

Anda dapat menggunakan konsol untuk menghentikan eksekusi masuk. Eksekusi inbound adalah eksekusi pipeline yang menunggu untuk memasuki tahap di mana transisi telah dinonaktifkan. Ketika transisi diaktifkan, eksekusi inbound yang `InProgress` terus memasuki tahap. Eksekusi inbound yang `Stopped` tidak memasuki tahap.

> **Note**
>
> Setelah eksekusi inbound dihentikan, itu tidak dapat dicoba lagi.

Jika Anda tidak melihat eksekusi masuk, maka tidak ada eksekusi yang tertunda pada transisi tahap dinonaktifkan.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda akan ditampilkan.
2. Pilih nama pipeline yang ingin Anda hentikan eksekusi masuk, Lakukan salah satu hal berikut:

   - Dalam tampilan Pipeline, pilih ID eksekusi masuk dan kemudian pilih untuk menghentikan eksekusi.
   - Pilih pipeline dan pilih Lihat riwayat. Dalam riwayat eksekusi, pilih ID eksekusi masuk dan kemudian pilih untuk menghentikan eksekusi.
3. Dalam modal Stop execution, ikuti langkah-langkah di bagian di atas untuk memilih ID eksekusi dan tentukan metode stop.

   Gunakan get-pipeline-state perintah untuk melihat status eksekusi masuk.

Hentikan eksekusi pipeline (CLI)

Untuk menggunakan AWS CLI untuk menghentikan pipa secara manual, gunakan stop-pipeline-execution perintah dengan parameter berikut:

- ID Eksekusi (wajib)
- Komentar (opsional)
- Nama pipa (wajib)
- Abaikan bendera (opsional, defaultnya salah)

Format perintah:

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --pipeline-
execution-id Execution_ID [--abandon | --no-abandon] [--reason STOP_EXECUTION_REASON]
```

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows).
2. Untuk menghentikan eksekusi pipeline, pilih salah satu dari berikut ini:

   - Untuk memastikan eksekusi tidak berhenti sampai semua tindakan yang sedang berlangsung selesai, pilih untuk berhenti dan menunggu. Anda dapat melakukan ini dengan memasukkan no-abandon parameter. Jika Anda tidak menentukan parameter, perintah default untuk berhenti dan menunggu. Gunakan AWS CLI untuk menjalankan stop-pipeline-execution perintah, menentukan nama pipeline dan ID eksekusi. Misalnya, untuk menghentikan pipeline bernama *MyFirstPipeline* dengan opsi berhenti dan tunggu yang ditentukan:

     ```
     aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
     pipeline-execution-id d-EXAMPLE --no-abandon
     ```

     Misalnya, untuk menghentikan pipeline bernama*MyFirstPipeline*, default ke opsi berhenti dan tunggu, dan memilih untuk menyertakan komentar:

     ```
     aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
     pipeline-execution-id d-EXAMPLE --reason "Stopping execution after the build
      action is done"
     ```

> **Note**
>
> Anda tidak dapat memilih untuk berhenti dan menunggu jika eksekusi sudah dalam status Berhenti. Anda dapat memilih untuk berhenti dan meninggalkan eksekusi yang sudah dalam status Berhenti.

- Untuk berhenti tanpa menunggu tindakan yang sedang berlangsung selesai, pilih untuk berhenti dan tinggalkan. Sertakan abandon parameternya. Gunakan AWS CLI untuk menjalankan stop-pipeline-execution perintah, menentukan nama pipeline dan ID eksekusi.

  Misalnya, untuk menghentikan pipeline bernama*MyFirstPipeline*, menentukan opsi abaikan, dan memilih untuk menyertakan komentar:

```
aws codepipeline stop-pipeline-execution --pipeline-name MyFirstPipeline --
pipeline-execution-id d-EXAMPLE --abandon --reason "Stopping execution for a bug
 fix"
```

Hentikan Eksekusi Masuk (CLI)

Anda dapat menggunakan CLI untuk menghentikan eksekusi masuk. Eksekusi inbound adalah eksekusi pipeline yang menunggu untuk memasuki tahap di mana transisi telah dinonaktifkan. Ketika transisi diaktifkan, eksekusi inbound yang `InProgress` terus memasuki tahap. Eksekusi inbound yang `Stopped` tidak memasuki tahap.

> **Note**
>
> Setelah eksekusi inbound dihentikan, itu tidak dapat dicoba lagi.

Jika Anda tidak melihat eksekusi masuk, maka tidak ada eksekusi yang tertunda pada transisi tahap dinonaktifkan.

Untuk menggunakan AWS CLI untuk menghentikan eksekusi masuk secara manual, gunakan stop-pipeline-execution perintah dengan parameter berikut:

- ID Eksekusi Masuk (wajib)
- Komentar (opsional)

- Nama pipa (wajib)
- Abaikan bendera (opsional, defaultnya salah)

Format perintah:

```
aws codepipeline stop-pipeline-execution --pipeline-name Pipeline_Name --
pipeline-execution-id Inbound_Execution_ID [--abandon | --no-abandon] [--
reason STOP_EXECUTION_REASON]
```

Ikuti langkah-langkah dalam prosedur di atas untuk memasukkan perintah dan tentukan metode berhenti.

Gunakan get-pipeline-state perintah untuk melihat status eksekusi masuk.

Lihat riwayat dan atur mode untuk eksekusi pipeline

Untuk menganalisis kemajuan pipeline, Anda dapat melihat alur pipa dan riwayat eksekusi tindakan. Untuk mengkonfigurasi metode rilis untuk memproses eksekusi pipeline, atur mode eksekusi.

Topik

- Lihat eksekusi di CodePipeline
- Mengatur atau mengubah mode eksekusi pipa

Lihat eksekusi di CodePipeline

Anda dapat menggunakan AWS CodePipeline konsol atau AWS CLI untuk melihat status eksekusi, melihat riwayat eksekusi, dan mencoba lagi tahapan atau tindakan yang gagal.

Topik

- Lihat riwayat eksekusi pipeline (konsol)
- Lihat status eksekusi (konsol)
- Melihat eksekusi masuk (Konsol)
- Lihat revisi sumber eksekusi pipeline (konsol)
- Lihat eksekusi tindakan (konsol)
- Lihat artefak aksi dan informasi penyimpanan artefak (konsol)
- Lihat detail dan riwayat saluran pipa (CLI)

Lihat riwayat eksekusi pipeline (konsol)

Anda dapat menggunakan CodePipeline konsol untuk melihat daftar semua pipeline di akun Anda. Anda juga dapat melihat detail untuk setiap pipeline, termasuk kapan tindakan terakhir dijalankan di pipeline, apakah transisi antar tahapan diaktifkan atau dinonaktifkan, apakah ada tindakan yang gagal, dan informasi lainnya. Anda juga dapat melihat halaman riwayat yang menampilkan detail untuk semua eksekusi pipeline yang histori telah direkam.

> **Note**
>
> Saat beralih di antara mode eksekusi tertentu, tampilan dan riwayat pipeline mungkin berubah. Untuk informasi selengkapnya, lihat Mengatur atau mengubah mode eksekusi pipa.

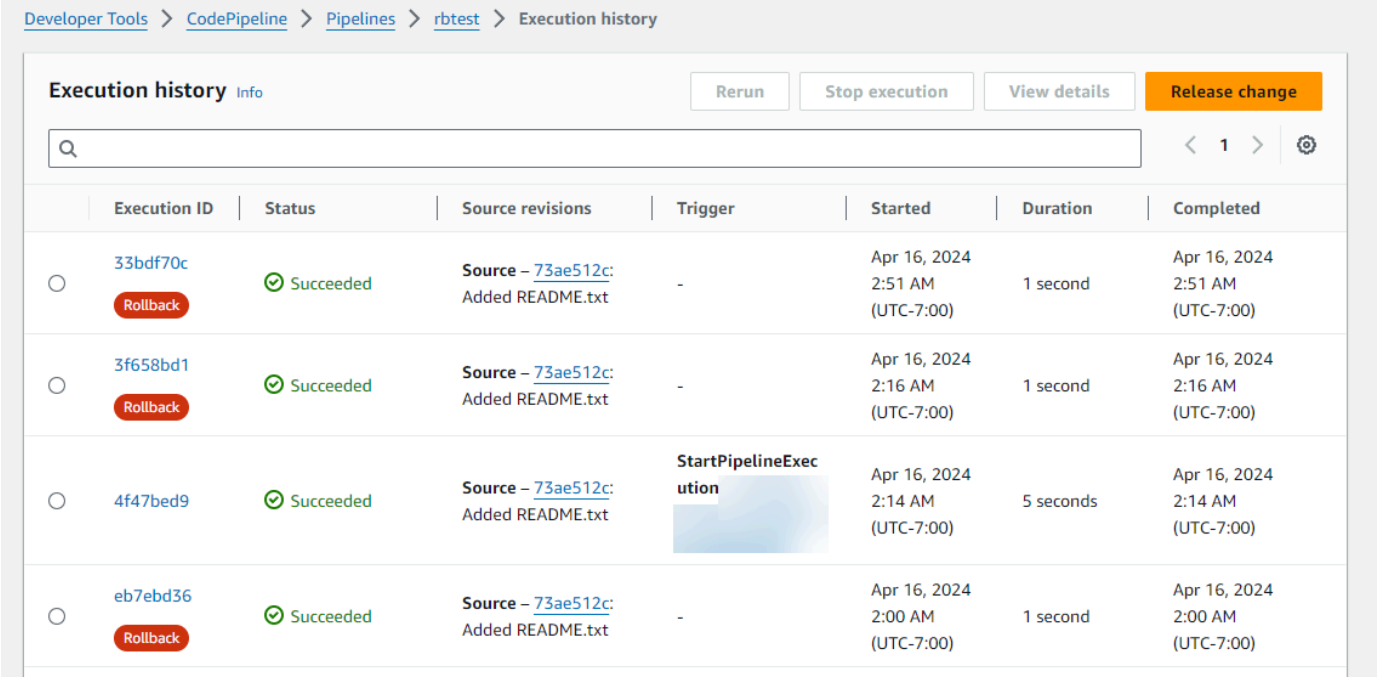
Riwayat eksekusi dipertahankan hingga 12 bulan.

Anda dapat menggunakan konsol untuk melihat riwayat eksekusi dalam pipeline, termasuk status, revisi sumber, dan detail waktu untuk setiap eksekusi.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan, bersama dengan statusnya.
2. Di Nama, pilih nama pipa.
3. Pilih Lihat riwayat.

> **Note**
>
> Untuk pipeline dalam mode eksekusi PARALLEL, tampilan pipeline utama tidak menunjukkan struktur pipa atau eksekusi yang sedang berlangsung. Untuk pipeline dalam mode eksekusi PARALLEL, Anda mengakses struktur pipeline dengan memilih ID untuk eksekusi yang ingin Anda lihat dari halaman riwayat eksekusi. Pilih Riwayat di navigasi kiri, pilih ID eksekusi untuk eksekusi paralel, lalu lihat pipeline pada tab Visualisasi.



4. Lihat status, revisi sumber, detail perubahan, dan pemicu yang terkait dengan setiap eksekusi untuk pipeline Anda. Eksekusi pipeline yang telah diputar kembali akan menampilkan jenis eksekusi Rollback pada layar detail di konsol. Untuk eksekusi gagal yang memicu rollback otomatis, ID eksekusi gagal ditampilkan.
5. Pilih eksekusi. Tampilan detail menunjukkan detail eksekusi, tab Timeline, tab Visualisasi, dan tab Variabel. Nilai variabel untuk variabel pada tingkat pipeline diselesaikan pada saat eksekusi pipeline dan dapat dilihat dalam riwayat eksekusi untuk setiap eksekusi.

> **Note**
>
> Variabel keluaran dari tindakan pipeline dapat dilihat pada tab Variabel keluaran di bawah riwayat untuk setiap eksekusi tindakan.

Lihat status eksekusi (konsol)

Anda dapat melihat status pipeline di Status pada halaman riwayat eksekusi. Pilih tautan ID eksekusi, lalu lihat status tindakan.

Berikut ini adalah status yang valid untuk jaringan pipa, tahapan, dan tindakan:

> **Note**
>
> Status pipeline berikut juga berlaku untuk eksekusi pipeline yang merupakan eksekusi inbound. Untuk melihat eksekusi masuk dan statusnya, lihatMelihat eksekusi masuk (Konsol).

Negara bagian tingkat pipa

Status pipa	Deskripsi
InProgress	Eksekusi pipeline saat ini sedang berjalan.
Stopping	Eksekusi pipeline berhenti karena permintaan untuk berhenti dan menunggu atau menghentikan dan meninggalkan eksekusi pipa.
Dihentikan	Proses penghentian selesai, dan eksekusi pipa dihentikan.
Berhasil	Eksekusi pipa selesai dengan sukses.
Digantikan	Sementara eksekusi pipeline ini menunggu tahap berikutnya selesai, eksekusi pipeline yang lebih baru maju dan dilanjutkan melalui pipeline sebagai gantinya.
Gagal	Eksekusi pipeline tidak berhasil diselesaikan.

Negara bagian tingkat panggung

Status panggung	Deskripsi
InProgress	Panggung sedang berjalan.
Stopping	Eksekusi tahap berhenti karena permintaan untuk berhenti dan menunggu atau berhenti dan meninggalkan eksekusi pipa.
Dihentikan	Proses penghentian selesai, dan eksekusi panggung dihentikan.
Berhasil	Panggung selesai dengan sukses.
Gagal	Panggung tidak selesai dengan sukses.

Negara bagian tingkat aksi

Status tindakan	Deskripsi
InProgress	Tindakan sedang berjalan.
Ditinggalkan	Tindakan ini ditinggalkan karena permintaan untuk menghentikan dan meninggalkan eksekusi pipa.
Berhasil	Tindakan itu selesai dengan sukses.
Gagal	Untuk tindakan persetujuan, status GAGAL berarti tindakan ditolak oleh peninjau atau gagal karena konfigurasi tindakan yang salah.

Melihat eksekusi masuk (Konsol)

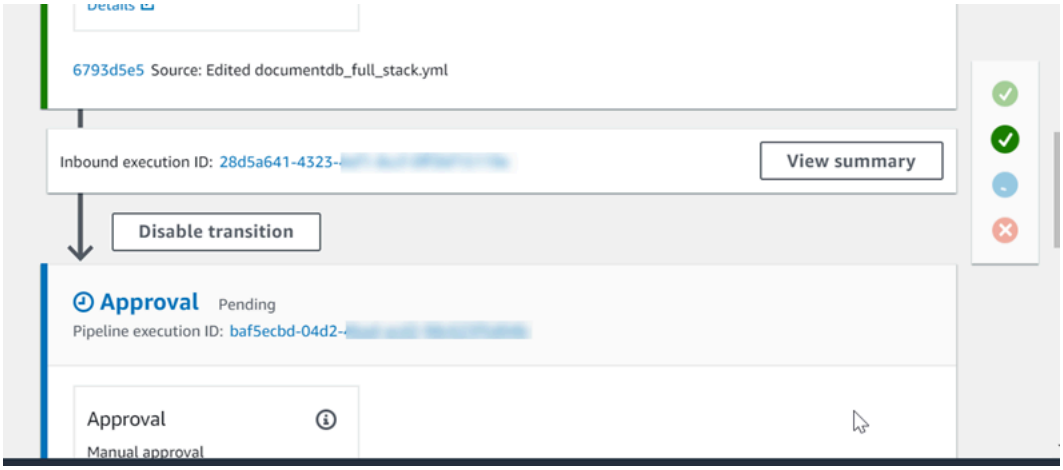
Anda dapat menggunakan konsol untuk melihat status dan detail untuk eksekusi masuk. Ketika transisi diaktifkan atau tahap menjadi tersedia, eksekusi inbound yang `InProgress` berlanjut dan memasuki tahap. Eksekusi inbound dengan `Stopped` status tidak memasuki tahap. Status eksekusi masuk berubah menjadi `Failed` jika pipeline diedit. Saat Anda mengedit pipeline, semua eksekusi yang sedang berlangsung tidak dilanjutkan, dan status eksekusi berubah menjadi. `Failed`

Jika Anda tidak melihat eksekusi masuk, maka tidak ada eksekusi yang tertunda pada transisi tahap dinonaktifkan.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda akan ditampilkan.
2. Pilih nama pipeline yang ingin Anda lihat eksekusi masuk, Lakukan salah satu hal berikut:

   - Pilih Lihat. Dalam diagram pipeline, di bidang ID eksekusi masuk di depan transisi yang dinonaktifkan, Anda dapat melihat ID eksekusi masuk.



Pilih Lihat ringkasan untuk melihat detail eksekusi, seperti ID eksekusi, pemicu sumber, dan nama tahap berikutnya.

- Pilih pipeline dan pilih Lihat riwayat.

Lihat revisi sumber eksekusi pipeline (konsol)

Anda dapat melihat detail tentang artefak sumber (artefak keluaran yang berasal dari tahap pertama pipa) yang digunakan dalam eksekusi pipa. Detailnya mencakup pengidentifikasi, seperti komit IDs, komentar check-in, dan, saat Anda menggunakan CLI, nomor versi tindakan pembuatan pipeline. Untuk beberapa jenis revisi, Anda dapat melihat dan membuka URL komit. Revisi sumber terdiri dari yang berikut:

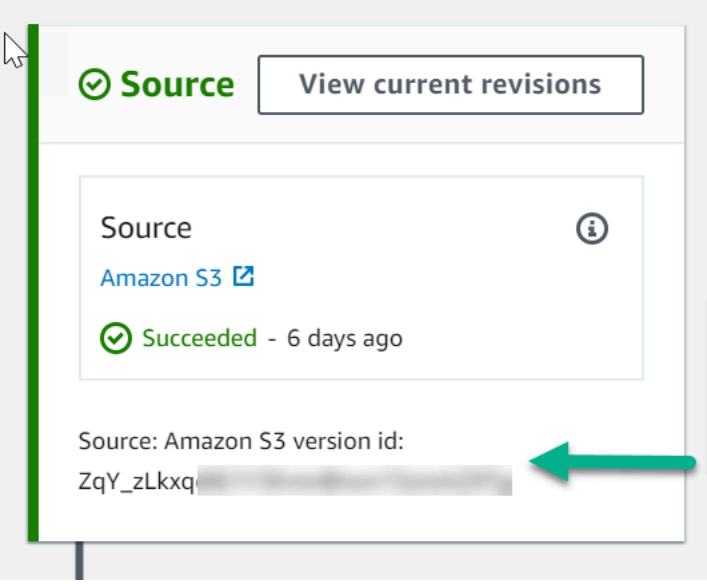
- Ringkasan: Ringkasan informasi tentang revisi artefak terbaru. Untuk GitHub dan CodeCommit repositori, pesan komit. Untuk bucket atau tindakan Amazon S3, konten codepipeline-artifact-revision-summary kunci yang disediakan pengguna yang ditentukan dalam metadata objek.
- revisionUrl: URL revisi untuk revisi artefak (misalnya, URL repositori eksternal).
- RevisionID: ID revisi untuk revisi artefak. Misalnya, untuk perubahan sumber dalam GitHub repositori CodeCommit atau, ini adalah ID komit. Untuk artefak yang disimpan dalam GitHub atau CodeCommit repositori, ID komit ditautkan ke halaman detail komit.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

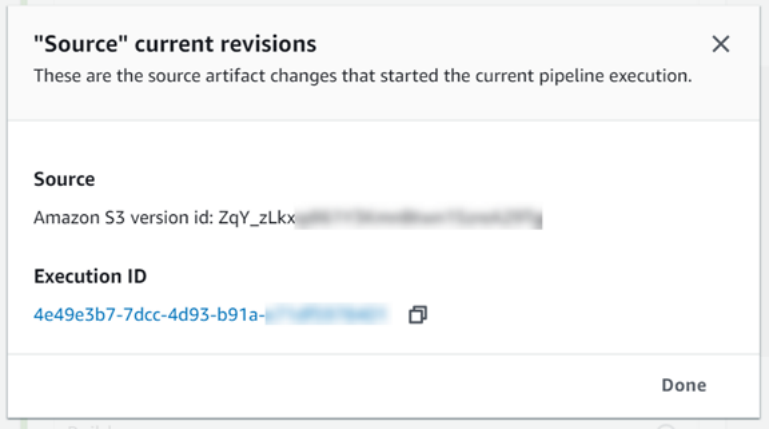
   Nama-nama semua jaringan pipa yang terkait dengan Anda Akun AWS akan ditampilkan.

2. Pilih nama pipeline yang ingin Anda lihat detail revisi sumbernya. Lakukan salah satu tindakan berikut:

   - Pilih Lihat riwayat. Dalam revisi Sumber, perubahan sumber untuk setiap eksekusi terdaftar.
   - Temukan tindakan yang ingin Anda lihat detail revisi sumber, lalu temukan informasi revisi di bagian bawah tahapnya:



   Pilih Lihat revisi saat ini untuk melihat informasi sumber. Dengan pengecualian artefak yang disimpan dalam bucket Amazon S3, pengidentifikasi seperti IDs komit dalam tampilan detail informasi ini ditautkan ke halaman informasi sumber untuk artefak.



Lihat eksekusi tindakan (konsol)

Anda dapat melihat detail tindakan untuk pipeline, seperti ID eksekusi tindakan, artefak input, artefak keluaran, dan status. Anda dapat melihat detail tindakan dengan memilih pipeline di konsol dan kemudian memilih ID eksekusi.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Pilih nama pipeline yang ingin Anda lihat detail tindakannya, lalu pilih Lihat riwayat.
3. Di Execution ID, pilih ID eksekusi yang ingin Anda lihat detail eksekusi tindakan.
4. Anda dapat melihat informasi berikut di tab Timeline:

   a. Di Nama tindakan, pilih tautan untuk membuka halaman detail untuk tindakan di mana Anda dapat melihat status, nama panggung, nama tindakan, data konfigurasi, dan informasi artefak.

   b. Di Penyedia, pilih tautan untuk melihat detail penyedia tindakan. Misalnya, di pipeline contoh sebelumnya, jika Anda memilih CodeDeploy tahap Staging atau Produksi, halaman CodeDeploy konsol untuk CodeDeploy aplikasi yang dikonfigurasi untuk tahap tersebut akan ditampilkan.

Lihat artefak aksi dan informasi penyimpanan artefak (konsol)

Anda dapat melihat detail artefak input dan output untuk suatu tindakan. Anda juga dapat memilih tautan yang membawa Anda ke informasi artefak untuk tindakan itu. Karena penyimpanan artefak menggunakan versi, setiap eksekusi tindakan memiliki lokasi artefak input dan output yang unik.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Pilih nama pipeline yang ingin Anda lihat detail tindakannya, lalu pilih Lihat riwayat.
3. Di Execution ID, pilih ID eksekusi yang ingin Anda lihat detail tindakannya.
4. Pada tab Timeline, di Nama tindakan, pilih tautan untuk membuka halaman detail untuk tindakan tersebut.

5. Pada halaman detail, pada tab Eksekusi, lihat status dan waktu eksekusi tindakan.
6. Pada tab Konfigurasi, lihat konfigurasi sumber daya untuk tindakan (misalnya, nama proyek CodeBuild build).
7. Pada tab Artefak, lihat detail artefak di Jenis Artifact dan penyedia Artifact. Pilih tautan di bawah Nama Artifact untuk melihat artefak di toko artefak.
8. Pada tab Variabel keluaran, lihat variabel yang diselesaikan dari tindakan dalam pipeline untuk eksekusi tindakan.

Lihat detail dan riwayat saluran pipa (CLI)

Anda dapat menjalankan perintah berikut untuk melihat detail tentang pipeline dan eksekusi pipeline Anda:

- list-pipelinesperintah untuk melihat ringkasan semua pipeline yang terkait dengan Anda Akun AWS.
- get-pipelineperintah untuk meninjau detail dari satu pipa.
- list-pipeline-executionsuntuk melihat ringkasan eksekusi terbaru untuk pipa.
- get-pipeline-executionuntuk melihat informasi tentang eksekusi pipeline, termasuk detail tentang artefak, ID eksekusi pipeline, dan nama, versi, dan status pipeline.
- get-pipeline-stateperintah untuk melihat pipeline, stage, dan status tindakan.
- list-action-executionsuntuk melihat detail eksekusi tindakan untuk pipeline.

Topik

- Lihat riwayat eksekusi dengan list-pipeline-executions (CLI)
- Lihat status pipa dengan get-pipeline-state (CLI)
- Lihat status eksekusi masuk dengan get-pipeline-state (CLI)
- Lihat status dan revisi sumber dengan get-pipeline-execution (CLI)
- Lihat eksekusi tindakan dengan list-action-executions (CLI)

Lihat riwayat eksekusi dengan **`list-pipeline-executions`** (CLI)

Anda dapat melihat riwayat eksekusi pipeline.

- Untuk melihat detail tentang eksekusi pipeline sebelumnya, jalankan list-pipeline-executions perintah, dengan menentukan nama unik pipeline. Misalnya, untuk melihat detail tentang status pipeline saat ini bernama*MyFirstPipeline*, masukkan yang berikut ini:

```
aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline
```

Perintah ini mengembalikan informasi ringkasan tentang semua eksekusi pipeline yang histori telah dicatat. Ringkasan mencakup waktu mulai dan akhir, durasi, dan status.

Eksekusi pipeline yang telah digulung kembali akan menunjukkan jenis `Rollback` eksekusi. Untuk eksekusi gagal yang memicu rollback otomatis, ID eksekusi gagal ditampilkan.

Contoh berikut menunjukkan data yang dikembalikan untuk pipeline bernama *MyFirstPipeline* yang telah memiliki tiga eksekusi:

```
{
    "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"
        }
```

Untuk melihat detail lebih lanjut tentang eksekusi pipeline, jalankanget-pipeline-execution, dengan menentukan ID unik dari eksekusi pipeline. Misalnya, untuk melihat detail lebih lanjut tentang eksekusi pertama pada contoh sebelumnya, masukkan yang berikut ini:

```
aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
execution-id 7cf7f7cb-3137-539g-j458-d7eu3EXAMPLE
```

Perintah ini mengembalikan informasi ringkasan tentang eksekusi pipeline, termasuk detail tentang artefak, ID eksekusi pipeline, dan nama, versi, dan status pipeline.

Contoh berikut menunjukkan data yang dikembalikan untuk pipeline bernama*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"
            }
        ]
```

```
    }
}
```

Lihat status pipa dengan **`get-pipeline-state`** (CLI)

Anda dapat menggunakan CLI untuk melihat pipeline, stage, dan status tindakan.

- Untuk melihat detail tentang status pipeline saat ini, jalankan get-pipeline-state perintah, dengan menentukan nama unik pipeline. Misalnya, untuk melihat detail tentang status pipeline saat ini bernama*MyFirstPipeline*, masukkan yang berikut ini:

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

  Perintah ini mengembalikan status saat ini dari semua tahapan pipa dan status tindakan dalam tahapan tersebut.

  Contoh berikut menunjukkan data yang dikembalikan untuk pipa tiga tahap bernama*MyFirstPipeline*, di mana dua tahap pertama dan tindakan menunjukkan keberhasilan, yang ketiga menunjukkan kegagalan, dan transisi antara tahap kedua dan ketiga dinonaktifkan:

  ```
  {
      "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"
        }
    ]
}
```

Lihat status eksekusi masuk dengan **`get-pipeline-state`** (CLI)

Anda dapat menggunakan CLI untuk melihat status eksekusi masuk. Ketika transisi diaktifkan atau tahap menjadi tersedia, eksekusi inbound yang `InProgress` berlanjut dan memasuki tahap. Eksekusi inbound dengan `Stopped` status tidak memasuki tahap. Status eksekusi masuk berubah menjadi `Failed` jika pipeline diedit. Saat Anda mengedit pipeline, semua eksekusi yang sedang berlangsung tidak dilanjutkan, dan status eksekusi berubah menjadi. `Failed`

- Untuk melihat detail tentang status pipeline saat ini, jalankan get-pipeline-state perintah, dengan menentukan nama unik pipeline. Misalnya, untuk melihat detail tentang status pipeline saat ini bernama*MyFirstPipeline*, masukkan yang berikut ini:

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

  Perintah ini mengembalikan status saat ini dari semua tahapan pipa dan status tindakan dalam tahapan tersebut. Output juga menunjukkan ID eksekusi pipeline di setiap tahap, dan apakah ada ID eksekusi masuk untuk tahap dengan transisi yang dinonaktifkan.

  Contoh berikut menunjukkan data yang dikembalikan untuk pipeline dua tahap bernama*MyFirstPipeline*, di mana tahap pertama menunjukkan transisi yang diaktifkan dan eksekusi pipeline yang berhasil, dan tahap kedua, bernama`Beta`, menunjukkan transisi yang dinonaktifkan dan ID eksekusi masuk. Eksekusi inbound dapat memiliki`InProgress`,`Stopped`, atau `FAILED` state.

  ```
  {
      "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
}
```

Lihat status dan revisi sumber dengan **`get-pipeline-execution`** (CLI)

Anda dapat melihat detail tentang artefak sumber (artefak keluaran yang berasal dari tahap pertama pipa) yang digunakan dalam eksekusi pipa. Detailnya mencakup pengidentifikasi, seperti komit IDs, komentar check-in, waktu sejak artefak dibuat atau diperbarui dan, saat Anda menggunakan CLI, nomor versi tindakan build. Untuk beberapa jenis revisi, Anda dapat melihat dan membuka URL komit untuk versi artefak. Revisi sumber terdiri dari yang berikut:

- Ringkasan: Ringkasan informasi tentang revisi artefak terbaru. Untuk GitHub dan AWS CodeCommit repositori, pesan komit. Untuk bucket atau tindakan Amazon S3, konten codepipeline-artifact-revision-summary kunci yang disediakan pengguna yang ditentukan dalam metadata objek.
- revisionUrl: ID komit untuk revisi artefak. Untuk artefak yang disimpan dalam GitHub atau AWS CodeCommit repositori, ID komit ditautkan ke halaman detail komit.

Anda dapat menjalankan get-pipeline-execution perintah untuk melihat informasi tentang revisi sumber terbaru yang disertakan dalam eksekusi pipeline. Setelah Anda pertama kali menjalankan get-pipeline-state perintah untuk mendapatkan detail tentang semua tahapan dalam pipeline, Anda mengidentifikasi ID eksekusi yang berlaku untuk tahap yang Anda inginkan rincian revisi sumber. Kemudian Anda menggunakan ID eksekusi dalam get-pipeline-execution perintah. (Karena tahapan dalam pipeline mungkin terakhir berhasil diselesaikan selama proses pipeline yang berbeda, mereka dapat memiliki eksekusi yang berbeda IDs.)

Dengan kata lain, jika Anda ingin melihat detail tentang artefak yang saat ini dalam tahap Pementasan, jalankan get-pipeline-state perintah, identifikasi ID eksekusi saat ini dari tahap Pementasan, dan kemudian jalankan get-pipeline-execution perintah menggunakan ID eksekusi itu.

Untuk melihat status dan revisi sumber dalam pipeline

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan perintah. get-pipeline-state Untuk pipeline bernama*MyFirstPipeline*, Anda akan memasukkan:

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

   Perintah ini mengembalikan status terbaru dari pipeline, termasuk ID eksekusi pipeline terbaru untuk setiap tahap.

2. Untuk melihat detail tentang eksekusi pipeline, jalankan get-pipeline-execution perintah, tentukan nama unik pipeline dan ID eksekusi pipeline dari eksekusi yang ingin Anda lihat detail artefak. Misalnya, untuk melihat detail tentang eksekusi pipeline bernama*MyFirstPipeline*, dengan ID eksekusi 3137F7CB-7CF7-039J-S83L-D7EU3Example, Anda akan memasukkan yang berikut ini:

   ```
   aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
   execution-id 3137f7cb-7cf7-039j-s83l-d7eu3EXAMPLE
   ```

   Perintah ini mengembalikan informasi tentang setiap revisi sumber yang merupakan bagian dari eksekusi pipeline dan mengidentifikasi informasi tentang pipeline. Hanya informasi tentang tahapan pipa yang termasuk dalam eksekusi itu yang disertakan. Mungkin ada tahapan lain dalam pipa yang bukan bagian dari eksekusi pipa itu.

   Contoh berikut menunjukkan data yang dikembalikan untuk sebagian pipa bernama*MyFirstPipeline*, di mana artefak bernama "MyApp" disimpan dalam GitHub repositori:

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"
         }
    }
}
```

Lihat eksekusi tindakan dengan **`list-action-executions`** (CLI)

Anda dapat melihat detail eksekusi tindakan untuk pipeline, seperti ID eksekusi tindakan, artefak input, artefak keluaran, hasil eksekusi, dan status. Anda memberikan filter ID Eksekusi untuk menampilkan daftar tindakan dalam eksekusi pipeline:

> **Note**
>
> Riwayat eksekusi terperinci tersedia untuk eksekusi yang dijalankan pada atau setelah 21 Februari 2019.

- Untuk melihat eksekusi tindakan untuk pipeline, lakukan salah satu hal berikut:
  - Untuk melihat detail untuk semua eksekusi tindakan dalam pipeline, jalankan list-action-executions perintah, dengan menentukan nama unik pipeline. Misalnya, untuk melihat eksekusi tindakan dalam pipeline bernama*`MyFirstPipeline`*, masukkan yang berikut ini:

    ```
    aws codepipeline list-action-executions --pipeline-name MyFirstPipeline
    ```

    Berikut ini menunjukkan sebagian dari output sampel untuk perintah ini:

```
{
    "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"
                }
            }
        },
```

```
. . .
```

- Untuk melihat semua eksekusi tindakan dalam eksekusi pipeline, jalankan list-action-executions perintah, dengan menentukan nama unik pipeline dan ID eksekusi. Misalnya, untuk melihat eksekusi tindakan untuk sebuah*Execution_ID*, masukkan yang berikut ini:

```
aws codepipeline list-action-executions --pipeline-name MyFirstPipeline --filter
 pipelineExecutionId=Execution_ID
```

- Berikut ini menunjukkan sebagian dari output sampel untuk perintah ini:

```
{
    "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"
                    }
                ]
            },
```

```
. . .
```

Mengatur atau mengubah mode eksekusi pipa

Anda dapat mengatur mode eksekusi untuk pipeline Anda untuk menentukan bagaimana beberapa eksekusi ditangani.

Untuk informasi selengkapnya tentang mode eksekusi pipeline, lihatBagaimana eksekusi pipa bekerja.

Important

Untuk saluran pipa dalam mode PARALLEL, saat mengedit mode eksekusi pipa ke ANTRIAN atau DIGANTI, status pipa tidak akan menampilkan status yang diperbarui sebagai PARALLEL. Untuk informasi selengkapnya, lihat Pipelines diubah dari mode PARALLEL akan menampilkan mode eksekusi sebelumnya.

Important

Untuk saluran pipa dalam mode PARALLEL, saat mengedit mode eksekusi pipa ke ANTRIAN atau DIGANTI, definisi pipa untuk pipa di setiap mode tidak akan diperbarui. Untuk informasi selengkapnya, lihat Pipelines dalam mode PARALLEL memiliki definisi pipeline yang sudah ketinggalan zaman jika diedit saat mengubah ke mode QUEUED atau SUPERSEDED.

Important

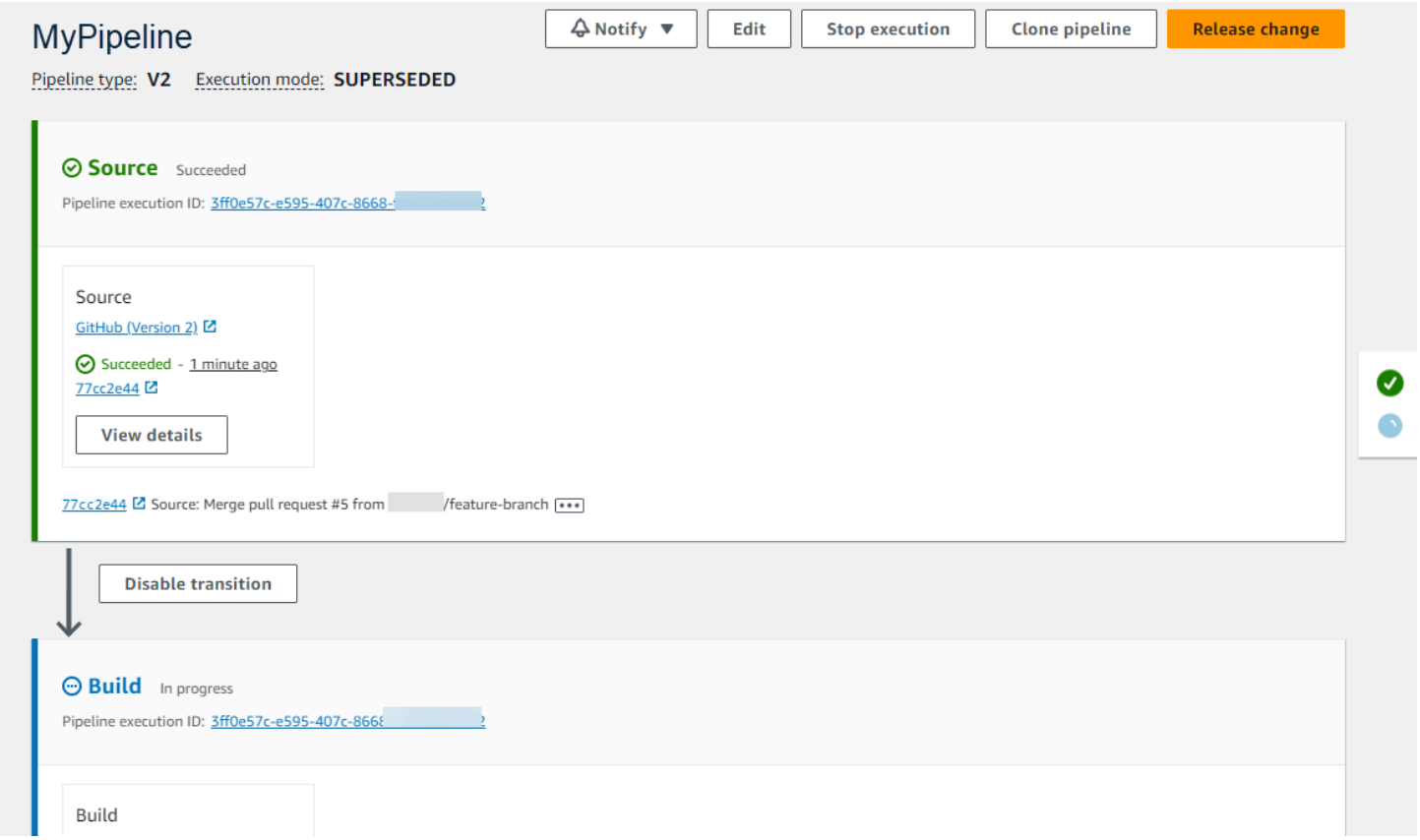
Untuk saluran pipa dalam mode PARALLEL, rollback panggung tidak tersedia. Demikian pula, kondisi kegagalan dengan jenis hasil rollback tidak dapat ditambahkan ke pipa mode PARALLEL.

Pertimbangan untuk melihat mode eksekusi

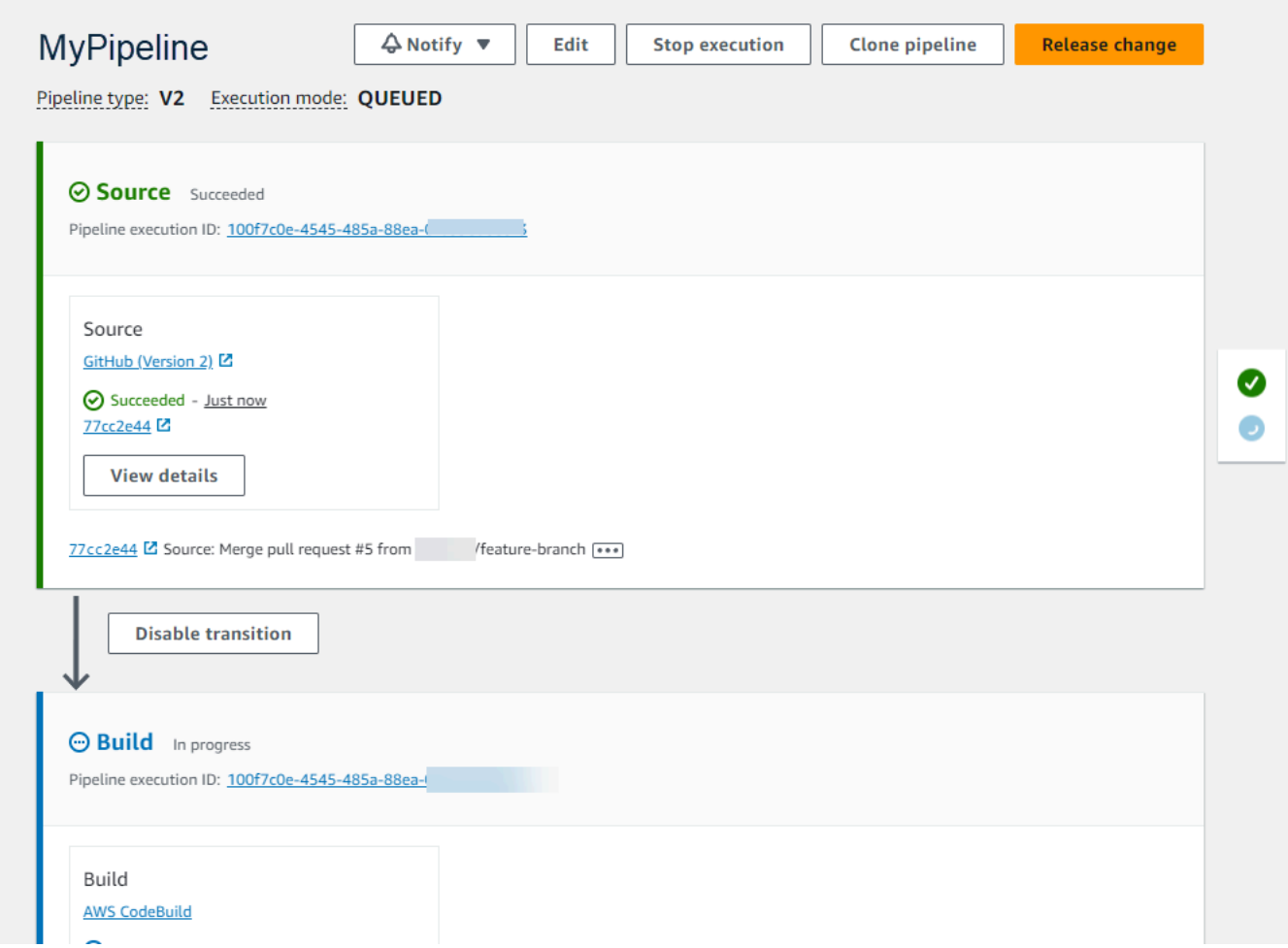
Ada pertimbangan untuk melihat saluran pipa dalam mode eksekusi tertentu.

Untuk mode SUPERSEDED dan QUEUED, gunakan tampilan pipeline untuk melihat eksekusi yang sedang berlangsung, dan klik ID eksekusi untuk melihat detail dan riwayat. Untuk mode PARALLEL, klik ID eksekusi untuk melihat eksekusi yang sedang berlangsung pada tab Visualisasi.

Berikut ini menunjukkan tampilan untuk mode SUPERSEDED di. CodePipeline

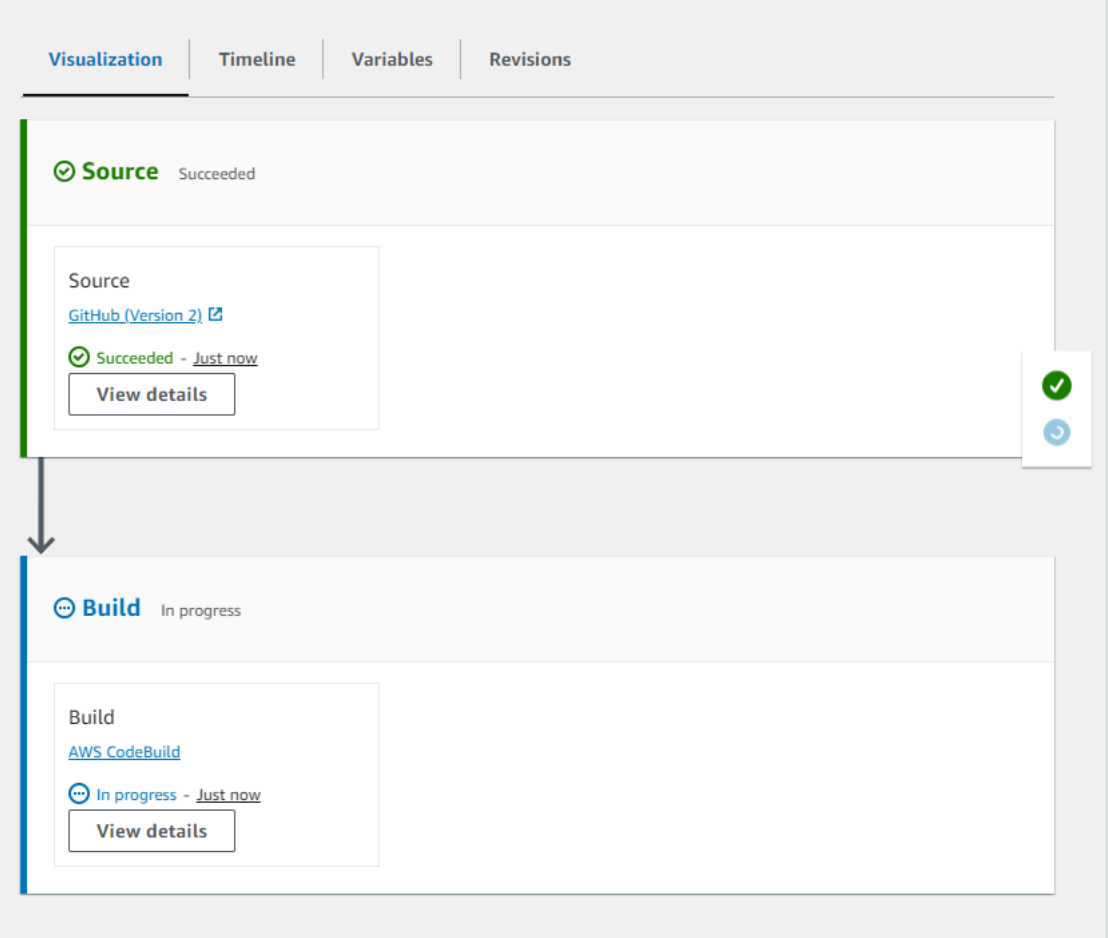


Berikut ini menunjukkan tampilan untuk mode ANTRIAN di. CodePipeline



Berikut ini menunjukkan tampilan untuk mode PARALLEL di CodePipeline.

> **Important**
>
> Untuk saluran pipa dalam mode PARALLEL, rollback panggung tidak tersedia. Demikian pula, kondisi kegagalan dengan jenis hasil rollback tidak dapat ditambahkan ke pipa mode PARALLEL.



Pertimbangan untuk beralih di antara mode eksekusi

Berikut ini adalah pertimbangan untuk jaringan pipa saat mengubah mode untuk pipa. Saat beralih dari satu mode eksekusi ke mode Edit lainnya dan kemudian menyimpan perubahan, tampilan atau status tertentu mungkin menyesuaikan.

Misalnya, ketika beralih dari mode PARALLEL ke mode ANTRIAN atau SUPERSEDED, eksekusi yang dimulai dalam mode PARALEL akan terus berjalan. Ini dapat dilihat di halaman riwayat eksekusi. Tampilan pipeline akan menampilkan eksekusi yang berjalan pada mode QUEUED atau SUPERSEDED sebelumnya atau status kosong sebaliknya.

Sebagai contoh lain, saat beralih dari QUEUED atau SUPERSEDED ke mode PARALLEL, Anda tidak akan lagi melihat halaman pipeline. view/state Untuk melihat eksekusi dalam mode PARALLEL, gunakan tab visualisasi pada halaman detail eksekusi. Eksekusi yang dimulai dalam mode SUPERSEDED atau ANTRIAN akan dibatalkan.

Tabel berikut memberikan detail lebih lanjut.

Perubahan mode	Detail eksekusi yang tertunda dan aktif	Detail status saluran pipa
DIGANTIKAN menjadi DIGANTIKAN /DIGANTIKAN ke ANTRIAN	• Eksekusi aktif dibatalkan setelah tindakan yang sedang berlangsung selesai. • Eksekusi yang tertunda dibatalkan.	Status pipa, seperti dibatalkan, dipertahankan antara versi mode pertama dan mode kedua.
ANTRIAN ke ANTRIAN / ANTRIAN ke DIGANTIKAN	• Eksekusi aktif dibatalkan setelah tindakan yang sedang berlangsung selesai. • Eksekusi yang tertunda dibatalkan.	Status pipa, seperti dibatalkan, dipertahankan antara versi mode pertama dan mode kedua.
PARALEL ke PARALEL	Semua eksekusi diizinkan untuk berjalan secara independen dari pembaruan definisi pipa.	Kosong. Mode paralel tidak memiliki status pipa.
DIGANTIKAN ke PARALLEL / ANTRIAN ke PARALLEL	• Eksekusi aktif dibatalkan setelah tindakan yang sedang berlangsung selesai. • Eksekusi yang tertunda dibatalkan.	Kosong. Mode paralel tidak memiliki status pipa.

Mengatur atau mengubah mode eksekusi pipeline (konsol)

Anda dapat menggunakan konsol untuk mengatur mode eksekusi pipeline.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua pipeline yang terkait dengan AWS akun Anda akan ditampilkan.

2. Di Nama, pilih nama pipeline yang ingin Anda edit.
3. Pada halaman detail pipeline, pilih Edit.
4. Pada halaman Edit, pilih Edit: Properti Pipeline.
5. Pilih mode untuk pipeline Anda.

   - Digantikan
   - Antrian (Diperlukan tipe pipa V2)
   - Paralel (Diperlukan tipe pipa V2)
6. Pada halaman Edit, pilih Selesai.

Mengatur mode eksekusi pipa (CLI)

Untuk menggunakan AWS CLI untuk mengatur mode eksekusi pipeline, gunakan `update-pipeline` perintah `create-pipeline` or.

1. Buka sesi terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan jalankan get-pipeline perintah untuk menyalin struktur pipeline ke file JSON. Misalnya, untuk pipeline bernama**`MyFirstPipeline`**, masukkan perintah berikut:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Perintah ini tidak mengembalikan apa pun, tetapi file yang Anda buat akan muncul di direktori tempat Anda menjalankan perintah.
2. Buka file JSON di editor teks biasa dan modifikasi struktur file untuk mencerminkan mode eksekusi pipeline yang ingin Anda atur, seperti ANTRIAN.

   ```
   "executionMode": "QUEUED"
   ```

   Contoh berikut menunjukkan bagaimana Anda akan mengatur mode eksekusi ke QUEUED dalam contoh pipeline dengan dua tahap.

   ```
   {
       "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. Jika Anda bekerja dengan struktur pipa yang diambil menggunakan get-pipeline perintah, Anda harus memodifikasi struktur dalam file JSON. Anda harus menghapus `metadata` baris dari file sehingga update-pipeline perintah dapat menggunakannya. Hapus bagian dari struktur pipa di file JSON (`"metadata": { }`garis dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang).

   Misalnya, hapus baris berikut dari struktur:

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

   Simpan file tersebut.

4. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, dengan menentukan file JSON pipeline:

> **Important**
>
> Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

Perintah ini mengembalikan seluruh struktur pipa yang diedit.

> **Note**
>
> update-pipelinePerintah menghentikan pipa. Jika revisi sedang dijalankan melalui pipeline saat Anda menjalankan update-pipeline perintah, proses itu dihentikan. Anda harus memulai pipeline secara manual untuk menjalankan revisi itu melalui pipeline yang diperbarui.

Gulung kembali atau coba lagi tahapan

Anda dapat menggunakan AWS CodePipeline konsol atau AWS CLI untuk memutar kembali secara manual atau mencoba lagi tahap atau tindakan dalam satu tahap. Untuk mengonfigurasi kondisi untuk tahapan di mana Anda ingin menggunakan Rollback atau Coba lagi sebagai hasil yang dikonfigurasi, lihat. Konfigurasikan kondisi untuk sebuah panggung

Topik

- Mengkonfigurasi percobaan ulang tahap untuk tahap yang gagal atau tindakan yang gagal
- Mengkonfigurasi rollback panggung

Mengkonfigurasi percobaan ulang tahap untuk tahap yang gagal atau tindakan yang gagal

Anda dapat mencoba lagi tahap yang gagal tanpa harus menjalankan pipeline lagi dari awal. Anda melakukan ini dengan mencoba kembali tindakan yang gagal dalam satu tahap atau dengan mencoba kembali semua tindakan di tahap mulai dari tindakan pertama di panggung. Saat Anda mencoba lagi tindakan yang gagal dalam satu tahap, semua tindakan yang masih berlangsung terus bekerja, dan tindakan yang gagal dipicu lagi. Ketika Anda mencoba lagi tahap yang gagal dari tindakan pertama di panggung, tahap tidak dapat memiliki tindakan apa pun yang sedang berlangsung. Sebelum sebuah tahap dapat dicoba lagi, itu harus memiliki semua tindakan gagal atau beberapa tindakan gagal dan beberapa berhasil.

> **Important**
>
> Mencoba lagi tahap yang gagal mencoba kembali semua tindakan di panggung dari aksi pertama di panggung, dan mencoba kembali tindakan yang gagal mencoba kembali semua tindakan yang gagal di panggung. Ini mengesampingkan artefak keluaran dari tindakan yang sebelumnya berhasil dalam eksekusi yang sama.
>
> Meskipun artefak dapat diganti, riwayat eksekusi tindakan yang sebelumnya berhasil masih dipertahankan.

Jika Anda menggunakan konsol untuk melihat pipeline, tombol tahap Coba lagi atau tombol Tindakan gagal Coba lagi muncul di panggung yang dapat dicoba ulang.

Jika Anda menggunakan AWS CLI, Anda dapat menggunakan get-pipeline-state perintah untuk menentukan apakah ada tindakan yang gagal.

> **Note**
>
> Dalam kasus berikut, Anda mungkin tidak dapat mencoba lagi tahap:
>
> - Semua tindakan di panggung berhasil, sehingga panggung tidak dalam status gagal.
> - Struktur pipa keseluruhan berubah setelah tahap gagal.
> - Upaya coba lagi di panggung sudah berlangsung.

Topik

- Pertimbangan untuk percobaan ulang tahap
- Coba lagi tahap yang gagal secara manual
- Konfigurasikan tahapan untuk coba lagi otomatis pada kegagalan

Pertimbangan untuk percobaan ulang tahap

Pertimbangan untuk percobaan ulang tahap adalah sebagai berikut:

- Anda hanya dapat mengonfigurasi percobaan ulang otomatis pada kegagalan panggung untuk satu upaya coba lagi.
- Anda dapat mengonfigurasi percobaan ulang otomatis pada kegagalan panggung untuk semua tindakan, termasuk `Source` tindakan.

Coba lagi tahap yang gagal secara manual

Anda dapat mencoba lagi tahap yang gagal secara manual menggunakan konsol atau CLI.

Anda juga dapat mengonfigurasi tahapan untuk mencoba lagi secara otomatis pada kegagalan seperti yang dijelaskan dalamKonfigurasikan tahapan untuk coba lagi otomatis pada kegagalan.

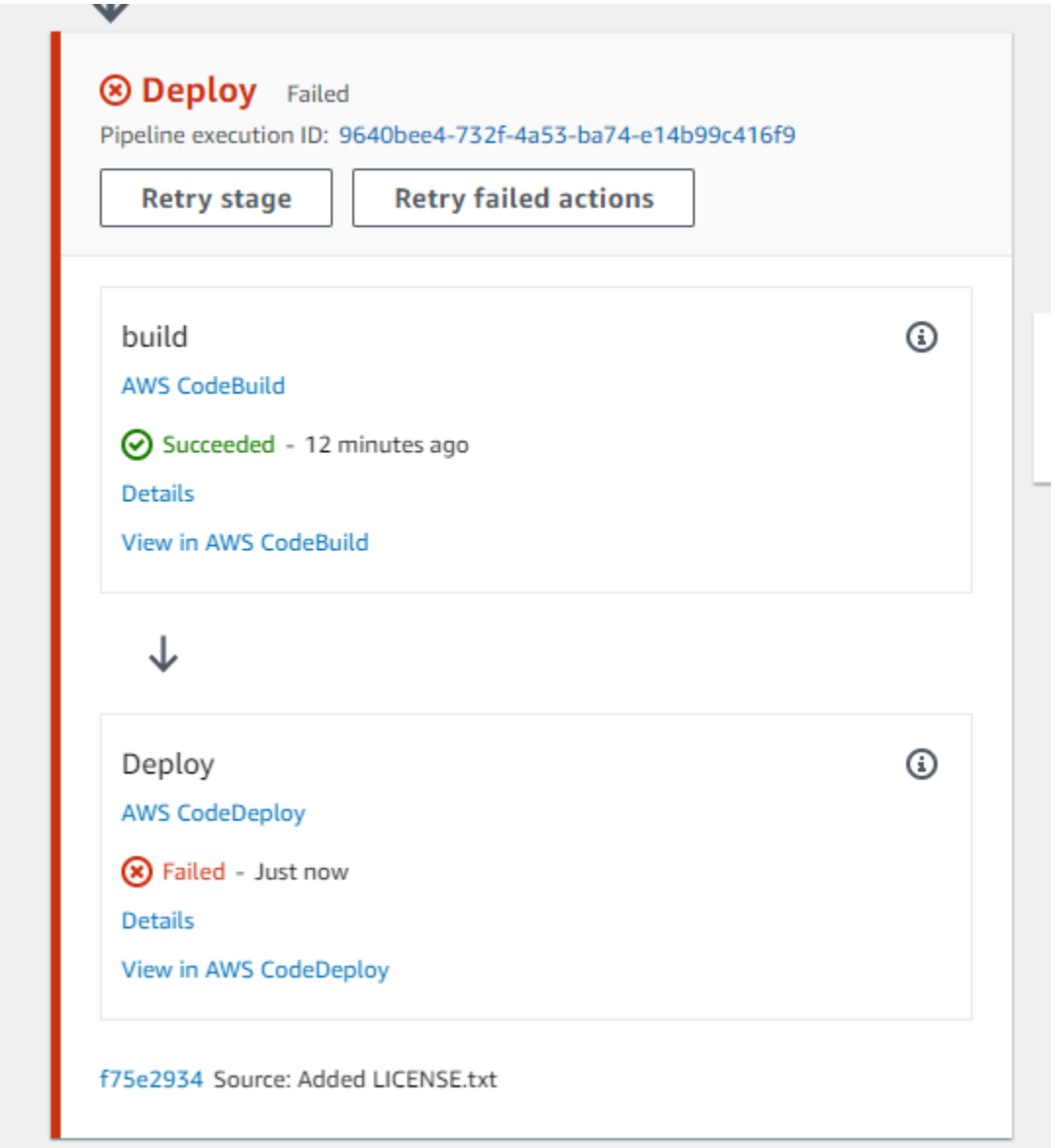
Coba lagi tahap yang gagal secara manual (konsol)

Untuk mencoba lagi tahap yang gagal atau tindakan yang gagal dalam tahap - konsol

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipa.
3. Temukan panggung dengan tindakan yang gagal, lalu pilih salah satu dari yang berikut ini:

   - Untuk mencoba kembali semua tindakan di panggung, pilih Coba lagi tahap.
   - Untuk mencoba lagi hanya tindakan yang gagal di panggung, pilih Coba lagi tindakan yang gagal.



Jika semua tindakan yang dicoba ulang di tahap selesai dengan sukses, pipa terus berjalan.

Coba lagi tahap yang gagal secara manual (CLI)

Untuk mencoba lagi tahap yang gagal atau tindakan yang gagal dalam satu tahap - CLI

Untuk menggunakan AWS CLI untuk mencoba kembali semua tindakan atau semua tindakan yang gagal, Anda menjalankan retry-stage-execution perintah dengan parameter berikut:

```
--pipeline-name <value>
```

```
--stage-name <value>
--pipeline-execution-id <value>
--retry-mode ALL_ACTIONS/FAILED_ACTIONS
```

> **Note**
>
> Nilai yang dapat Anda gunakan `retry-mode` adalah `FAILED_ACTIONS` dan`ALL_ACTIONS`.

1. Di terminal (Linux, macOS, atau Unix) atau command prompt (Windows), jalankan retry-stage-executionperintah, seperti yang ditunjukkan pada contoh berikut untuk pipeline bernama. `MyPipeline`

   ```
   aws codepipeline retry-stage-execution --pipeline-name MyPipeline --stage-name
    Deploy --pipeline-execution-id b59babff-5f34-EXAMPLE --retry-mode FAILED_ACTIONS
   ```

   Output mengembalikan ID eksekusi:

   ```
   {
       "pipelineExecutionId": "b59babff-5f34-EXAMPLE"
   }
   ```

2. Anda juga dapat menjalankan perintah dengan file input JSON. Pertama-tama Anda membuat file JSON yang mengidentifikasi pipeline, tahapan yang berisi tindakan gagal, dan eksekusi pipeline terbaru di tahap itu. Anda kemudian menjalankan retry-stage-execution perintah dengan `--cli-input-json` parameter. Untuk mengambil detail yang Anda butuhkan untuk file JSON, paling mudah untuk menggunakan perintah. get-pipeline-state

   a. Di terminal (Linux, macOS, atau Unix) atau command prompt (Windows), jalankan get-pipeline-stateperintah pada pipeline. Misalnya, untuk pipeline bernama MyFirstPipeline, Anda akan mengetik sesuatu yang mirip dengan berikut ini:

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

      Respons terhadap perintah mencakup informasi status pipa untuk setiap tahap. Dalam contoh berikut, respons menunjukkan bahwa satu atau lebih tindakan gagal dalam tahap Pementasan:

      ```
      {
      ```

```
    "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. Dalam editor teks biasa, buat file tempat Anda akan merekam yang berikut ini, dalam format JSON:

- Nama pipa yang berisi tindakan yang gagal
- Nama panggung yang berisi tindakan yang gagal
- ID eksekusi pipeline terbaru di panggung
- Mode coba lagi.

Untuk MyFirstPipeline contoh sebelumnya, file Anda akan terlihat seperti ini:

```
{
    "pipelineName": "MyFirstPipeline",
    "stageName": "Staging",
    "pipelineExecutionId": "3137f7cb-7cf7-EXAMPLE",
    "retryMode": "FAILED_ACTIONS"
}
```

c. Simpan file dengan nama seperti**`retry-failed-actions.json`**.

d. Panggil file yang Anda buat saat menjalankan retry-stage-executionperintah. Contoh:

> **Important**
>
> Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

```
aws codepipeline retry-stage-execution --cli-input-json file://retry-failed-actions.json
```

e. Untuk melihat hasil percobaan ulang, buka CodePipeline konsol dan pilih pipeline yang berisi tindakan yang gagal, atau gunakan get-pipeline-state perintah lagi. Untuk informasi selengkapnya, lihat Lihat saluran pipa dan detailnya di CodePipeline.

Konfigurasikan tahapan untuk coba lagi otomatis pada kegagalan

Anda dapat mengonfigurasi panggung untuk coba lagi otomatis pada kegagalan. Tahap akan membuat satu percobaan lagi dan menampilkan status coba lagi pada tahap gagal pada halaman View pipeline.

Anda dapat mengonfigurasi mode coba lagi dengan menentukan bahwa tahap harus secara otomatis mencoba kembali semua tindakan dalam tahap gagal atau hanya tindakan yang gagal di panggung.

Konfigurasikan panggung untuk coba lagi otomatis pada kegagalan (konsol)

Anda dapat menggunakan konsol untuk mengonfigurasi panggung untuk coba lagi otomatis.

Konfigurasikan panggung untuk coba lagi otomatis (konsol)

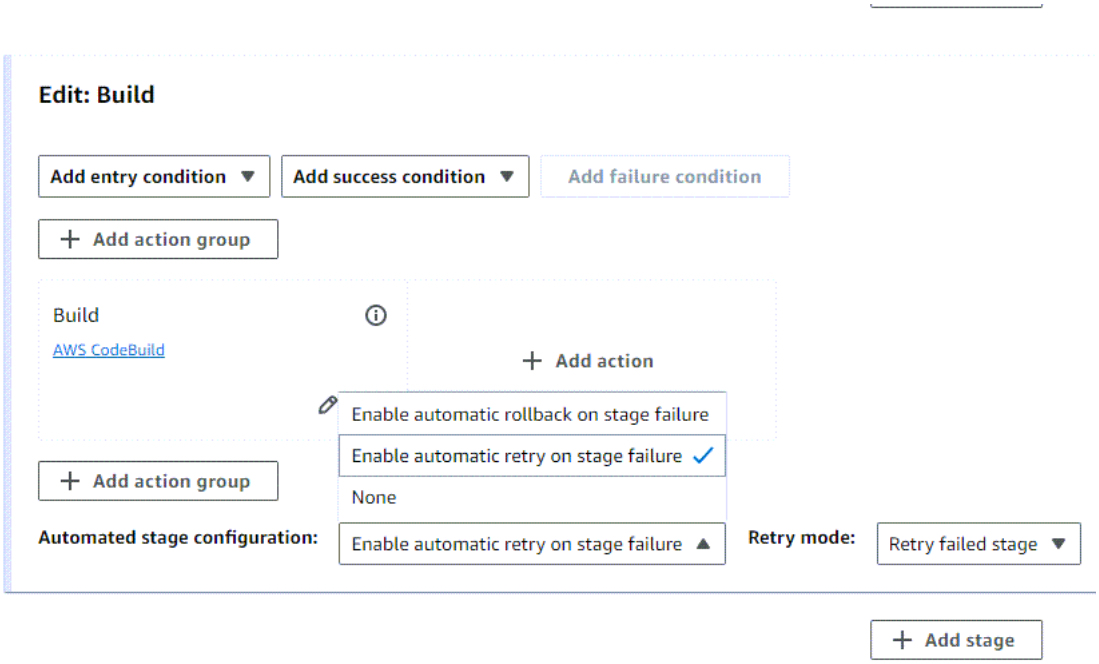
1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda edit.
3. Pada halaman detail pipeline, pilih Edit.
4. Pada halaman Edit, untuk tindakan yang ingin Anda edit, pilih Edit tahap.

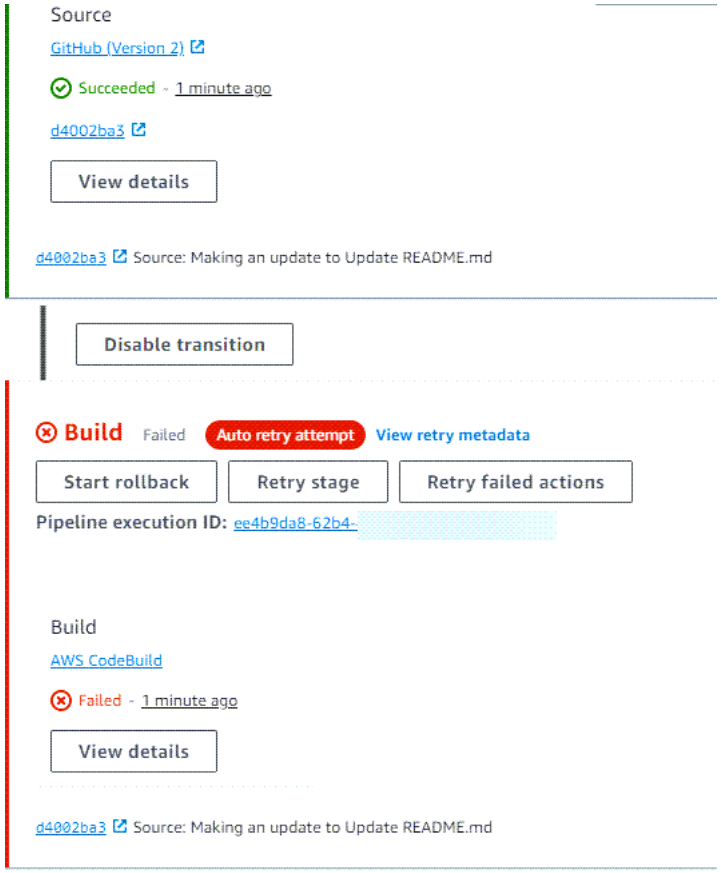
5. Pilih Konfigurasi tahap otomatis:, lalu pilih Aktifkan coba ulang otomatis pada kegagalan panggung. Simpan perubahan pada pipeline Anda.
6. Dalam konfigurasi tahap otomatis:, pilih salah satu mode coba lagi berikut:

   - Untuk menentukan bahwa mode akan mencoba kembali semua tindakan di panggung, pilih Coba lagi tahap gagal.
   - Untuk menentukan bahwa mode hanya akan mencoba lagi tindakan yang gagal di panggung, pilih Coba lagi tindakan yang gagal.

   Simpan perubahan pada pipeline Anda.



7. Setelah pipeline Anda berjalan, jika kegagalan tahap terjadi, upaya coba ulang otomatis akan dilakukan. Contoh berikut menunjukkan tahap build yang telah dicoba ulang secara otomatis.



8. Untuk melihat detail tentang upaya coba lagi, pilih. Jendela menampilkan.

Konfigurasikan panggung untuk percobaan ulang otomatis (CLI)

Untuk menggunakan AWS CLI untuk mengkonfigurasi tahap untuk secara otomatis mencoba kembali pada kegagalan, gunakan perintah untuk membuat atau memperbarui pipeline seperti yang dijelaskan dalam Buat pipeline, tahapan, dan tindakan danEdit pipa di CodePipeline.

- Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan `update-pipeline` perintah, menentukan kondisi kegagalan dalam

struktur pipa. Contoh berikut mengonfigurasi percobaan ulang otomatis untuk nama bertahap: `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",
                    },
            }
```

Konfigurasikan tahapan untuk coba ulang otomatis ()CloudFormation

CloudFormation Untuk menggunakan konfigurasi tahapan untuk percobaan ulang otomatis pada kegagalan, gunakan parameter siklus hidup `OnFailure` tahap. Gunakan `RetryConfiguration` parameter untuk mengkonfigurasi mode coba lagi.

```
OnFailure:
     Result: RETRY
     RetryConfiguration:
         RetryMode: ALL_ACTIONS
```

- Perbarui template seperti yang ditunjukkan pada cuplikan berikut. Contoh berikut mengonfigurasi coba ulang otomatis untuk tahap bernama: `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'
```

Untuk informasi selengkapnya tentang mengonfigurasi percobaan ulang tahap pada kegagalan, lihat OnFailure`StageDeclaration`di bawah CloudFormation Panduan Pengguna.

Mengkonfigurasi rollback panggung

Anda dapat memutar kembali panggung ke eksekusi yang berhasil di tahap itu. Anda dapat mengkonfigurasi tahap untuk rollback pada kegagalan, atau Anda dapat memutar kembali panggung secara manual. Operasi yang digulung kembali akan menghasilkan eksekusi baru. Eksekusi pipeline target yang dipilih untuk rollback digunakan untuk mengambil revisi sumber dan variabel.

Jenis eksekusi, baik standar atau rollback, ditampilkan dalam riwayat pipeline, status pipeline, dan detail eksekusi pipeline.

Topik

- Pertimbangan untuk rollback
- Gulung kembali panggung secara manual
- Konfigurasikan panggung untuk rollback otomatis
- Lihat status rollback dalam daftar eksekusi
- Lihat detail status rollback

Pertimbangan untuk rollback

Pertimbangan untuk rollback tahap adalah sebagai berikut:

- Anda tidak dapat memutar kembali tahap sumber.
- Pipeline hanya dapat memutar kembali ke eksekusi sebelumnya jika eksekusi sebelumnya dimulai dalam versi struktur pipa saat ini.
- Anda tidak dapat memutar kembali ke ID eksekusi target yang merupakan jenis eksekusi rollback.
- CodePipeline akan menggunakan variabel dan artefak dari eksekusi yang digulirkan kembali.

Gulung kembali panggung secara manual

Anda dapat memutar kembali panggung secara manual menggunakan konsol atau CLI. Pipeline hanya dapat memutar kembali ke eksekusi sebelumnya jika eksekusi sebelumnya dimulai dalam versi struktur pipa saat ini.

Anda juga dapat mengonfigurasi panggung untuk memutar kembali secara otomatis pada kegagalan seperti yang dijelaskan dalamKonfigurasikan panggung untuk rollback otomatis.

Gulung kembali panggung secara manual (konsol)

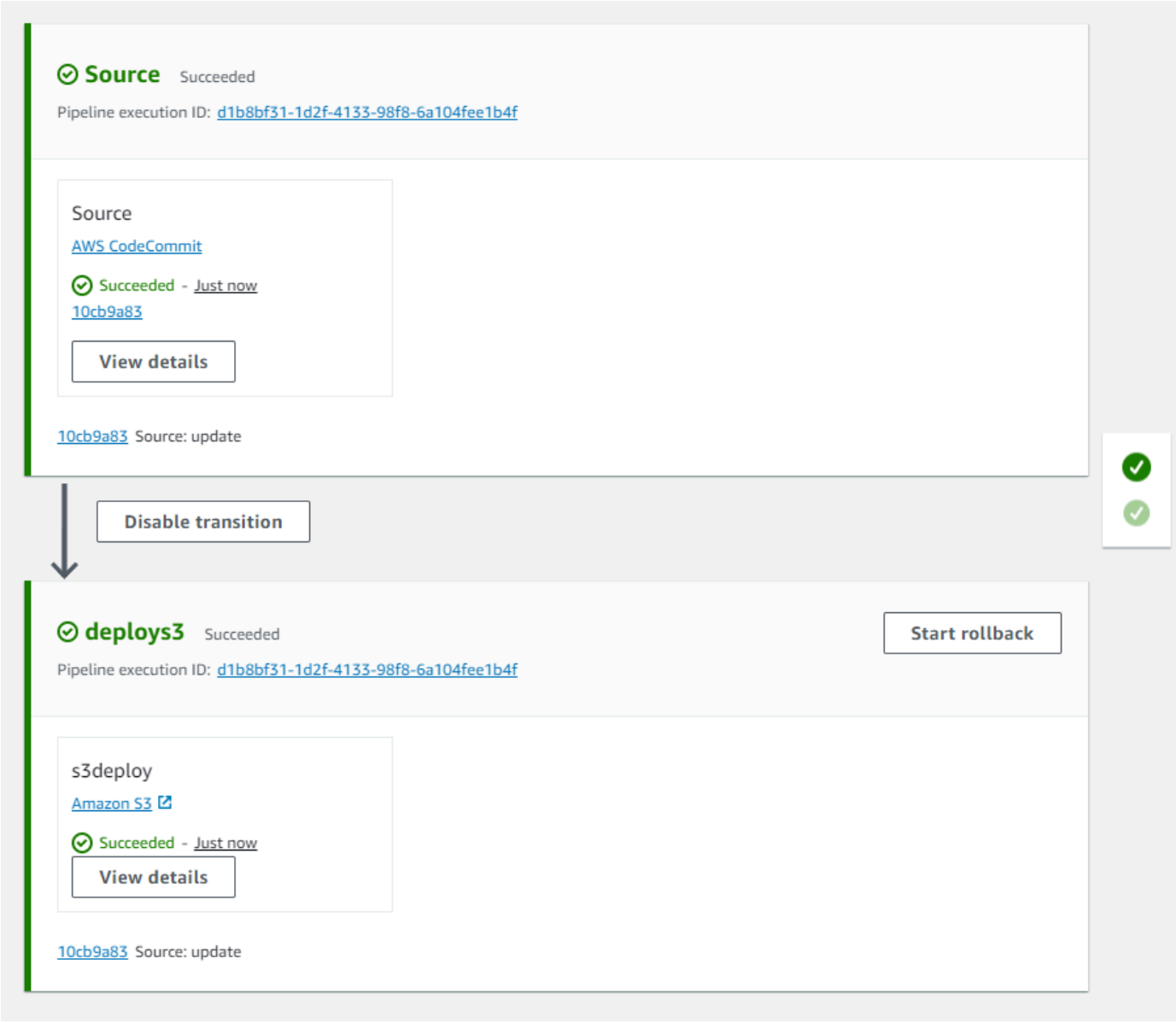
Anda dapat menggunakan konsol untuk memutar kembali tahap secara manual ke eksekusi pipeline target. Saat panggung diputar kembali, label Rollback ditampilkan pada visualisasi pipeline di konsol.

Gulung kembali panggung secara manual (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua pipeline yang terkait dengan AWS akun Anda ditampilkan.

2. Dalam Nama, pilih nama pipa dengan panggung untuk memutar kembali.



3. Di atas panggung, pilih Mulai rollback. Tampilan Roll back to page.
4. Pilih eksekusi target yang ingin Anda putar kembali panggung.

> **Note**
>
> Daftar eksekusi pipa target yang tersedia adalah semua eksekusi dalam versi pipeline saat ini yang dimulai pada 1 Februari 2024.

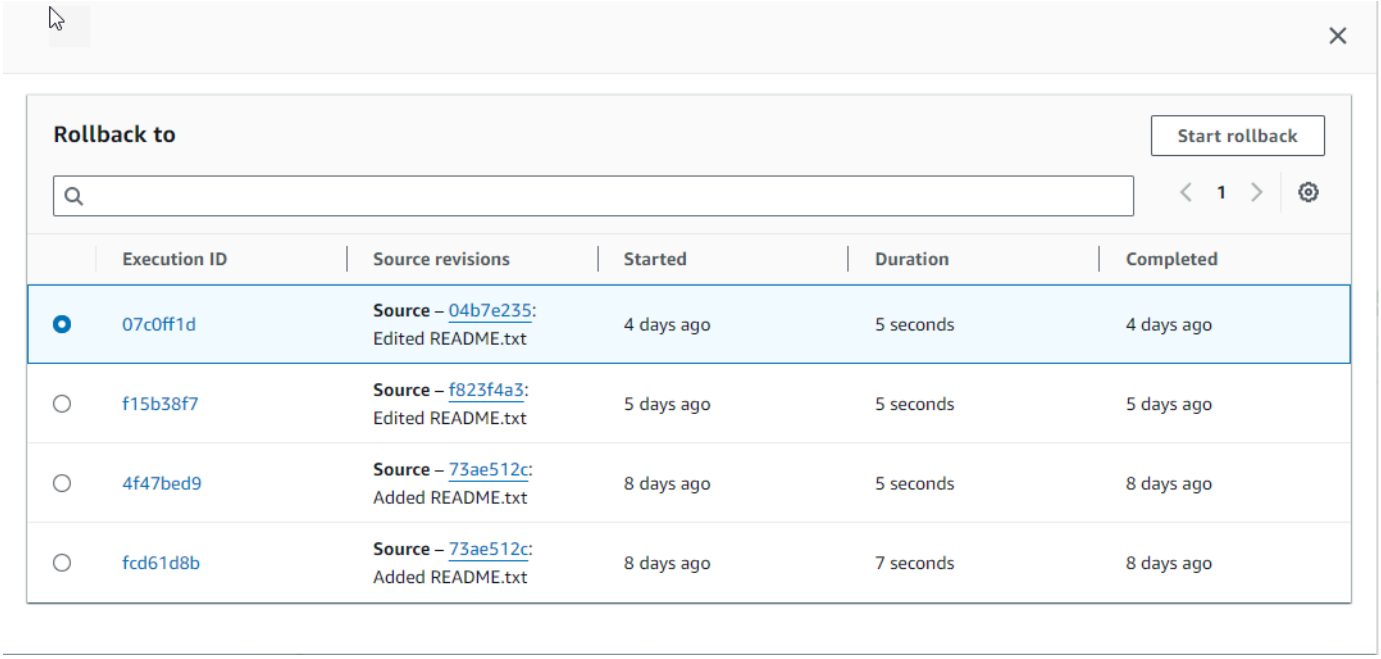
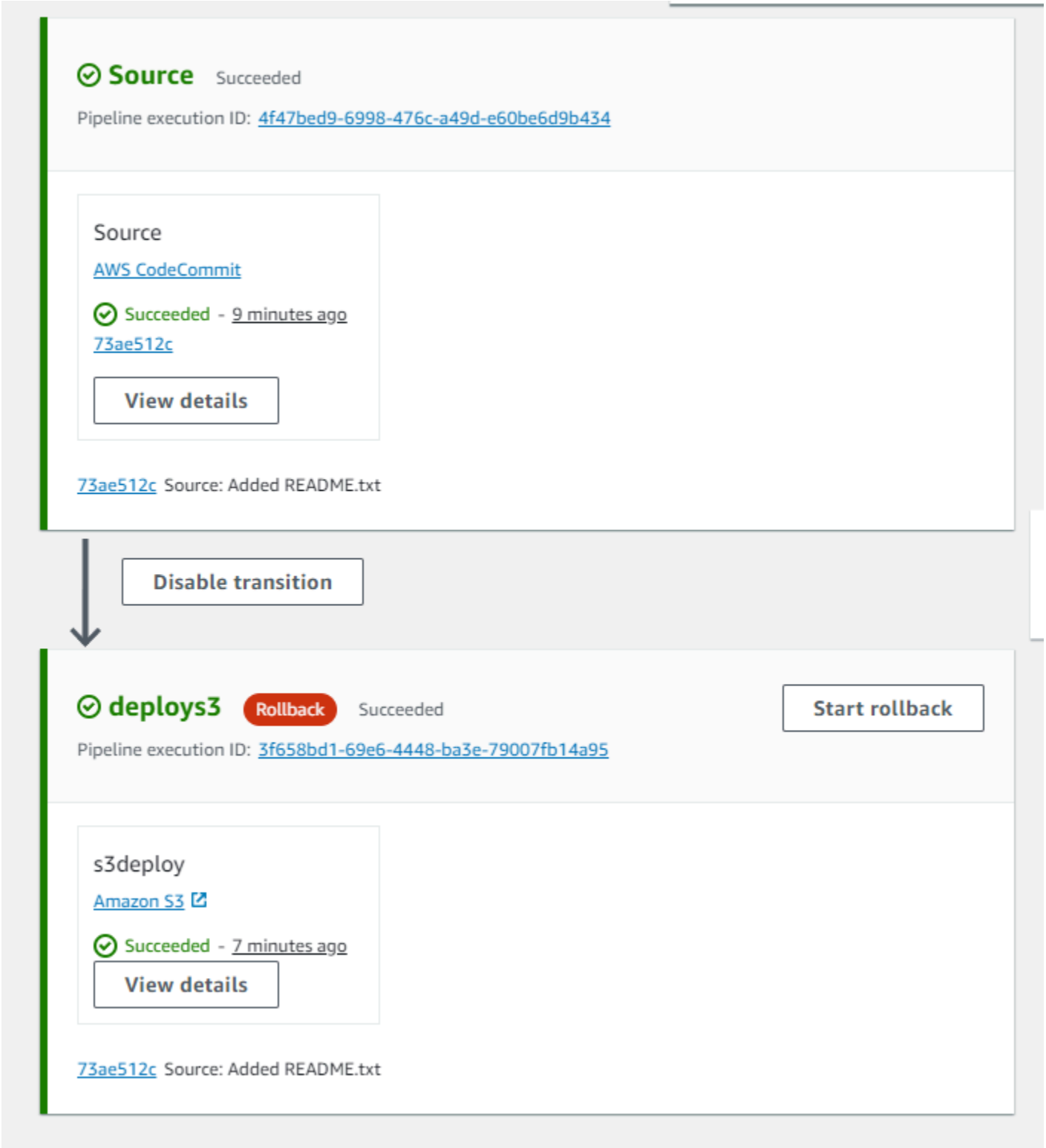


Diagram berikut menunjukkan contoh tahap yang digulung kembali dengan ID eksekusi baru.



Putar kembali panggung secara manual (CLI)

Untuk menggunakan AWS CLI untuk memutar kembali panggung secara manual, gunakan `rollback-stage` perintah.

Anda juga dapat memutar kembali panggung secara manual seperti yang dijelaskan dalamGulung kembali panggung secara manual.

> **Note**
>
> Daftar eksekusi pipa target yang tersedia adalah semua eksekusi dalam versi pipeline saat ini yang dimulai pada 1 Februari 2024.

Untuk memutar kembali panggung secara manual (CLI)

1. Perintah CLI untuk rollback manual akan memerlukan ID eksekusi dari eksekusi pipeline yang sebelumnya berhasil di panggung. Untuk mendapatkan ID eksekusi pipeline target yang akan Anda tentukan, gunakan list-pipeline-executions perintah dengan filter yang akan mengembalikan eksekusi yang berhasil di panggung. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan `list-pipeline-executions` perintah, menentukan nama pipeline dan filter untuk eksekusi yang berhasil di panggung. Dalam contoh ini, output akan mencantumkan eksekusi pipeline untuk pipeline bernama MyFirstPipeline dan untuk eksekusi yang berhasil dalam tahap bernama. `deploys3`

   ```
   aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline --filter
    succeededInStage={stageName=deploys3}
   ```

   Dalam output, salin ID eksekusi dari eksekusi yang sebelumnya berhasil yang ingin Anda tentukan untuk rollback. Anda akan menggunakan ini di langkah berikutnya sebagai ID eksekusi target.

2. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan `rollback-stage` perintah, menentukan nama pipeline, nama panggung, dan eksekusi target yang ingin Anda putar kembali. Misalnya, untuk memutar kembali tahap bernama Deploy untuk pipeline bernama*`MyFirstPipeline`*:

   ```
   aws codepipeline rollback-stage --pipeline-name MyFirstPipeline --stage-name Deploy
    --target-pipeline-execution-id bc022580-4193-491b-8923-9728dEXAMPLE
   ```

   Output mengembalikan ID eksekusi untuk eksekusi rolled-back baru. Ini adalah ID terpisah yang menggunakan revisi sumber dan parameter eksekusi target yang ditentukan.

Konfigurasikan panggung untuk rollback otomatis

Anda dapat mengonfigurasi tahapan dalam pipeline untuk memutar kembali secara otomatis pada kegagalan. Ketika tahap gagal, panggung digulung kembali ke eksekusi sukses terbaru. Pipeline hanya dapat memutar kembali ke eksekusi sebelumnya jika eksekusi sebelumnya dimulai dalam versi struktur pipa saat ini. Karena, konfigurasi rollback otomatis adalah bagian dari definisi pipeline, tahap pipeline Anda akan otomatis rollback hanya setelah eksekusi pipeline berhasil dalam tahap pipeline.

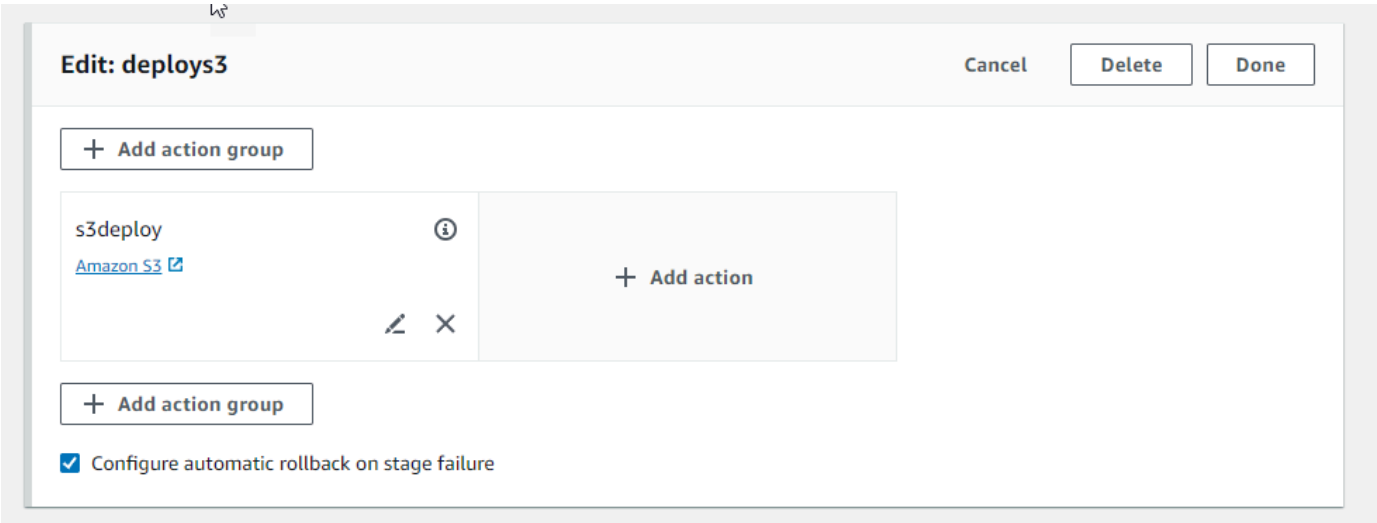
Konfigurasikan panggung untuk rollback otomatis (konsol)

Anda dapat memutar kembali tahap ke eksekusi sukses sebelumnya yang ditentukan. Untuk informasi selengkapnya, lihat RollbackStagedi Panduan CodePipeline API.

Konfigurasikan panggung untuk rollback otomatis (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda edit.
3. Pada halaman detail pipeline, pilih Edit.
4. Pada halaman Edit, untuk tindakan yang ingin Anda edit, pilih Edit tahap.
5. Pilih Konfigurasi tahap otomatis:, lalu pilih Konfigurasi rollback otomatis pada kegagalan panggung. Simpan perubahan pada pipeline Anda.



Konfigurasikan panggung untuk rollback otomatis (CLI)

Untuk menggunakan AWS CLI untuk mengkonfigurasi tahap gagal untuk secara otomatis memutar kembali ke eksekusi sukses terbaru, gunakan perintah untuk membuat atau memperbarui pipeline seperti yang dijelaskan dalam Buat pipeline, tahapan, dan tindakan danEdit pipa di CodePipeline.

- Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan `update-pipeline` perintah, menentukan kondisi kegagalan dalam struktur pipa. Contoh berikut mengonfigurasi rollback otomatis untuk nama yang dipentaskan: `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"
                }
            }
```

Untuk informasi selengkapnya tentang mengonfigurasi kondisi kegagalan untuk rollback tahap, lihat FailureConditionsdi Referensi API. CodePipeline

Konfigurasikan panggung untuk rollback otomatis ()CloudFormation

Untuk digunakan CloudFormation untuk mengkonfigurasi panggung untuk memutar kembali secara otomatis pada kegagalan, gunakan `OnFailure` parameter. Pada kegagalan, panggung akan secara otomatis memutar kembali ke eksekusi sukses terbaru.

```
OnFailure:
     Result: ROLLBACK
```

- Perbarui template seperti yang ditunjukkan pada cuplikan berikut. Contoh berikut mengonfigurasi rollback otomatis untuk nama yang dipentaskan: `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'
```

Untuk informasi selengkapnya tentang mengonfigurasi kondisi kegagalan untuk rollback tahap, lihat OnFailure*StageDeclaration*di bawah Panduan Pengguna.CloudFormation

Lihat status rollback dalam daftar eksekusi

Anda dapat melihat status dan ID eksekusi target untuk eksekusi rollback.

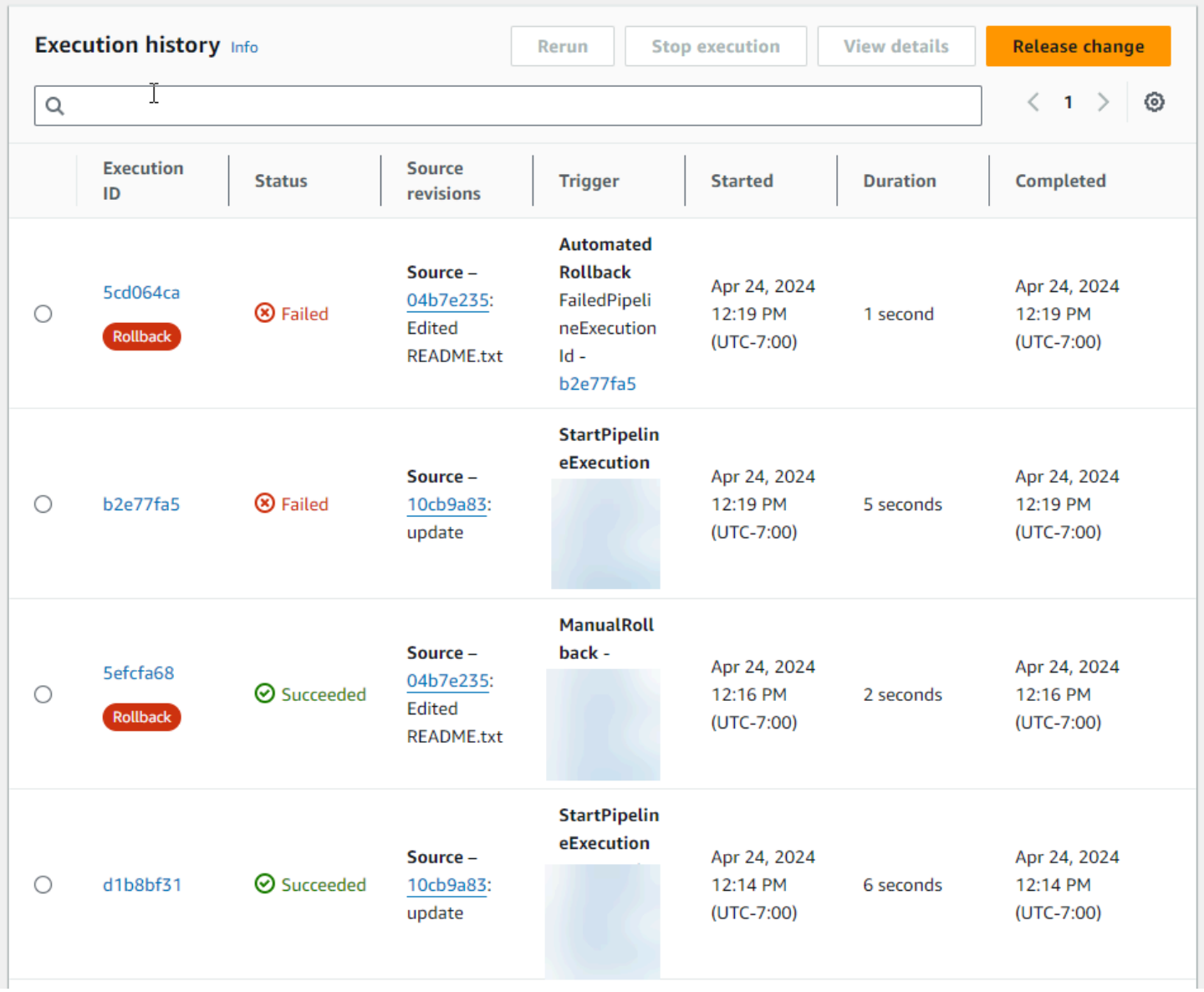
Lihat status rollback dalam daftar eksekusi (konsol)

Anda dapat menggunakan konsol untuk melihat status dan ID eksekusi target untuk eksekusi rollback dalam daftar eksekusi.

Lihat status eksekusi rollback dalam daftar eksekusi (konsol)

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua saluran pipa yang terkait dengan Anda Akun AWS ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda lihat.
3. Pilihriwayat. Daftar eksekusi menunjukkan label Rollback.



Pilih ID eksekusi yang ingin Anda lihat detailnya.

Lihat status rollback dengan (**`list-pipeline-executions`**CLI)

Anda dapat menggunakan CLI untuk melihat status dan ID eksekusi target untuk eksekusi rollback.

- Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan perintah: list-pipeline-executions

  ```
  aws codepipeline list-pipeline-executions --pipeline-name MyFirstPipeline
  ```

  Perintah ini mengembalikan daftar semua eksekusi selesai yang terkait dengan pipeline.

Contoh berikut menunjukkan data yang dikembalikan untuk pipeline bernama *MyFirstPipeline* tempat eksekusi rollback memulai 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"
                }
            ],
```

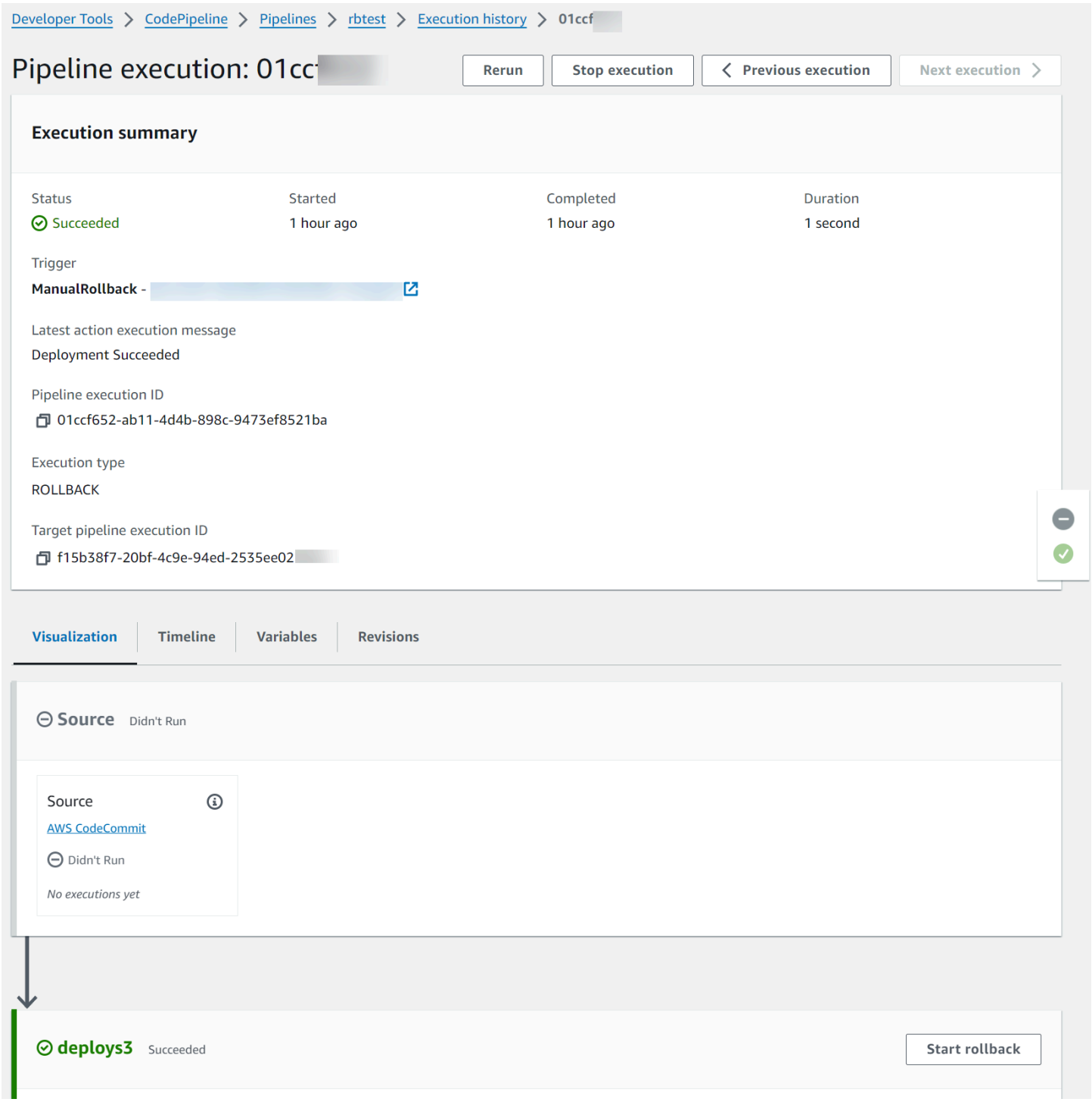
```
            "trigger": {
                "triggerType": "StartPipelineExecution",
                "triggerDetail": "arn:aws:sts::<account_ID>:assumed-role/<role>"
            },
            "executionMode": "SUPERSEDED"
        },
        {
            "pipelineExecutionId": "5cd064ca-bff7-425f-8653-f41d9EXAMPLE",
            "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"
            }
         },
```

Lihat detail status rollback

Anda dapat melihat status dan ID eksekusi target untuk eksekusi rollback.

Lihat status rollback pada halaman detail (konsol)

Anda dapat menggunakan konsol untuk melihat status dan target ID eksekusi pipeline untuk eksekusi rollback.



Lihat detail rollback dengan (**`get-pipeline-execution`**CLI)

Eksekusi pipeline yang telah diputar kembali akan ditampilkan dalam output untuk mendapatkan eksekusi pipeline.

- Untuk melihat detail tentang pipeline, jalankan get-pipeline-execution perintah, dengan menentukan nama unik pipeline. Misalnya, untuk melihat detail tentang pipeline bernama*MyFirstPipeline*, masukkan yang berikut ini:

```
aws codepipeline get-pipeline-execution --pipeline-name MyFirstPipeline --pipeline-
execution-id 3f658bd1-69e6-4448-ba3e-79007EXAMPLE
```

Perintah ini mengembalikan struktur pipa.

Contoh berikut menunjukkan data yang dikembalikan untuk sebagian dari pipeline bernama*MyFirstPipeline*, di mana ID eksekusi rollback dan metadata ditampilkan.

```
{
    "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"
        }
    }
}
```

Lihat status rollback dengan (**`get-pipeline-state`**CLI)

Eksekusi pipa yang telah diputar kembali akan ditampilkan dalam output untuk mendapatkan status pipa.

- Untuk melihat detail tentang pipeline, jalankan get-pipeline-state perintah, dengan menentukan nama unik pipeline. Misalnya, untuk melihat detail status tentang pipeline bernama*`MyFirstPipeline`*, masukkan yang berikut ini:

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

Contoh berikut menunjukkan data yang dikembalikan dengan tipe eksekusi 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"
}
```

Konfigurasikan kondisi untuk sebuah panggung

Anda dapat menentukan kondisi untuk tahap, seperti memeriksa variabel tertentu dalam proses pipeline, dan kemudian melibatkan hasil untuk kondisi tersebut, seperti melewatkan tahap atau gagal tahap. Pipeline dapat diatur untuk memeriksa kondisi tahap selama proses, di mana Anda menentukan pemeriksaan untuk tahap dan kemudian menentukan bagaimana tahap harus dilanjutkan ketika kondisi tertentu terpenuhi. Ketentuan berisi satu atau lebih aturan yang tersedia dalam daftar aturan di CodePipeline. Jika semua aturan dalam suatu kondisi berhasil, maka kondisinya terpenuhi. Anda dapat mengonfigurasi kondisi sehingga ketika kriteria tidak terpenuhi, hasil yang ditentukan akan terlibat.

Setiap kondisi memiliki set aturan yang merupakan seperangkat aturan yang diurutkan yang dievaluasi bersama. Karena itu, jika satu aturan gagal dalam kondisi tersebut, maka kondisinya gagal. Anda dapat mengganti kondisi aturan saat runtime pipeline.

Kondisi digunakan untuk jenis ekspresi tertentu dan masing-masing memiliki opsi khusus untuk hasil yang tersedia sebagai berikut:

- Entri - Ketentuan untuk melakukan pemeriksaan yang, jika terpenuhi, memungkinkan masuk ke suatu panggung. Aturan terlibat dengan opsi hasil berikut: Gagal atau Lewati
- Pada Kegagalan - Kondisi untuk melakukan pemeriksaan untuk tahap ketika gagal. Aturan terlibat dengan opsi hasil berikut: Rollback
- Tentang Sukses - Kondisi untuk melakukan pemeriksaan untuk tahap ketika berhasil. Aturan terlibat dengan opsi hasil berikut: Rollback atau Gagal

Kondisi didukung oleh seperangkat aturan untuk setiap jenis kondisi.

Untuk setiap jenis kondisi, ada tindakan spesifik yang diatur oleh kondisi tersebut. Tindakan tersebut adalah hasil dari pemeriksaan kondisi yang berhasil atau gagal. Misalnya, kondisi untuk entri (kondisi entri) menemui alarm (aturan), maka cek berhasil dan hasil (tindakan) adalah entri tahap diblokir.

Anda juga dapat menggunakan AWS CodePipeline konsol atau AWS CLI untuk memutar kembali secara manual atau mencoba lagi tahap atau tindakan dalam satu tahap. Lihat Konfigurasikan kondisi untuk sebuah panggung.

Topik

- Gunakan kasus untuk kondisi panggung

- Pertimbangan untuk hasil yang dikonfigurasi untuk kondisi panggung
- Pertimbangan untuk aturan yang dikonfigurasi untuk kondisi panggung
- Membuat kondisi Entri
- Menciptakan kondisi Kegagalan
- Menciptakan kondisi Sukses
- Menghapus kondisi panggung
- Kondisi panggung utama

Gunakan kasus untuk kondisi panggung

Kondisi panggung memiliki beberapa kasus penggunaan untuk mengatur pelepasan dan perubahan keamanan di saluran pipa. Berikut ini adalah contoh kasus penggunaan untuk kondisi panggung.

- Gunakan kondisi Entri untuk menentukan kondisi yang akan memeriksa status CloudWatch alarm, yang kemudian akan memblokir perubahan jika lingkungan produksi tidak dalam keadaan sehat.
- Gunakan kondisi Entri dengan waktu tunggu 60 untuk menentukan kondisi yang akan dievaluasi ketika semua tindakan dalam tahap telah berhasil diselesaikan, dan kemudian putar kembali perubahan jika CloudWatch alarm masuk ke status ALARM dalam waktu 60 menit.
- Gunakan kondisi On Success untuk menentukan kondisi sehingga ketika tahap selesai dengan sukses, aturan akan memeriksa apakah waktu saat ini ada di jendela penerapan dan kemudian menyebarkan jika aturan berhasil.

Pertimbangan untuk hasil yang dikonfigurasi untuk kondisi panggung

Pertimbangan untuk kondisi panggung adalah sebagai berikut:

- Anda tidak dapat menggunakan percobaan ulang tahap otomatis dengan kondisi OnFailure.
- Saat mengonfigurasi kondisi dengan hasil Rollback, tahap hanya dapat memutar kembali ke eksekusi sebelumnya jika tersedia dalam versi struktur pipa saat ini.
- Saat mengonfigurasi kondisi dengan hasil Rollback, Anda tidak dapat memutar kembali ke ID eksekusi target yang merupakan tipe eksekusi rollback.
- Untuk kondisi Entri yang menggunakan hasil Lewati untuk melewati tahap jika kondisi gagal, hanya **`VariableCheck`** aturan **`LambdaInvoke`** dan yang didukung.

- Anda tidak dapat melakukan percobaan ulang tahap manual di panggung dalam status Skipped.
- Anda tidak dapat melakukan rollback manual ke tahap dalam status Skipped.
- Anda tidak dapat mengganti kondisi jika kondisi dikonfigurasi dengan hasil Lewati.
- Dengan pengecualian hasil Lewati, Anda dapat mengganti kondisi tahap saat memulai eksekusi pipeline. Untuk kondisi tahap di mana override dilibatkan, eksekusi akan dilakukan seperti yang dijelaskan dalam tabel berikut.

Tipe	Hasil yang dikonfigurasi pada kegagalan kondisi	Status panggung	Mengesampingkan perilaku
Entry	Gagal	Sedang berlangsung	Panggung berlangsung.
Entry	Lewati	Dilewati	Tidak berlaku.
OnFailure	Rollback	Gagal	Panggung gagal.
OnSuccess	Rollback	Berhasil	Panggung berlangsung.
OnSuccess	Gagal	Gagal	Panggung berlangsung.

Pertimbangan untuk aturan yang dikonfigurasi untuk kondisi panggung

Pertimbangan untuk aturan yang tersedia untuk kondisi tahap adalah sebagai berikut:

- Untuk `LambdaInvoke` aturan, Anda harus terlebih dahulu mengkonfigurasi fungsi Lambda untuk digunakan dalam aturan. Siapkan fungsi Lambda ARN saat Anda mengonfigurasi aturan.
- Untuk `CloudWatchAlarm` aturan, Anda harus terlebih dahulu mengkonfigurasi CloudWatch acara Acara yang akan digunakan dalam aturan. Siapkan acara ARN saat Anda mengonfigurasi aturan.

Membuat kondisi Entri

Anda dapat mengonfigurasi kondisi Entri untuk panggung menggunakan konsol atau CLI. Anda akan mengonfigurasi aturan dan hasil yang sesuai untuk setiap kondisi. Untuk hasil rollback, pipeline hanya dapat memutar kembali ke eksekusi sebelumnya jika eksekusi sebelumnya dimulai dalam versi struktur pipa saat ini.

Langkah-langkah memberikan contoh kondisi Entri yang menggunakan aturan monitor.

Untuk informasi selengkapnya, lihat Kondisi RuleTypeId, dan RuleExecutiondi Panduan CodePipeline API.

Membuat kondisi Entri - contoh CloudWatchAlarm aturan (Konsol)

Anda dapat mengonfigurasi kondisi Entri untuk tahap, bersama dengan aturan dan hasil yang Anda inginkan untuk dilakukan tahapan saat kondisi terpenuhi.

Konfigurasikan kondisi Entri (konsol)

1. Lengkapi prasyarat apa pun, seperti membuat sumber daya dan ARN untuk aturan di mana sumber daya disediakan, seperti. AWS CloudWatchAlarm
2. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua saluran pipa yang terkait dengan Anda Akun AWS ditampilkan.
3. Di Nama, pilih nama pipeline yang ingin Anda edit.
4. Pada halaman detail pipeline, pilih Edit.
5. Pada halaman Edit, untuk tindakan yang ingin Anda edit, pilih Edit tahap.
6. Pilih Tambahkan kondisi entri. Kartu kondisi entri tahap Sebelum ditampilkan dengan opsi Gagal yang tersedia untuk kondisi ini.
7. Pilih Tambahkan aturan, lalu selesaikan yang berikut ini.

   a. Di Nama aturan, masukkan nama untuk aturan Anda. Untuk contoh ini, masukkan`MyAlarmRule`.

   b. Di Penyedia aturan, pilih penyedia aturan yang telah dikonfigurasi sebelumnya untuk ditambahkan ke kondisi Anda. Untuk contoh ini, pilih AWS CloudWatchAlarm, lalu selesaikan langkah-langkah berikut.

   c. Di Wilayah, pilih Wilayah untuk kondisi Anda atau biarkan default.

d. Dalam nama Alarm, pilih CloudWatch sumber daya yang akan digunakan untuk aturan. Anda harus sudah membuat sumber daya di akun Anda.

e. (Opsional) Dalam Waktu tunggu, masukkan jumlah waktu yang CodePipeline akan menunggu jika alarm dalam keadaan ALARM saat pertama kali dievaluasi. Jika alarm status OK ketika aturan pertama kali diperiksa, aturan akan segera berhasil.

f. (Opsional) Masukkan status alarm tertentu untuk dipantau, dan masukkan peran ARN jika sesuai.

g. Setelah selesai mengedit panggung, pilih Selesai. Pada halaman edit pipeline, pilih Simpan.

8. Setelah lari, lihat hasilnya.

Membuat kondisi Entri dengan Lewati hasil dan **`VariableCheck`** aturan (konsol)

Anda dapat mengonfigurasi kondisi Entri untuk suatu tahap sehingga jika kondisi entri tidak terpenuhi, tahap dilewati Jika kondisi gagal, maka hasilnya terlibat dan tahap dilewati. Ketika sebuah panggung dilewati, status panggung dilewati, dan status tindakannya tidak dijalankan. Untuk pertimbangan kondisi panggung dengan hasil Lewati, lihatPertimbangan untuk hasil yang dikonfigurasi untuk kondisi panggung.

Dalam contoh berikut, aturan pemeriksaan variabel menemukan bahwa nilainya tidak cocok, dan tahap build dilewati.

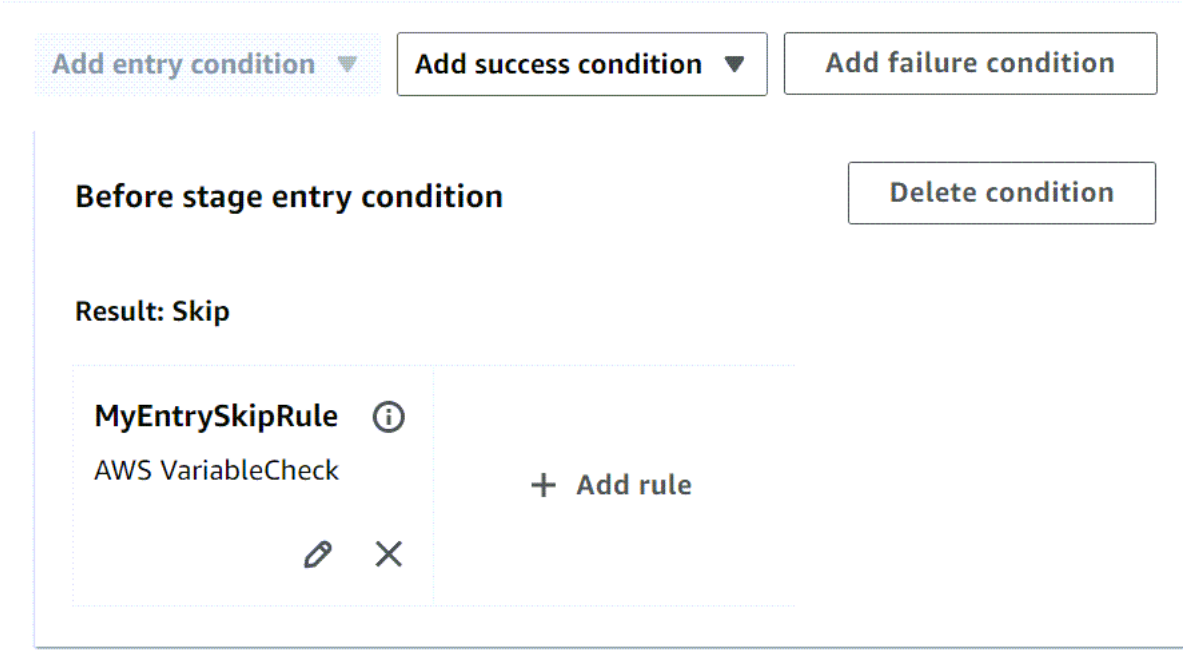
Konfigurasikan kondisi Entri dengan hasil Lewati (konsol)

1. Lengkapi prasyarat apa pun, seperti membuat sumber daya dan ARN untuk aturan di mana sumber daya disediakan, seperti. AWS CloudWatchAlarm
2. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua saluran pipa yang terkait dengan Anda Akun AWS ditampilkan.
3. Di Nama, pilih nama pipeline yang ingin Anda edit.
4. Pada halaman detail pipeline, pilih Edit.
5. Pada halaman Edit, untuk tindakan yang ingin Anda edit, pilih Edit tahap.
6. Pilih Tambahkan kondisi entri, lalu pilih Lewati sebagai hasilnya.
7. Pilih Tambahkan aturan, lalu selesaikan yang berikut ini.

a. Di Nama aturan, masukkan nama untuk aturan Anda. Untuk contoh ini, masukkan`MyAlarmRule`.

b. Di Penyedia aturan, pilih penyedia aturan yang telah dikonfigurasi sebelumnya untuk ditambahkan ke kondisi Anda. Untuk contoh ini, pilih VariableCheck, lalu selesaikan langkah-langkah berikut.

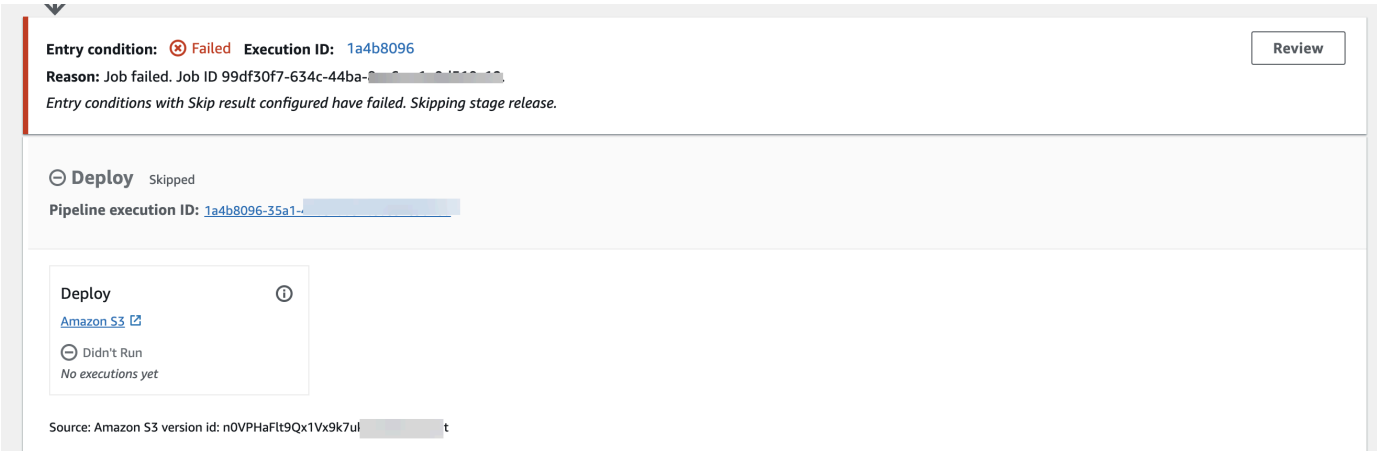


c. Di Wilayah, pilih Wilayah untuk kondisi Anda atau biarkan default.

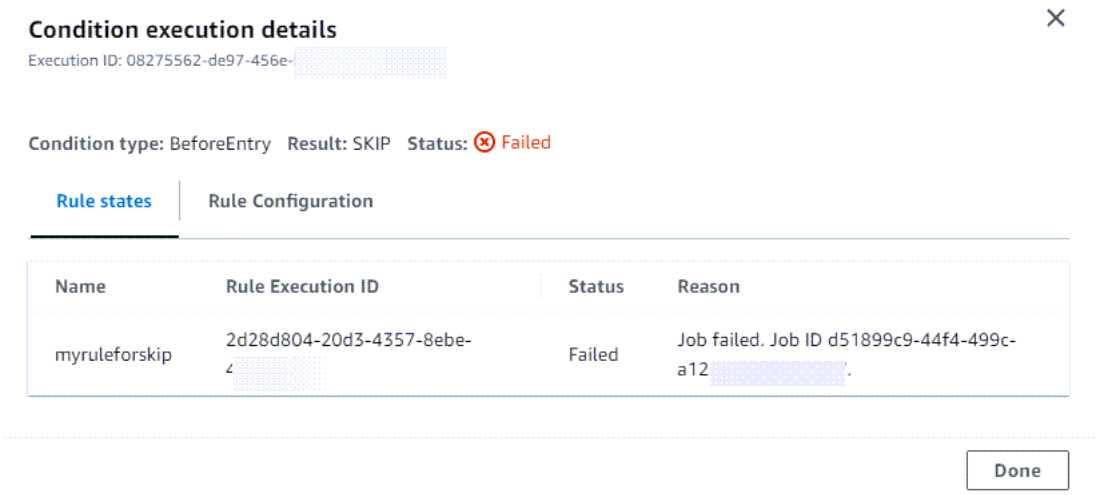
d. Di Variable, pilih variabel yang akan dibandingkan, seperti `#{SourceVariables.FullRepositoryName}` untuk pipeline yang memiliki aksi sumber GitHub (via GitHub App). Masukkan nama repositori dan pilih operator, seperti Equals.

e. Setelah selesai mengedit panggung, pilih Selesai. Pada halaman edit pipeline, pilih Simpan.

8. Setelah lari, lihat hasilnya.



9. Untuk meninjau detail, pilih Tinjau. Detail dalam contoh berikut menunjukkan bahwa hasil yang dikonfigurasi untuk kondisi tersebut adalah Lewati, yang tidak dapat diganti. Status aturan Gagal karena kondisi tidak terpenuhi.



Membuat kondisi Entri (CLI)

Untuk menggunakan AWS CLI untuk mengkonfigurasi kondisi Entri, gunakan perintah untuk membuat atau memperbarui pipeline seperti yang dijelaskan dalam Buat pipeline, tahapan, dan tindakan danEdit pipa di CodePipeline.

Konfigurasikan kondisi dan aturan atau aturan (CLI)

- Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan `update-pipeline` perintah, menentukan kondisi kegagalan dalam struktur pipa. Contoh berikut mengonfigurasi kondisi Entri untuk nama bertahap: `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"
                    }
                ]
            }
        ]
    }
}
```

Untuk informasi selengkapnya tentang mengonfigurasi kondisi keberhasilan untuk rollback tahap, lihat SuccessConditionsdi Referensi API. CodePipeline

Membuat kondisi Entri (CFN)

Untuk menggunakan CloudFormation untuk mengkonfigurasi kondisi Entri, gunakan `beforeEntry` parameter. Saat masuk, panggung akan menjalankan aturan dan melakukan hasilnya.

```
beforeEntry:
     Result: FAIL
```

- Perbarui template seperti yang ditunjukkan pada cuplikan berikut. Contoh berikut mengkonfigurasi kondisi Entri dengan aturan bernama`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
```

Untuk informasi selengkapnya tentang mengonfigurasi kondisi beforeEntry, lihat AWS::CodePipeline::Pipeline BeforeEntryConditions*StageDeclaration*di bawah Panduan Pengguna.CloudFormation

Menciptakan kondisi Kegagalan

Anda dapat mengonfigurasi kondisi On Failure untuk panggung menggunakan konsol atau CLI. Anda akan mengonfigurasi aturan dan hasil yang sesuai untuk setiap kondisi. Untuk hasil rollback, pipeline hanya dapat memutar kembali ke eksekusi sebelumnya jika eksekusi sebelumnya dimulai dalam versi struktur pipa saat ini.

Membuat kondisi On Failure (Konsol)

Anda dapat mengonfigurasi kondisi On Failure untuk sebuah panggung, bersama dengan aturan dan hasil yang Anda inginkan untuk dilakukan tahapan saat kondisi terpenuhi.

Konfigurasikan kondisi On Failure (konsol)

1. Lengkapi prasyarat apa pun, seperti membuat sumber daya dan ARN untuk aturan di mana sumber daya disediakan, seperti aturan. LambdaInvoke
2. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua saluran pipa yang terkait dengan Anda Akun AWS ditampilkan.
3. Di Nama, pilih nama pipeline yang ingin Anda edit.
4. Pada halaman detail pipeline, pilih Edit.
5. Pada halaman Edit, untuk tindakan yang ingin Anda edit, pilih Edit tahap.
6. Pilih Tambahkan kondisi kegagalan. Kartu kondisi Kegagalan ditampilkan dengan opsi Rollback yang tersedia untuk kondisi ini.
7. Pilih Tambahkan aturan, lalu selesaikan yang berikut ini.

   a. Di Nama aturan, masukkan nama untuk aturan Anda. Untuk contoh ini, masukkan`MyLambdaRule`.

b. Di Penyedia aturan, pilih penyedia aturan yang telah dikonfigurasi sebelumnya untuk ditambahkan ke kondisi Anda. Untuk contoh ini, pilih AWS LambdaInvoke, lalu selesaikan langkah-langkah berikut.

c. Di Wilayah, pilih Wilayah untuk kondisi Anda atau biarkan default.

d. Di artefak Input, pilih artefak sumber.

e. Di nama Fungsi, pilih sumber daya Lambda yang akan digunakan untuk aturan. Anda harus sudah membuat sumber daya di akun Anda.

f. (Opsional) Dalam parameter Pengguna, masukkan pasangan apa pun yang mewakili parameter untuk konfigurasi tambahan.

g. (Opsional) Di Role Arn, masukkan peran ARN jika dikonfigurasi.

h. (Opsional) Di Timeout in Minutes, masukkan waktu dalam menit yang harus ditunggu aturan sebelum batas waktu.

i. Setelah selesai mengedit panggung, pilih Selesai. Pada halaman edit pipeline, pilih Simpan.

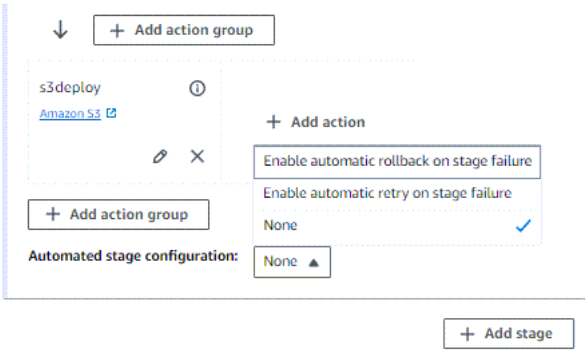
Membuat kondisi onFailure dengan contoh hasil Coba Ulang (Konsol)

Anda dapat mengonfigurasi kondisi onFailure untuk tahap sehingga jika kondisi entri tidak terpenuhi, tahap dicoba lagi. Sebagai bagian dari hasil ini, Anda mengonfigurasi mode coba lagi, menentukan apakah akan mencoba lagi tindakan yang gagal atau mencoba lagi tahap yang gagal.

Konfigurasikan kondisi onFailure dengan hasil Coba Ulang (konsol)

1. Lengkapi prasyarat apa pun, seperti membuat sumber daya dan ARN untuk aturan di mana sumber daya disediakan, seperti. AWS CloudWatchAlarm
2. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua saluran pipa yang terkait dengan Anda Akun AWS ditampilkan.
3. Di Nama, pilih nama pipeline yang ingin Anda edit.
4. Pada halaman detail pipeline, pilih Edit.
5. Pada halaman Edit, untuk tindakan yang ingin Anda edit, pilih Edit tahap.
6. Di bagian bawah panggung, di bawah Konfigurasi tahap otomatis:, pilih Aktifkan coba ulang otomatis pada kegagalan panggung. Dalam mode Coba lagi, pilih salah satu Coba lagi tahap gagal atau Coba lagi tindakan yang gagal.



7. Pilih untuk menambahkan kondisi OnFailure dan kemudian pilih Tambah aturan dan masukkan aturan untuk kondisi tersebut.

   a. Di Nama aturan, masukkan nama untuk aturan Anda. Untuk contoh ini, masukkan`MyAlarmRule`.

   b. Di Penyedia aturan, pilih penyedia aturan yang telah dikonfigurasi sebelumnya untuk ditambahkan ke kondisi Anda. Untuk contoh ini, pilih CloudWatchAlarm, lalu selesaikan langkah-langkah berikut.

   c. Di Wilayah, pilih Wilayah untuk kondisi Anda atau biarkan default.

   d. Di Nama Alarm, pilih sumber daya yang dikonfigurasi untuk peringatan.

   e. Setelah selesai mengedit panggung, pilih Selesai. Pada halaman edit pipeline, pilih Simpan.

8. Setelah lari, lihat hasilnya.

Membuat kondisi Kegagalan (CLI)

Untuk menggunakan AWS CLI untuk mengkonfigurasi kondisi On Failure, gunakan perintah untuk membuat atau memperbarui pipeline seperti yang dijelaskan dalam Buat pipeline, tahapan, dan tindakan danEdit pipa di CodePipeline.

Konfigurasikan kondisi dan aturan atau aturan (CLI)

- Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan `update-pipeline` perintah, menentukan kondisi kegagalan dalam struktur pipa. Contoh berikut mengonfigurasi kondisi On Failure untuk nama bertahap: `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"
                    }
                ]
            }
        ]
    }
}
```

Untuk informasi selengkapnya tentang mengonfigurasi kondisi kegagalan, lihat FailureConditionsdi Referensi CodePipeline API.

Membuat kondisi Kegagalan (CFN)

Untuk menggunakan CloudFormation untuk mengkonfigurasi kondisi On Failure, gunakan `OnFailure` parameter. Pada kesuksesan, panggung akan menjalankan aturan dan melakukan hasilnya.

```
OnFailure:
     Result: ROLLBACK
```

- Perbarui template seperti yang ditunjukkan pada cuplikan berikut. Contoh berikut mengkonfigurasi OnFailure kondisi dengan aturan bernama`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
```

Untuk informasi selengkapnya tentang mengonfigurasi kondisi kegagalan, lihat OnFailure`StageDeclaration`di bawah Panduan CloudFormation Pengguna.

Menciptakan kondisi Sukses

Anda dapat mengonfigurasi kondisi On Success untuk panggung menggunakan konsol atau CLI. Anda akan mengonfigurasi aturan dan hasil yang sesuai untuk setiap kondisi. Untuk hasil rollback, pipeline hanya dapat memutar kembali ke eksekusi sebelumnya jika eksekusi sebelumnya dimulai dalam versi struktur pipa saat ini.

Langkah-langkah tersebut memberikan contoh kondisi On Success yang menggunakan aturan jendela deployment.

Untuk informasi selengkapnya, lihat Kondisi RuleTypeId, dan RuleExecutiondi Panduan CodePipeline API.

Membuat kondisi Sukses (Konsol)

Anda dapat mengonfigurasi kondisi On Success untuk sebuah panggung, bersama dengan aturan dan hasil yang Anda inginkan untuk dilakukan panggung ketika kondisi terpenuhi.

Konfigurasikan kondisi On Success (konsol)

1. Lengkapi prasyarat apa pun, seperti membuat sumber daya dan ARN untuk aturan di mana sumber daya disediakan, seperti. AWS LambdaRule
2. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama dan status semua saluran pipa yang terkait dengan Anda Akun AWS ditampilkan.
3. Di Nama, pilih nama pipeline yang ingin Anda edit.
4. Pada halaman detail pipeline, pilih Edit.
5. Pada halaman Edit, untuk tindakan yang ingin Anda edit, pilih Edit tahap.
6. Pilih Tambahkan kondisi sukses. Kartu kondisi sukses di atas panggung ditampilkan. Pilih opsi Rollback atau Fail yang ditampilkan sebagai hasil yang tersedia untuk jenis kondisi ini.
7. Pilih Tambahkan aturan, lalu selesaikan yang berikut ini.

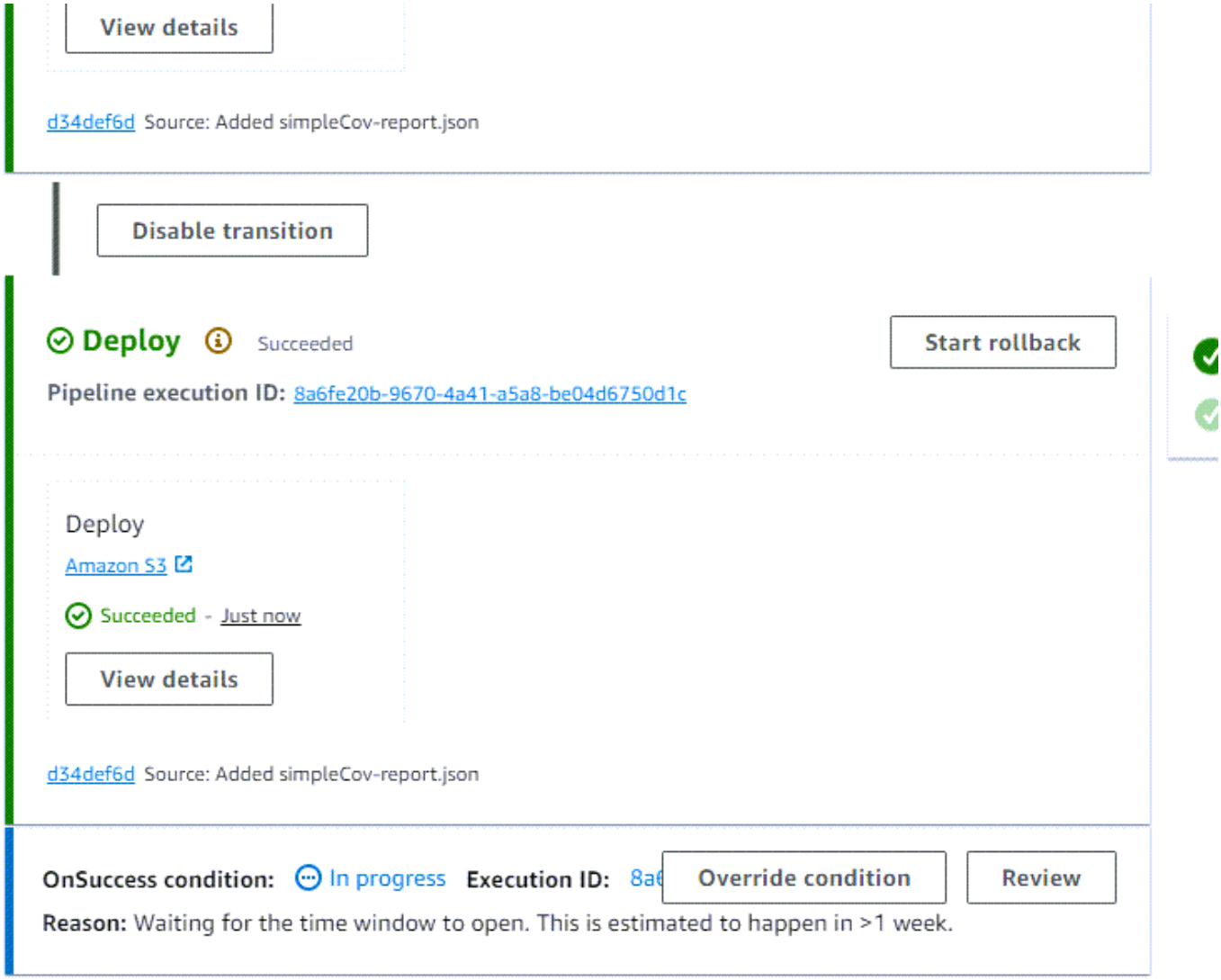
   a. Di Nama aturan, masukkan nama untuk kondisi Anda. Untuk contoh ini, masukkan`MyDeploymentRule`.

   b. Di Penyedia aturan, pilih aturan yang telah dikonfigurasi untuk ditambahkan ke kondisi Anda. Untuk contoh ini, pilih AWS DeploymentWindow, lalu selesaikan langkah-langkah berikut.

   c. Di Wilayah, pilih Wilayah untuk kondisi Anda atau biarkan default.

   d. Di Cron, masukkan ekspresi cron untuk jendela penerapan. Ekspresi cron mendefinisikan hari dan waktu ketika penerapan harus diizinkan. Untuk informasi referensi tentang ekspresi cron, lihat Menggunakan ekspresi cron dan rate untuk menjadwalkan aturan.

   e. (Opsional) Masuk TimeZone, masukkan zona waktu untuk jendela penyebaran.
8. Setelah lari, lihat hasilnya.



Menciptakan kondisi Sukses (CLI)

Untuk menggunakan AWS CLI untuk mengkonfigurasi kondisi On Success, gunakan perintah untuk membuat atau memperbarui pipeline seperti yang dijelaskan dalam Buat pipeline, tahapan, dan tindakan danEdit pipa di CodePipeline.

Konfigurasikan kondisi dan aturan atau aturan (CLI)

- Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan `update-pipeline` perintah, menentukan kondisi kegagalan dalam

struktur pipa. Contoh berikut mengonfigurasi kondisi On Success untuk nama bertahap`Deploy`, di mana aturan diberi nama: `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"
                    }
                ]
            }
        ]
    }
}
```

Untuk informasi selengkapnya tentang mengonfigurasi kondisi keberhasilan, lihat SuccessConditionsdi Referensi CodePipeline API.

Buat kondisi On Success (CFN)

Untuk menggunakan CloudFormation untuk mengkonfigurasi kondisi On Success, gunakan `OnSuccess` parameter. Pada kesuksesan, panggung akan menjalankan aturan dan melakukan hasilnya.

```
OnSuccess:
     Result: ROLLBACK
```

- Perbarui template seperti yang ditunjukkan pada cuplikan berikut. Contoh berikut mengkonfigurasi OnSuccess kondisi dengan aturan bernama`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
```

Untuk informasi selengkapnya tentang mengonfigurasi kondisi kegagalan untuk rollback tahap, lihat OnFailure*StageDeclaration*di bawah Panduan Pengguna.CloudFormation

Menghapus kondisi panggung

Anda dapat menghapus kondisi tahap yang telah dikonfigurasi untuk pipeline Anda.

Untuk menghapus kondisi panggung

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

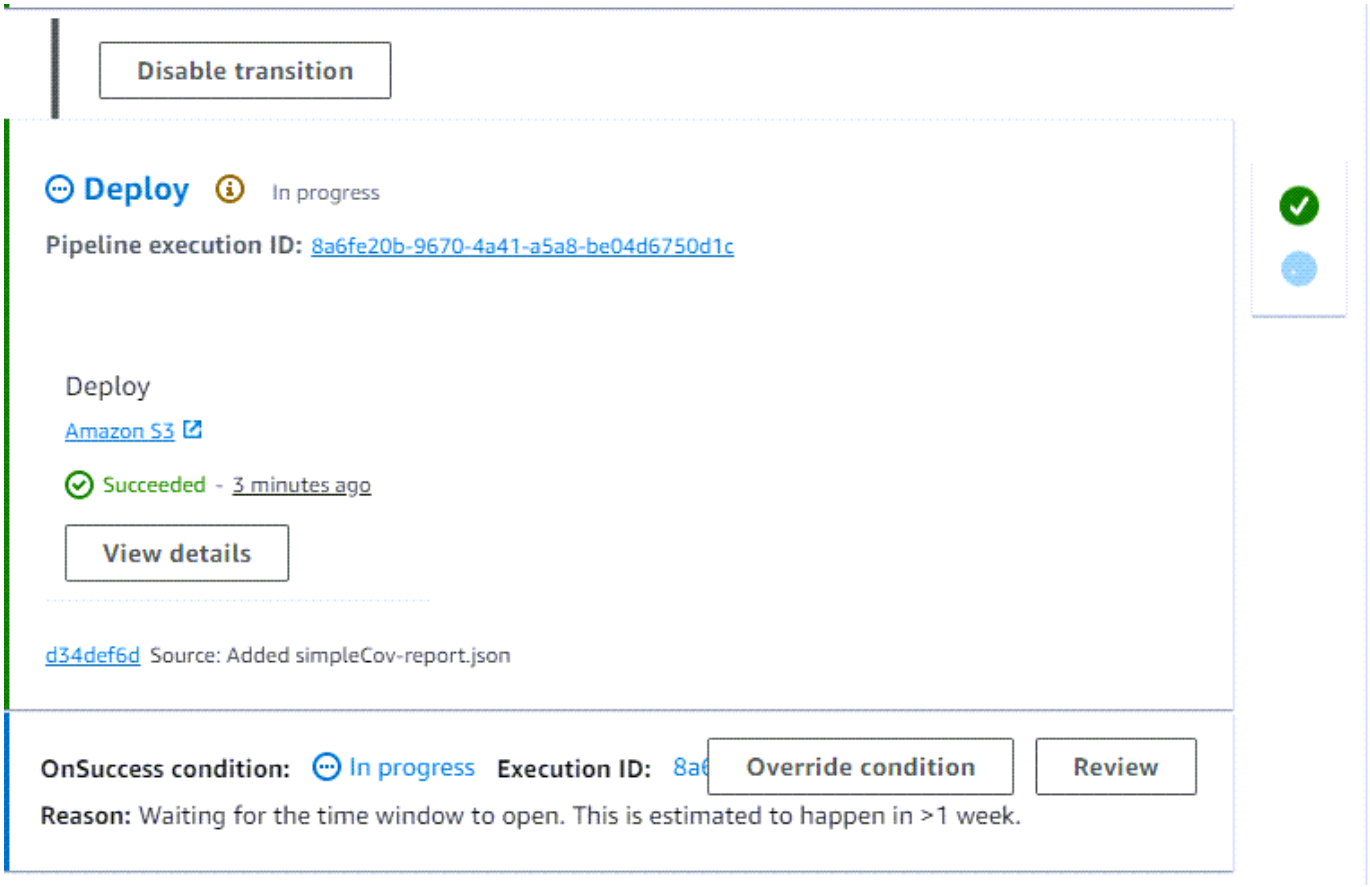
   Nama dan status semua saluran pipa yang terkait dengan Anda Akun AWS ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda edit.
3. Pada halaman detail pipeline, pilih Edit.
4. Pada halaman Edit, untuk kondisi yang ingin Anda edit, pilih Edit tahap.
5. Di samping kondisi yang ingin Anda hapus, pilih Delete condition.

Kondisi panggung utama

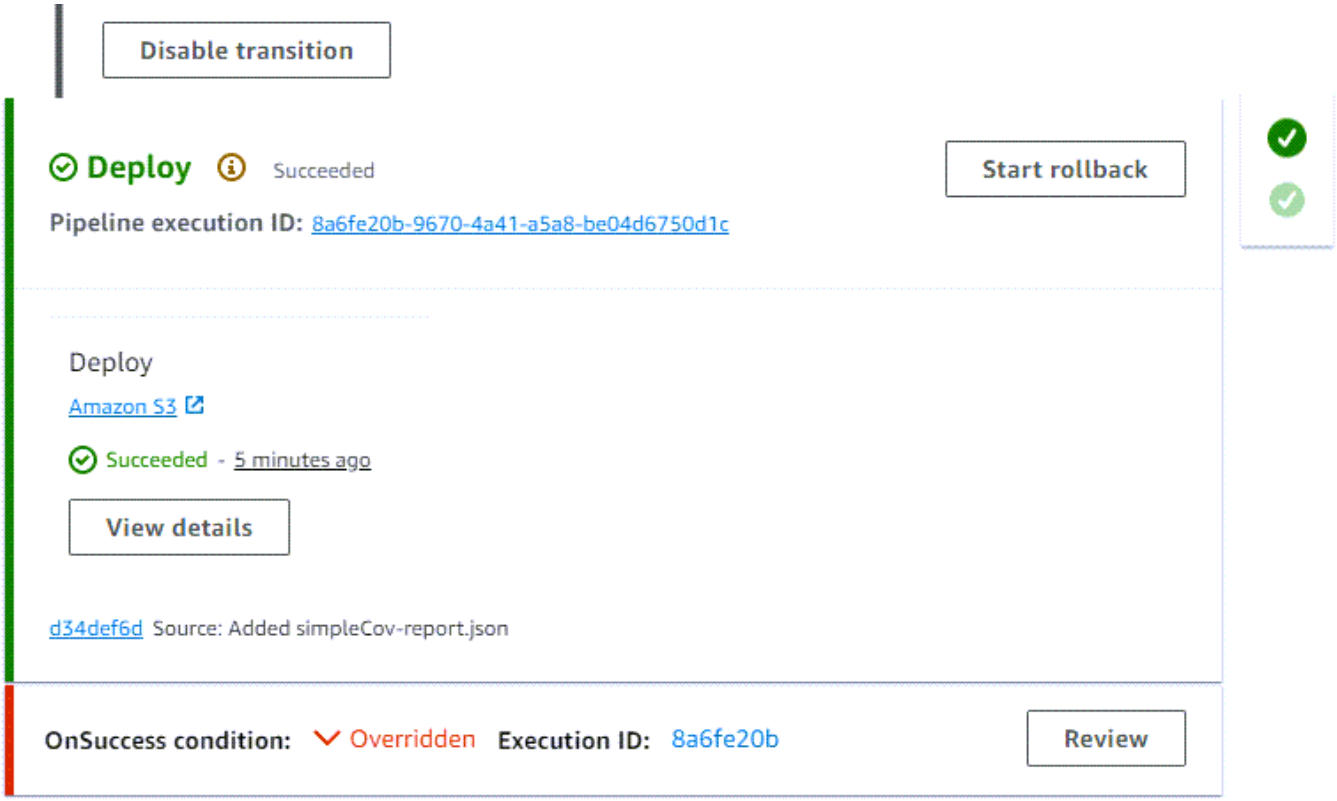
Anda dapat mengganti kondisi panggung yang telah dikonfigurasi untuk pipeline Anda. Di konsol, saat tahap dan aturan berjalan, Anda dapat memilih untuk mengganti kondisi panggung. Ini menghasilkan tahapan berjalan

Untuk mengesampingkan kondisi panggung

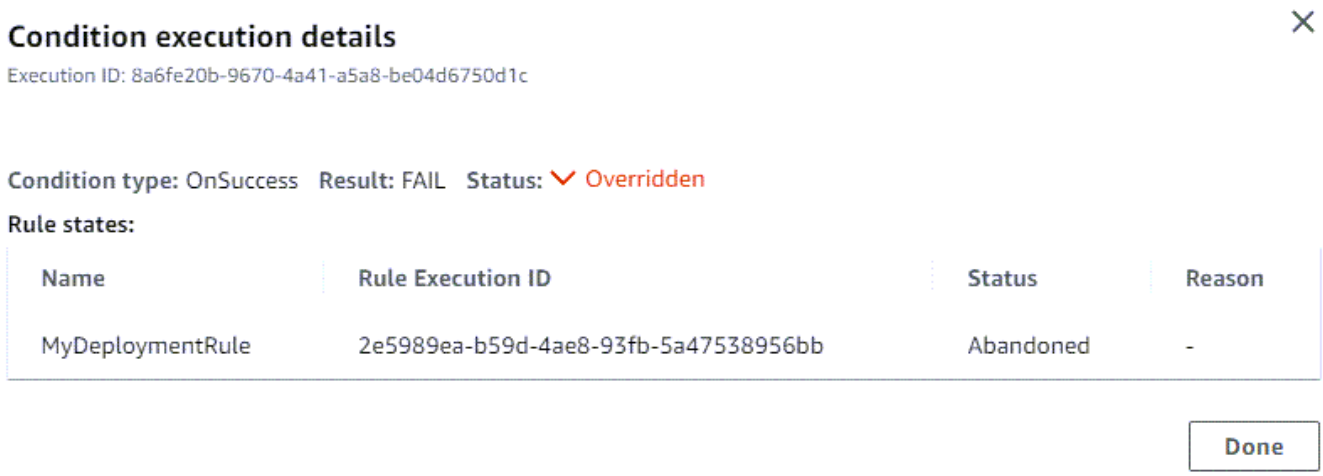
1. Dalam contoh ini, tahap pipa berjalan dengan suatu kondisi. Tombol Override diaktifkan.



2. Di samping kondisi yang ingin Anda ganti, pilih Override.



3. Untuk meninjau detail, pilih Tinjau. Detail dalam contoh berikut menunjukkan bahwa hasil yang dikonfigurasi untuk kondisi tersebut adalah Gagal, yang telah diganti. Status aturan Ditinggalkan karena penggantian.



Menggunakan jenis tindakan, tindakan kustom, dan tindakan persetujuan

Dalam AWS CodePipeline, tindakan adalah bagian dari urutan dalam tahap pipa. Ini adalah tugas yang dilakukan pada artefak di tahap itu. Tindakan pipa terjadi dalam urutan tertentu, secara berurutan atau paralel, sebagaimana ditentukan dalam konfigurasi panggung.

CodePipeline memberikan dukungan untuk enam jenis tindakan:

- Sumber
- Build
- Uji
- Deploy
- Persetujuan
- Panggil

Untuk informasi tentang produk Layanan AWS dan layanan mitra yang dapat Anda integrasikan ke dalam pipeline berdasarkan jenis tindakan, lihatIntegrasi dengan tipe CodePipeline tindakan.

Topik

- Bekerja dengan tipe tindakan
- Buat dan tambahkan tindakan kustom di CodePipeline
- Menandai tindakan kustom di CodePipeline
- Memanggil AWS Lambda fungsi dalam pipeline di CodePipeline
- Menambahkan tindakan persetujuan manual ke panggung
- Menambahkan tindakan Lintas wilayah di CodePipeline
- Bekerja dengan variabel

Bekerja dengan tipe tindakan

Jenis tindakan adalah tindakan yang telah dikonfigurasi sebelumnya yang Anda buat sebagai penyedia untuk pelanggan dengan menggunakan salah satu model integrasi yang didukung. AWS CodePipeline

Anda dapat meminta, melihat, dan memperbarui jenis tindakan. Jika tipe tindakan dibuat untuk akun Anda sebagai pemilik, Anda dapat menggunakan AWS CLI untuk melihat atau memperbarui properti dan struktur tipe tindakan Anda. Jika Anda adalah penyedia atau pemilik tipe tindakan, pelanggan Anda dapat memilih tindakan dan menambahkannya ke saluran pipa mereka setelah tersedia di CodePipeline.

> **Note**
>
> Anda membuat tindakan dengan `custom` di `owner` lapangan untuk dijalankan dengan pekerja kerja. Anda tidak membuatnya dengan model integrasi. Untuk informasi tentang tindakan kustom, lihatBuat dan tambahkan tindakan kustom di CodePipeline.

Komponen tipe aksi

Komponen berikut membentuk tipe tindakan.

- ID tipe tindakan - ID terdiri dari kategori, pemilik, penyedia, dan versi. Contoh berikut menunjukkan ID tipe tindakan dengan pemilik`ThirdParty`, kategori`Test`, penyedia bernama`TestProvider`, dan versi`1`.

```
{
    "Category": "Test",
    "Owner": "ThirdParty",
    "Provider": "TestProvider",
    "Version": "1"
},
```

- Konfigurasi pelaksana — Model integrasi, atau mesin tindakan, yang ditentukan saat tindakan dibuat. Saat Anda menentukan eksekutor untuk tipe tindakan, Anda memilih salah satu dari dua jenis:
  - Lambda: Pemilik tipe tindakan menulis integrasi sebagai fungsi Lambda, yang dipanggil oleh CodePipeline setiap kali ada pekerjaan yang tersedia untuk tindakan tersebut.
  - JobWorker: Pemilik tipe tindakan menulis integrasi sebagai pekerja kerja yang melakukan polling untuk pekerjaan yang tersedia di jaringan pipa pelanggan. Pekerja pekerjaan kemudian menjalankan pekerjaan dan menyerahkan hasil pekerjaan kembali CodePipeline dengan menggunakan CodePipeline APIs.

> **Note**
>
> Model integrasi pekerja kerja bukanlah model integrasi yang disukai.

- Artefak input dan output: Batas artefak yang ditentukan oleh pemilik tipe tindakan untuk pelanggan tindakan.
- Izin: Strategi izin yang menunjuk pelanggan yang dapat mengakses jenis tindakan pihak ketiga. Strategi izin yang tersedia bergantung pada model integrasi yang dipilih untuk jenis tindakan.
- URLs: Tautan mendalam ke sumber daya yang dapat berinteraksi dengan pelanggan, seperti halaman konfigurasi pemilik tipe tindakan.

Topik

- Meminta tipe tindakan
- Tambahkan tipe tindakan yang tersedia ke pipeline (konsol)
- Melihat tipe tindakan
- Memperbarui jenis tindakan

Meminta tipe tindakan

Jika jenis CodePipeline tindakan baru diminta oleh penyedia pihak ketiga, tipe tindakan akan dibuat untuk pemilik tipe tindakan CodePipeline, dan pemilik dapat mengelola dan melihat jenis tindakan.

Jenis tindakan dapat berupa tindakan pribadi atau publik. Ketika jenis tindakan Anda dibuat, itu bersifat pribadi. Untuk meminta jenis tindakan diubah menjadi tindakan publik, hubungi tim CodePipeline layanan.

Sebelum membuat file definisi tindakan, sumber daya pelaksana, dan permintaan tipe tindakan untuk CodePipeline tim, Anda harus memilih model integrasi.

Langkah 1: Pilih model integrasi Anda

Pilih model integrasi Anda dan kemudian buat konfigurasi untuk model itu. Setelah Anda memilih model integrasi, Anda harus mengonfigurasi sumber daya integrasi Anda.

- Untuk model integrasi Lambda, Anda membuat fungsi Lambda dan menambahkan izin. Tambahkan izin ke fungsi Lambda integrator Anda untuk menyediakan CodePipeline

layanan dengan izin untuk memanggilnya menggunakan prinsip layanan:. CodePipeline `codepipeline.amazonaws.com` Izin dapat ditambahkan menggunakan CloudFormation atau baris perintah.

- Contoh untuk menambahkan izin menggunakan 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
```

- Dokumentasi untuk baris perintah

- Untuk model integrasi pekerja kerja, Anda membuat integrasi dengan daftar akun yang diizinkan tempat pekerja pekerjaan melakukan polling untuk pekerjaan dengan. CodePipeline APIs

Langkah 2: Buat file definisi tipe tindakan

Anda menentukan tipe tindakan dalam file definisi tipe tindakan menggunakan JSON. Dalam file, Anda menyertakan kategori tindakan, model integrasi yang digunakan untuk mengelola tipe tindakan, dan properti konfigurasi.

> **Note**
>
> Setelah membuat tindakan publik, Anda tidak dapat mengubah properti tipe tindakan di bawah `properties` dari `optional` kerequired. Anda juga tidak dapat mengubah`owner`.

Untuk informasi selengkapnya tentang parameter file definisi tipe tindakan, lihat ActionTypeDeclarationdan UpdateActionTypedi Referensi CodePipeline API.

Ada delapan bagian dalam file definisi tipe tindakan:

- `description`: Deskripsi untuk jenis tindakan yang akan diperbarui.
- `executor`: Informasi tentang pelaksana untuk tipe tindakan yang dibuat dengan model integrasi yang didukung, baik `Lambda` atau`job worker`. Anda hanya dapat memberikan salah satu `jobWorkerExecutorConfiguration` atau`lambdaExecutorConfiguration`, berdasarkan jenis eksekutor Anda.

- `configuration`: Sumber daya untuk konfigurasi tipe tindakan, berdasarkan model integrasi yang dipilih. Untuk model integrasi Lambda, gunakan fungsi Lambda ARN. Untuk model integrasi pekerja kerja, gunakan akun atau daftar akun dari tempat pekerja kerja berjalan.
- `jobTimeout`: Batas waktu dalam hitungan detik untuk pekerjaan itu. Eksekusi tindakan dapat terdiri dari beberapa pekerjaan. Ini adalah batas waktu untuk satu pekerjaan, dan bukan untuk seluruh eksekusi tindakan.

  > **Note**
  >
  > Untuk model integrasi Lambda, batas waktu maksimum adalah 15 menit.

- `policyStatementsTemplate`: Pernyataan kebijakan yang menentukan izin di akun CodePipeline pelanggan yang diperlukan untuk berhasil menjalankan eksekusi tindakan.
- `type`: Model integrasi yang digunakan untuk membuat dan memperbarui tipe tindakan, baik `Lambda` atau`JobWorker`.

- `id`: Kategori, pemilik, penyedia, dan ID versi untuk tipe tindakan:
  - `category`: Jenis tindakan dapat diambil dalam tahap: Sumber, Bangun, Deploy, Uji, Invoke, atau Persetujuan.
  - `provider`: Penyedia tipe tindakan yang dipanggil, seperti perusahaan penyedia atau nama produk. Nama penyedia diberikan saat tipe tindakan dibuat.
  - `owner`: Pencipta tipe tindakan yang disebut: `AWS` atau`ThirdParty`.
  - `version`: String yang digunakan untuk versi tipe tindakan. Untuk versi pertama, atur nomor versi ke 1.
- `inputArtifactDetails`: Jumlah artefak yang diharapkan dari tahap sebelumnya dalam pipa.
- `outputArtifactDetails`: Jumlah artefak yang diharapkan dari hasil dari tahap tipe aksi.
- `permissions`: Detail mengidentifikasi akun dengan izin untuk menggunakan tipe tindakan.
- `properties`: Parameter yang diperlukan untuk menyelesaikan tugas proyek Anda.
  - `description`: Deskripsi properti yang ditampilkan kepada pengguna.
  - `optional`: Apakah properti konfigurasi adalah opsional.
  - `noEcho`: Apakah nilai bidang yang dimasukkan oleh pelanggan dihilangkan dari log. Jika`true`, maka nilainya disunting saat dikembalikan dengan permintaan GetPipeline API.
  - `key`: Apakah properti konfigurasi adalah kunci.

- `queryable`: Apakah properti tersebut digunakan dengan polling. Tipe tindakan dapat memiliki hingga satu properti yang dapat dikueri. Jika ada, properti tersebut harus diperlukan dan bukan rahasia.
- `name`: Nama properti yang ditampilkan kepada pengguna.

- `urls`: Daftar URLs CodePipeline tampilan untuk pengguna Anda.
  - `entityUrlTemplate`: URL ke sumber daya eksternal untuk jenis tindakan, seperti halaman konfigurasi.
  - `executionUrlTemplate`: URL ke detail untuk menjalankan tindakan terbaru.
  - `revisionUrlTemplate`: URL ditampilkan di CodePipeline konsol ke halaman tempat pelanggan dapat memperbarui atau mengubah konfigurasi tindakan eksternal.
  - `thirdPartyConfigurationUrl`URL halaman tempat pengguna dapat mendaftar untuk layanan eksternal dan melakukan konfigurasi awal tindakan yang disediakan oleh layanan tersebut.

Kode berikut menunjukkan contoh file definisi tipe tindakan.

```
{
   "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"
      }
   }
}
```

Langkah 3: Daftarkan Integrasi Anda dengan CodePipeline

Untuk mendaftarkan jenis tindakan Anda CodePipeline, Anda menghubungi tim CodePipeline layanan dengan permintaan Anda.

Tim CodePipeline layanan mendaftarkan integrasi tipe tindakan baru dengan membuat perubahan dalam basis kode layanan. CodePipeline mencatat dua tindakan baru: tindakan publik dan tindakan pribadi. Anda menggunakan tindakan pribadi untuk pengujian, dan kemudian ketika siap, Anda mengaktifkan tindakan publik untuk melayani lalu lintas pelanggan.

Untuk mendaftarkan permintaan integrasi Lambda

- Kirim permintaan ke tim CodePipeline layanan menggunakan formulir berikut.

```
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 Wilayah AWS where your action will be available

5. Will this be available as a public action?
```

Untuk mendaftarkan permintaan integrasi pekerja kerja

- Kirim permintaan ke tim CodePipeline layanan menggunakan formulir berikut.

```
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 Wilayah AWS where your action will be available

5. Will this be available as a public action?
```

Langkah 4: Aktifkan Integrasi Baru Anda

Hubungi tim CodePipeline layanan ketika Anda siap untuk menggunakan integrasi baru secara publik.

Tambahkan tipe tindakan yang tersedia ke pipeline (konsol)

Anda menambahkan tipe tindakan Anda ke pipeline sehingga Anda dapat mengujinya. Anda dapat melakukan ini dengan membuat pipeline baru atau mengedit yang sudah ada.

> **Note**
>
> Jika jenis tindakan Anda adalah tindakan kategori sumber, build, atau deploy, Anda dapat menambahkannya dengan membuat pipeline. Jika tipe tindakan Anda ada dalam kategori pengujian, Anda harus menambahkannya dengan mengedit pipeline yang ada.

Untuk menambahkan tipe tindakan ke pipeline yang ada dari CodePipeline konsol

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Dalam daftar pipeline, pilih pipeline tempat Anda ingin menambahkan tipe tindakan.
3. Pada halaman tampilan ringkasan pipeline, pilih Edit.

4. Pilih untuk mengedit panggung. Pada tahap di mana Anda ingin menambahkan tipe tindakan Anda, pilih Tambahkan grup tindakan. Halaman Edit tindakan ditampilkan.
5. Pada halaman Edit tindakan, dalam nama Tindakan, masukkan nama untuk tindakan tersebut. Ini adalah nama yang ditampilkan untuk panggung di pipeline Anda.
6. Di Penyedia tindakan, pilih jenis tindakan Anda dari daftar.

   Perhatikan bahwa nilai dalam daftar didasarkan pada yang `provider` ditentukan dalam file definisi tipe tindakan.
7. Di artefak Input, masukkan nama artefak dalam format ini:

   `Artifactname``::`*`FileName`*

   Perhatikan bahwa jumlah minimum dan maksimum yang diizinkan ditentukan berdasarkan yang `inputArtifactDetails` ditentukan dalam file definisi tipe tindakan.
8. Pilih Connect to<Action_Name>.

   Jendela browser terbuka dan terhubung ke situs web yang telah Anda buat untuk jenis tindakan Anda.
9. Masuk ke situs web Anda sebagai pelanggan dan selesaikan langkah-langkah yang diambil pelanggan untuk menggunakan jenis tindakan Anda. Langkah-langkah Anda akan bervariasi tergantung pada kategori tindakan, situs web, dan konfigurasi Anda, tetapi biasanya mencakup tindakan penyelesaian yang mengembalikan pelanggan ke halaman tindakan Edit.
10. Di halaman CodePipeline Edit tindakan, bidang konfigurasi tambahan untuk tampilan tindakan. Bidang yang ditampilkan adalah properti konfigurasi yang Anda tentukan dalam file definisi tindakan. Masukkan informasi di bidang yang disesuaikan untuk jenis tindakan Anda.

    Misalnya, jika file definisi tindakan menetapkan properti bernama`Host`, maka bidang dengan label Host ditampilkan di halaman Edit tindakan untuk tindakan Anda.
11. Dalam artefak Output, masukkan nama artefak dalam format ini:

    `Artifactname``::`*`FileName`*

    Perhatikan bahwa jumlah minimum dan maksimum yang diizinkan ditentukan berdasarkan yang `outputArtifactDetails` ditentukan dalam file definisi tipe tindakan.
12. Pilih Selesai untuk kembali ke halaman detail pipeline.

> **Note**
>
> Pelanggan Anda secara opsional dapat menggunakan CLI untuk menambahkan tipe tindakan ke pipeline mereka.

13. Untuk menguji tindakan Anda, lakukan perubahan ke sumber yang ditentukan dalam tahap sumber pipeline atau ikuti langkah-langkah dalam Memulai Pipeline secara Manual.

Untuk membuat pipeline dengan tipe tindakan Anda, ikuti langkah-langkahnya Buat pipeline, tahapan, dan tindakan dan pilih jenis tindakan Anda dari tahapan sebanyak yang akan Anda uji.

Melihat tipe tindakan

Anda dapat menggunakan CLI untuk melihat tipe tindakan Anda. Gunakan get-action-type perintah untuk melihat jenis tindakan yang telah dibuat menggunakan model integrasi.

Untuk melihat tipe tindakan

1. Buat file JSON input dan beri nama file`file.json`. Tambahkan ID tipe tindakan Anda dalam format JSON seperti yang ditunjukkan pada contoh berikut.

```
{
    "category": "Test",
    "owner": "ThirdParty",
    "provider": "TestProvider",
    "version": "1"
}
```

2. Di jendela terminal atau di baris perintah, jalankan get-action-type perintah.

```
aws codepipeline get-action-type --cli-input-json file://file.json
```

   Perintah ini mengembalikan output definisi tindakan untuk tipe tindakan. Contoh ini menunjukkan tipe tindakan yang dibuat dengan model integrasi 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": []
    }
}
```

Memperbarui jenis tindakan

Anda dapat menggunakan CLI untuk mengedit jenis tindakan yang dibuat dengan model integrasi.

Untuk jenis tindakan publik, Anda tidak dapat memperbarui pemilik, Anda tidak dapat mengubah properti opsional ke required, dan Anda hanya dapat menambahkan properti opsional baru.

1. Gunakan `get-action-type` perintah untuk mendapatkan struktur untuk tipe tindakan Anda. Salin strukturnya.

2. Buat file JSON input dan beri nama. `action.json` Tempelkan struktur tipe tindakan yang Anda salin pada langkah sebelumnya ke dalamnya. Perbarui parameter apa pun yang ingin Anda ubah. Anda juga dapat menambahkan parameter opsional.

   Untuk informasi selengkapnya tentang parameter untuk file input, lihat deskripsi file definisi tindakan diLangkah 2: Buat file definisi tipe tindakan.

   Contoh berikut menunjukkan cara memperbarui contoh jenis tindakan yang dibuat dengan model integrasi Lambda. Contoh ini membuat perubahan berikut:

   - Mengubah `provider` nama menjadi`TestProvider1`.
   - Tambahkan batas waktu kerja 900 detik.
   - Menambahkan properti konfigurasi tindakan bernama `Host` yang ditampilkan ke pelanggan menggunakan tindakan.

```
{
    "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. Di terminal atau baris perintah, jalankan update-action-type perintah

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

Perintah ini mengembalikan output tipe tindakan agar sesuai dengan parameter yang diperbarui.

Buat dan tambahkan tindakan kustom di CodePipeline

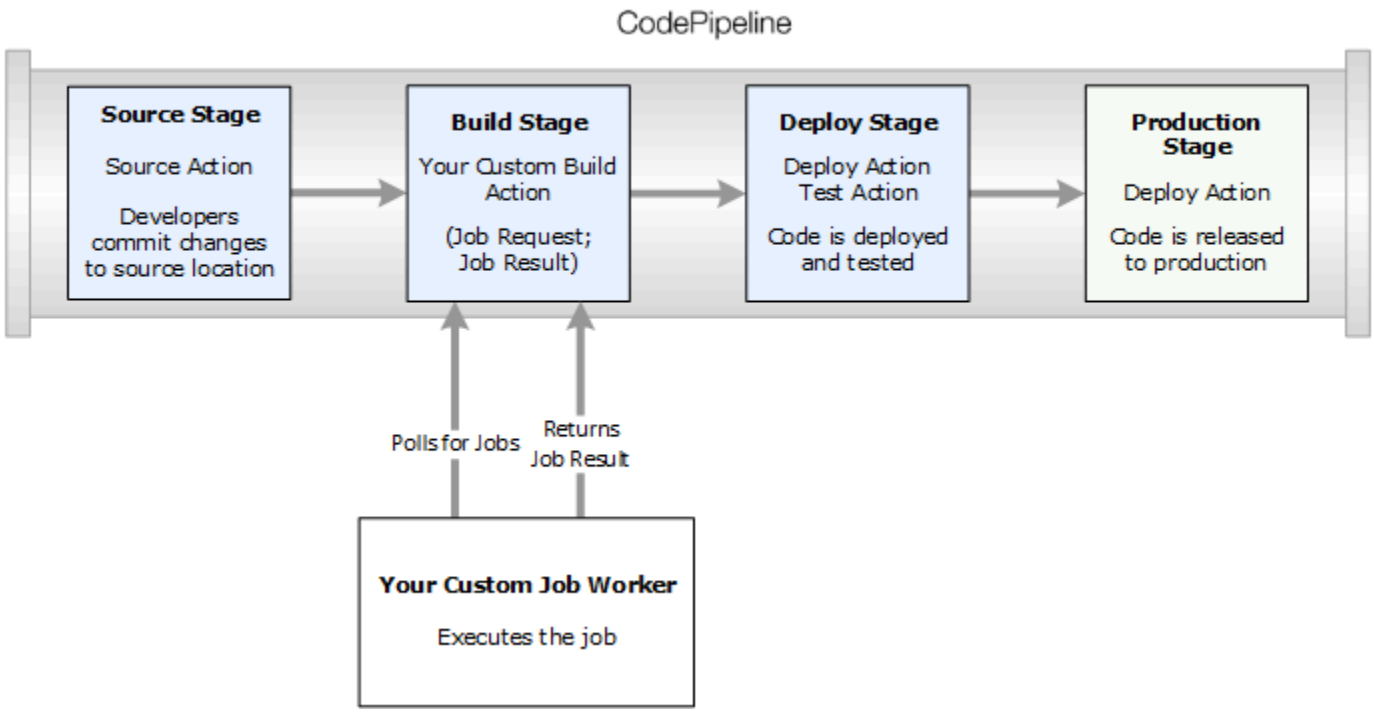
AWS CodePipeline menyertakan sejumlah tindakan yang membantu Anda mengonfigurasi pembuatan, pengujian, dan penerapan sumber daya untuk proses rilis otomatis Anda. Jika proses rilis Anda menyertakan aktivitas yang tidak disertakan dalam tindakan default, seperti proses build yang dikembangkan secara internal atau rangkaian pengujian, Anda dapat membuat tindakan kustom untuk tujuan tersebut dan memasukkannya ke dalam pipeline Anda. Anda dapat menggunakan AWS CLI untuk membuat tindakan kustom di pipeline yang terkait dengan AWS akun Anda.

Anda dapat membuat tindakan kustom untuk kategori AWS CodePipeline tindakan berikut:

- Tindakan pembuatan kustom yang membangun atau mengubah item
- Tindakan penerapan kustom yang menyebarkan item ke satu atau beberapa server, situs web, atau repositori
- Tindakan pengujian kustom yang mengonfigurasi dan menjalankan pengujian otomatis
- Tindakan pemanggilan khusus yang menjalankan fungsi

Saat Anda membuat tindakan kustom, Anda juga harus membuat pekerja pekerjaan yang akan melakukan polling CodePipeline untuk permintaan pekerjaan untuk tindakan kustom ini, menjalankan pekerjaan, dan mengembalikan hasil status ke CodePipeline. Pekerja pekerjaan ini dapat ditemukan di komputer atau sumber daya apa pun selama memiliki akses ke titik akhir publik untuk CodePipeline. Untuk mengelola akses dan keamanan dengan mudah, pertimbangkan untuk menghosting pekerja kerja Anda di instans Amazon EC2.

Diagram berikut menunjukkan tampilan tingkat tinggi dari pipeline yang menyertakan tindakan pembuatan kustom:



Ketika pipeline menyertakan tindakan kustom sebagai bagian dari tahapan, pipeline akan membuat permintaan pekerjaan. Pekerja pekerjaan khusus mendeteksi permintaan tersebut dan melakukan pekerjaan itu (dalam contoh ini, proses kustom menggunakan perangkat lunak pembuatan pihak ketiga). Ketika tindakan selesai, pekerja kerja mengembalikan hasil sukses atau hasil kegagalan. Jika hasil sukses diterima, pipa akan memberikan revisi dan artefaknya untuk tindakan selanjutnya. Jika kegagalan dikembalikan, pipa tidak akan memberikan revisi untuk tindakan selanjutnya dalam pipa.

> **Note**
>
> Instruksi ini mengasumsikan bahwa Anda telah menyelesaikan langkah-langkahnya Memulai dengan CodePipeline.

Topik

- Buat tindakan kustom
- Buat pekerja pekerjaan untuk tindakan kustom Anda
- Menambahkan tindakan kustom ke pipeline

Buat tindakan kustom

Untuk membuat tindakan kustom dengan AWS CLI

1. Buka editor teks dan buat file JSON untuk tindakan kustom Anda yang mencakup kategori tindakan, penyedia tindakan, dan pengaturan apa pun yang diperlukan oleh tindakan kustom Anda. Misalnya, untuk membuat tindakan pembuatan kustom yang hanya memerlukan satu properti, file JSON Anda mungkin terlihat seperti ini:

```
{
    "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
```

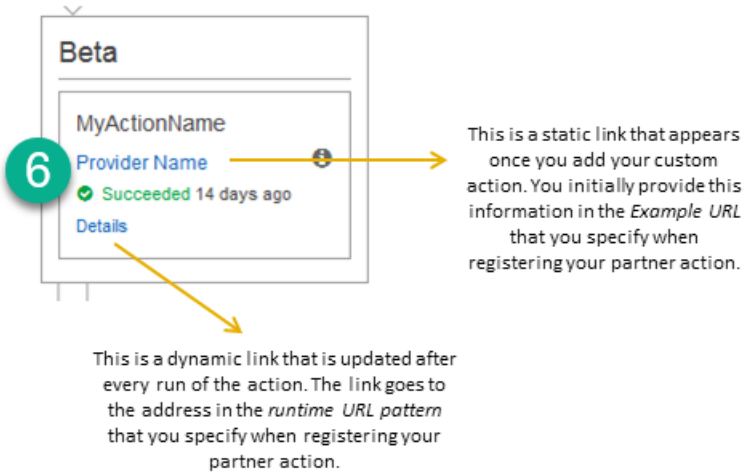
```
    },
    "tags": [{
      "key": "Project",
      "value": "ProjectA"
    }]
}
```

Contoh ini menambahkan penandaan ke tindakan kustom dengan menyertakan kunci `Project` tag dan `ProjectA` nilai pada tindakan kustom. Untuk informasi selengkapnya tentang menandai sumber daya CodePipeline, lihatPenandaan pada sumber daya.

Ada dua properti yang termasuk dalam file JSON, `entityUrlTemplate` dan`executionUrlTemplate`. Anda dapat merujuk ke nama di properti konfigurasi tindakan kustom dalam templat URL dengan mengikuti format`{Config:`*`name`*`}`, selama properti konfigurasi diperlukan dan bukan rahasia. Misalnya, dalam sampel di atas, `entityUrlTemplate` nilai mengacu pada properti konfigurasi*`ProjectName`*.

- `entityUrlTemplate`: tautan statis yang memberikan informasi tentang penyedia layanan untuk tindakan tersebut. Dalam contoh, sistem build menyertakan tautan statis ke setiap proyek build. Format tautan akan bervariasi, tergantung pada penyedia build Anda (atau, jika Anda membuat jenis tindakan yang berbeda, seperti pengujian, penyedia layanan lain). Anda harus menyediakan format tautan ini sehingga ketika tindakan kustom ditambahkan, pengguna dapat memilih tautan ini untuk membuka browser ke halaman di situs web Anda yang menyediakan spesifikasi untuk proyek pembangunan (atau lingkungan pengujian).
- `executionUrlTemplate`: tautan dinamis yang akan diperbarui dengan informasi tentang tindakan saat ini atau terbaru. Ketika pekerja pekerjaan kustom Anda memperbarui status pekerjaan (misalnya, keberhasilan, kegagalan, atau dalam proses), itu juga akan memberikan `externalExecutionId` yang akan digunakan untuk melengkapi tautan. Tautan ini dapat digunakan untuk memberikan rincian tentang menjalankan suatu tindakan.

Misalnya, saat Anda melihat tindakan di pipeline, Anda melihat dua tautan berikut:



1 Tautan statis ini muncul setelah Anda menambahkan tindakan kustom dan menunjuk ke alamat`entityUrlTemplate`, yang Anda tentukan saat membuat tindakan kustom.

2 Tautan dinamis ini diperbarui setelah setiap proses tindakan dan menunjuk ke alamat`executionUrlTemplate`, yang Anda tentukan saat membuat tindakan kustom.

Untuk informasi selengkapnya tentang jenis tautan ini, serta `RevisionURLTemplate` dan`ThirdPartyURL`, lihat ActionTypeSettingsdan CreateCustomActionTypedi Referensi CodePipeline API. Untuk informasi selengkapnya tentang persyaratan struktur tindakan dan cara membuat tindakan, lihatCodePipeline referensi struktur pipa.

2. Simpan file JSON dan berikan nama yang dapat Anda ingat dengan mudah (misalnya, *`MyCustomAction`* .json).
3. Buka sesi terminal (Linux, OS X, Unix) atau command prompt (Windows) di komputer tempat Anda menginstal file. AWS CLI
4. Gunakan AWS CLI untuk menjalankan aws codepipeline create-custom-action-type perintah, menentukan nama file JSON yang baru saja Anda buat.

   Misalnya, untuk membuat tindakan kustom build:

> **Important**
>
> Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

```
aws codepipeline create-custom-action-type --cli-input-json
 file://MyCustomAction.json
```

5. Perintah ini mengembalikan seluruh struktur tindakan kustom yang Anda buat, serta properti konfigurasi `JobList` tindakan, yang ditambahkan untuk Anda. Saat menambahkan tindakan kustom ke pipeline, Anda dapat menggunakannya `JobList` untuk menentukan proyek mana dari penyedia yang dapat Anda polling untuk pekerjaan. Jika Anda tidak mengonfigurasi ini, semua pekerjaan yang tersedia akan dikembalikan saat pekerja pekerjaan khusus Anda melakukan polling untuk pekerjaan.

   Misalnya, perintah sebelumnya mungkin mengembalikan struktur yang mirip dengan berikut ini:

```
{
    "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**
>
> Sebagai bagian dari output create-custom-action-type perintah, `id` bagian ini termasuk`"owner": "Custom"`. CodePipeline secara otomatis menetapkan `Custom` sebagai pemilik jenis tindakan kustom. Nilai ini tidak dapat ditetapkan atau diubah saat Anda menggunakan create-custom-action-type perintah atau update-pipeline perintah.

Buat pekerja pekerjaan untuk tindakan kustom Anda

Tindakan kustom memerlukan pekerja pekerjaan yang akan melakukan polling CodePipeline untuk permintaan pekerjaan untuk tindakan kustom, menjalankan pekerjaan, dan mengembalikan hasil status ke CodePipeline. Pekerja pekerjaan dapat ditemukan di komputer atau sumber daya apa pun selama memiliki akses ke titik akhir publik untuk CodePipeline.

Ada banyak cara untuk merancang pekerja kerja Anda. Bagian berikut memberikan beberapa panduan praktis untuk mengembangkan pekerja pekerjaan khusus Anda untuk CodePipeline.

Topik

- Pilih dan konfigurasikan strategi manajemen izin untuk pekerja kerja Anda
- Kembangkan pekerja kerja untuk tindakan kustom Anda
- Arsitektur dan contoh pekerja kerja khusus

Pilih dan konfigurasikan strategi manajemen izin untuk pekerja kerja Anda

Untuk mengembangkan pekerja pekerjaan khusus untuk tindakan kustom Anda di CodePipeline, Anda akan memerlukan strategi untuk integrasi pengguna dan manajemen izin.

Strategi paling sederhana adalah menambahkan infrastruktur yang Anda butuhkan untuk pekerja pekerjaan khusus Anda dengan membuat instans Amazon EC2 dengan peran instans IAM, yang memungkinkan Anda meningkatkan sumber daya yang Anda butuhkan untuk integrasi dengan mudah. Anda dapat menggunakan integrasi bawaan AWS untuk menyederhanakan interaksi antara pekerja pekerjaan khusus Anda dan CodePipeline.

Untuk mengatur instans Amazon EC2

1. Pelajari lebih lanjut tentang Amazon EC2 dan tentukan apakah itu pilihan yang tepat untuk integrasi Anda. Untuk selengkapnya, lihat Amazon EC2 - Hosting Server Virtual.
2. Mulai membuat instans Amazon EC2 Anda. Untuk selengkapnya, lihat Memulai Instans Linux Amazon EC2.

Strategi lain yang perlu dipertimbangkan adalah menggunakan federasi identitas dengan IAM untuk mengintegrasikan sistem dan sumber daya penyedia identitas yang ada. Strategi ini sangat berguna jika Anda sudah memiliki penyedia identitas perusahaan atau sudah dikonfigurasi untuk mendukung pengguna menggunakan penyedia identitas web. Federasi identitas memungkinkan Anda memberikan akses aman ke AWS sumber daya, termasuk CodePipeline, tanpa harus membuat atau mengelola pengguna IAM. Anda dapat menggunakan fitur dan kebijakan untuk persyaratan keamanan kata sandi dan rotasi kredensyal. Anda dapat menggunakan contoh aplikasi sebagai template untuk desain Anda sendiri.

Untuk mengatur federasi identitas

1. Pelajari lebih lanjut tentang federasi identitas IAM. Untuk selengkapnya, lihat Mengelola Federasi.
2. Tinjau contoh di Skenario untuk Memberikan Akses Sementara untuk mengidentifikasi skenario akses sementara yang paling sesuai dengan kebutuhan tindakan kustom Anda.
3. Tinjau contoh kode federasi identitas yang relevan dengan infrastruktur Anda, seperti:
   - Contoh Aplikasi Federasi Identitas untuk Kasus Penggunaan Direktori Aktif
4. Mulai mengonfigurasi federasi identitas. Untuk selengkapnya, lihat Penyedia Identitas dan Federasi di Panduan Pengguna IAM.

Hentikan salah satu dari berikut ini untuk digunakan di bawah Anda Akun AWS saat menjalankan tindakan kustom dan pekerja pekerjaan Anda.

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. Konsol Manajemen AWS Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
IAM	(Disarankan) Gunakan kredensyal konsol sebagai kredensyal sementara untuk menandatangani permintaan terprogram ke,, atau. AWS CLI AWS SDKs AWS APIs	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk itu AWS CLI, lihat Login untuk pengembangan AWS lokal di Panduan AWS Command Line Interface Pengguna. • Untuk AWS SDKs, lihat Login untuk pengembangan AWS lokal di Panduan Referensi Alat AWS SDKs dan Alat.
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensyal sementara untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengkonfigurasi yang akan AWS CLI digunakan AWS IAM Identity Center dalam Panduan AWS Command Line Interface Pengguna.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
		• Untuk AWS SDKs, alat, dan AWS APIs, lihat Autentikasi Pusat Identitas IAM di Panduan Referensi Alat AWS SDKs dan Alat.
IAM	Gunakan kredensyal sementara untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk dalam Menggunakan kredensyal sementara dengan AWS sumber daya di Panduan Pengguna IAM.
IAM	(Tidak direkomendasikan) Gunakan kredensyal jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI,, AWS SDKs atau. AWS APIs	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk mengetahui AWS CLI, lihat Mengautentikasi menggunakan kredensyal pengguna IAM di Panduan Pengguna.AWS Command Line Interface • Untuk AWS SDKs dan alat, lihat Mengautentikasi menggunakan kredensyal jangka panjang di Panduan Referensi Alat AWS SDKs dan Alat. • Untuk AWS APIs, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Berikut ini adalah contoh kebijakan yang mungkin Anda buat untuk digunakan dengan pekerja pekerjaan khusus Anda. Kebijakan ini dimaksudkan sebagai contoh saja dan disediakan apa adanya.

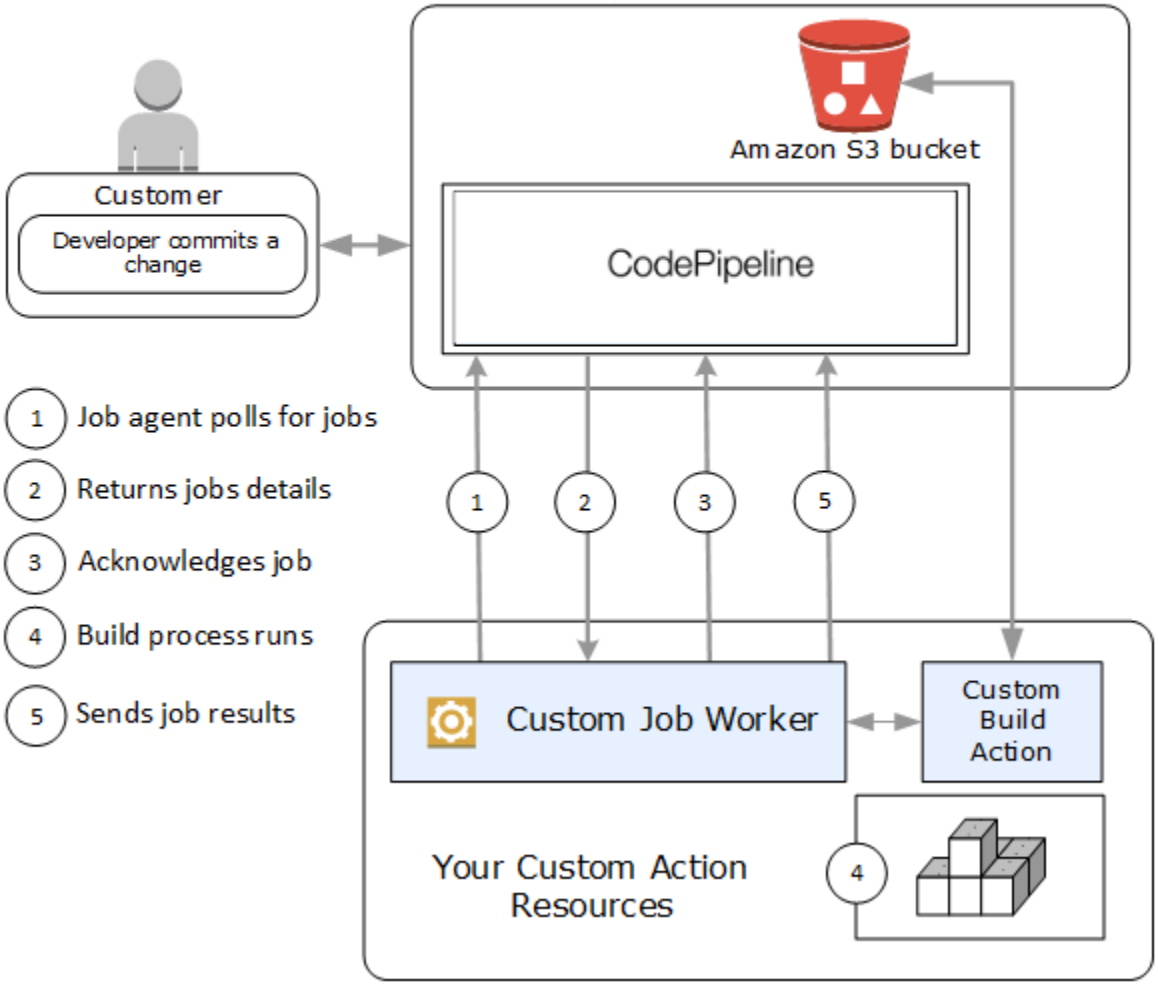
JSON

```
{
  "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**
>
> Pertimbangkan untuk menggunakan kebijakan yang `AWSCodePipelineCustomActionAccess` dikelola.

Kembangkan pekerja kerja untuk tindakan kustom Anda

Setelah Anda memilih strategi manajemen izin, Anda harus mempertimbangkan bagaimana pekerja kerja Anda akan berinteraksi CodePipeline. Diagram tingkat tinggi berikut menunjukkan alur kerja tindakan kustom dan pekerja pekerjaan untuk proses pembuatan.



1. Pekerja kerja Anda melakukan polling CodePipeline untuk pekerjaan yang digunakan`PollForJobs`.
2. Ketika pipeline dipicu oleh perubahan dalam tahap sumbernya (misalnya, ketika pengembang melakukan perubahan), proses rilis otomatis dimulai. Proses berlanjut hingga tahap di mana tindakan kustom Anda telah dikonfigurasi. Ketika mencapai tindakan Anda dalam tahap ini, CodePipeline antrian pekerjaan. Pekerjaan ini akan muncul jika pekerja kerja Anda menelepon `PollForJobs` lagi untuk mendapatkan status. Ambil detail pekerjaan dari `PollForJobs` dan berikan kembali ke pekerja kerja Anda.
3. Pekerja kerja menelepon `AcknowledgeJob` untuk CodePipeline mengirim pengakuan pekerjaan. CodePipeline mengembalikan pengakuan yang menunjukkan pekerja kerja harus melanjutkan job (`InProgress`), atau, jika Anda memiliki lebih dari satu polling pekerja kerja untuk pekerjaan dan pekerja pekerjaan lain telah mengklaim pekerjaan tersebut, respons `InvalidNonceException`

kesalahan akan dikembalikan. Setelah `InProgress` pengakuan, CodePipeline menunggu hasil dikembalikan.

4. Pekerja pekerjaan memulai tindakan kustom Anda pada revisi, dan kemudian tindakan Anda berjalan. Seiring dengan tindakan lainnya, tindakan kustom Anda mengembalikan hasil ke pekerja pekerjaan. Dalam contoh tindakan kustom build, aksi menarik artefak dari bucket Amazon S3, membangunnya, dan mendorong artefak yang berhasil dibangun kembali ke bucket Amazon S3.
5. Saat tindakan sedang berjalan, pekerja pekerjaan dapat memanggil `PutJobSuccessResult` dengan token kelanjutan (serialisasi status pekerjaan yang dihasilkan oleh pekerja pekerjaan, misalnya pengenal build dalam format JSON, atau kunci objek Amazon S3), serta `ExternalExecutionId` informasi yang akan digunakan untuk mengisi tautan. `executionUrlTemplate` Ini akan memperbarui tampilan konsol pipeline dengan tautan yang berfungsi ke detail tindakan tertentu saat sedang berlangsung. Meskipun tidak diperlukan, ini adalah praktik terbaik karena memungkinkan pengguna untuk melihat status tindakan kustom Anda saat berjalan.

   Setelah `PutJobSuccessResult` dipanggil, pekerjaan dianggap selesai. Pekerjaan baru dibuat di dalamnya CodePipeline termasuk token kelanjutan. Pekerjaan ini akan muncul jika pekerja pekerjaan Anda menelepon `PollForJobs` lagi. Pekerjaan baru ini dapat digunakan untuk memeriksa status tindakan, dan mengembalikan dengan token kelanjutan, atau kembali tanpa token kelanjutan setelah tindakan selesai.

   > **Note**
   >
   > Jika pekerja kerja Anda melakukan semua pekerjaan untuk tindakan khusus, Anda harus mempertimbangkan untuk memecah pemrosesan pekerja kerja Anda menjadi setidaknya dua langkah. Langkah pertama menetapkan halaman detail untuk tindakan Anda. Setelah Anda membuat halaman detail, Anda dapat membuat serial status pekerja pekerjaan dan mengembalikannya sebagai token kelanjutan, tunduk pada batas ukuran (lihatKuota di AWS CodePipeline). Misalnya, Anda dapat menulis status tindakan ke dalam string yang Anda gunakan sebagai token kelanjutan. Langkah kedua (dan langkah-langkah selanjutnya) dari pemrosesan pekerja kerja Anda melakukan pekerjaan aktual dari tindakan tersebut. Langkah terakhir mengembalikan keberhasilan atau kegagalan CodePipeline, tanpa token kelanjutan pada langkah terakhir.

   Untuk informasi selengkapnya tentang penggunaan token lanjutan, lihat spesifikasinya `PutJobSuccessResult` di Referensi CodePipeline API.

6. Setelah tindakan kustom selesai, pekerja pekerjaan mengembalikan hasil tindakan kustom CodePipeline dengan memanggil salah satu dari dua APIs:
   - `PutJobSuccessResult`tanpa token kelanjutan, yang menunjukkan tindakan kustom berhasil dijalankan
   - `PutJobFailureResult`, yang menunjukkan tindakan kustom tidak berhasil berjalan

   Bergantung pada hasilnya, pipeline akan melanjutkan ke tindakan berikutnya (sukses) atau berhenti (gagal).

Arsitektur dan contoh pekerja kerja khusus

Setelah Anda memetakan alur kerja tingkat tinggi Anda, Anda dapat membuat pekerja kerja Anda. Meskipun spesifik tindakan kustom Anda pada akhirnya akan menentukan apa yang diperlukan untuk pekerja pekerjaan Anda, sebagian besar pekerja kerja untuk tindakan kustom menyertakan fungsionalitas berikut:

- Polling untuk pekerjaan dari CodePipeline penggunaan`PollForJobs`.
- Mengakui pekerjaan dan mengembalikan hasil untuk CodePipeline menggunakan`AcknowledgeJob`,`PutJobSuccessResult`, dan. `PutJobFailureResult`
- Mengambil artefak dari and/or memasukkan artefak ke dalam ember Amazon S3 untuk pipa. Untuk mengunduh artefak dari bucket Amazon S3, Anda harus membuat klien Amazon S3 yang menggunakan penandatanganan Signature Version 4 (Sig V4). Sig V4 diperlukan untuk. AWS KMS

  Untuk mengunggah artefak ke bucket Amazon S3, Anda juga harus mengonfigurasi permintaan Amazon `PutObject` S3 untuk menggunakan enkripsi. Saat ini hanya AWS Key Management Service (AWS KMS) yang didukung untuk enkripsi. AWS KMS menggunakan AWS KMS keys. Untuk mengetahui apakah akan menggunakan Kunci yang dikelola AWS atau kunci yang dikelola pelanggan untuk mengunggah artefak, pekerja pekerjaan khusus Anda harus melihat data pekerjaan dan memeriksa properti kunci enkripsi. Jika properti disetel, Anda harus menggunakan ID kunci terkelola pelanggan tersebut saat mengonfigurasi. AWS KMS Jika properti kunci adalah null, Anda menggunakan. Kunci yang dikelola AWS CodePipeline menggunakan Kunci yang dikelola AWS kecuali dikonfigurasi lain.

  Untuk contoh yang menunjukkan cara membuat AWS KMS parameter di Java atau .NET, lihat Menentukan AWS Key Management Service di Amazon S3 Menggunakan. AWS SDKs Untuk informasi selengkapnya tentang bucket Amazon S3 CodePipeline, lihat. CodePipeline konsep

Contoh yang lebih kompleks dari pekerja pekerjaan khusus tersedia di GitHub. Sampel ini adalah open source dan disediakan apa adanya.

- Contoh Job Worker untuk CodePipeline: Unduh sampel dari GitHub repositori.

Menambahkan tindakan kustom ke pipeline

Setelah memiliki pekerja kerja, Anda dapat menambahkan tindakan kustom ke pipeline dengan membuat yang baru dan memilihnya saat Anda menggunakan wizard Buat Pipeline, dengan mengedit pipeline yang ada dan menambahkan tindakan kustom, atau dengan menggunakan AWS CLI, the SDKs, atau APIs.

> **Note**
>
> Anda dapat membuat pipeline di wizard Create Pipeline yang menyertakan tindakan kustom jika itu adalah tindakan build atau deploy. Jika tindakan kustom Anda ada dalam kategori pengujian, Anda harus menambahkannya dengan mengedit pipeline yang ada.

Topik

- Tambahkan tindakan khusus ke pipeline yang ada (CLI)

Tambahkan tindakan khusus ke pipeline yang ada (CLI)

Anda dapat menggunakan AWS CLI untuk menambahkan tindakan kustom ke pipeline yang ada.

1. Buka sesi terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan jalankan get-pipeline perintah untuk menyalin struktur pipeline yang ingin Anda edit ke dalam file JSON. Misalnya, untuk pipeline bernama**`MyFirstPipeline`**, Anda akan mengetik perintah berikut:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Perintah ini tidak mengembalikan apa pun, tetapi file yang Anda buat akan muncul di direktori tempat Anda menjalankan perintah.

2. Buka file JSON di editor teks apa pun dan ubah struktur file untuk menambahkan tindakan kustom Anda ke tahap yang ada.

> **Note**
>
> Jika Anda ingin tindakan Anda berjalan secara paralel dengan tindakan lain di tahap itu, pastikan Anda menetapkan `runOrder` nilai yang sama dengan tindakan itu.

Misalnya, untuk memodifikasi struktur pipeline untuk menambahkan tahapan bernama Build dan menambahkan tindakan kustom build ke tahap tersebut, Anda dapat memodifikasi JSON untuk menambahkan tahap Build sebelum tahap penerapan sebagai berikut:

```
,
    {
      "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. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, tentukan file JSON pipeline, mirip dengan berikut ini:

> **Important**
>
> Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

Perintah ini mengembalikan seluruh struktur pipa yang diedit.

4. Buka CodePipeline konsol dan pilih nama pipeline yang baru saja Anda edit.

Pipeline menunjukkan perubahan Anda. Lain kali Anda membuat perubahan ke lokasi sumber, pipa akan menjalankan revisi itu melalui struktur pipa yang direvisi.

Menandai tindakan kustom di CodePipeline

Tag adalah pasangan nilai kunci yang terkait dengan AWS sumber daya. Anda dapat menggunakan konsol atau CLI untuk menerapkan tag ke tindakan kustom Anda di. CodePipeline Untuk informasi tentang penandaan CodePipeline sumber daya, kasus penggunaan, kunci tag dan batasan nilai, serta jenis sumber daya yang didukung, lihat. Penandaan pada sumber daya

Anda dapat menambahkan, menghapus, dan memperbarui nilai tag dalam tindakan kustom. Anda dapat menambahkan hingga 50 tag untuk setiap tindakan kustom.

Topik

- Tambahkan tag ke tindakan kustom
- Lihat tag untuk tindakan kustom
- Mengedit tag untuk tindakan kustom
- Hapus tag dari tindakan kustom

Tambahkan tag ke tindakan kustom

Ikuti langkah-langkah ini untuk menggunakan AWS CLI untuk menambahkan tag ke tindakan kustom. Untuk menambahkan tag ke tindakan kustom saat Anda membuatnya, lihatBuat dan tambahkan tindakan kustom di CodePipeline.

Dalam langkah-langkah ini, kami menganggap bahwa Anda telah menginstal versi terbaru dari AWS CLI atau diperbarui ke versi terkini. Untuk informasi lebih lanjut, lihat Menginstal AWS Command Line Interface.

Di terminal atau baris perintah, jalankan tag-resource perintah, tentukan Nama Sumber Daya Amazon (ARN) dari tindakan khusus tempat Anda ingin menambahkan tag dan kunci serta nilai tag yang ingin Anda tambahkan. Anda dapat menambahkan lebih dari satu tag ke tindakan kustom. Misalnya, untuk menandai tindakan kustom dengan dua tag, kunci tag bernama *TestActionType* dengan nilai tag dari*UnitTest*, dan kunci tag bernama *ApplicationName* dengan nilai tag*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
```

Jika berhasil, perintah ini tidak mengembalikan apa pun.

Lihat tag untuk tindakan kustom

Ikuti langkah-langkah ini untuk menggunakan AWS CLI untuk melihat AWS tag untuk tindakan kustom. Jika tidak ada tanda yang telah ditambahkan, daftar yang ditampilkan kosong.

Pada terminal atau baris perintah, jalankan perintah list-tags-for-resource. Misalnya, untuk melihat daftar kunci tag dan nilai tag untuk tindakan kustom dengan `arn:aws:codepipeline:`*`us-west-2`*`:`*`account-id`*`:actiontype:`*`Owner`*`/`*`Category`*`/`*`Provider`*`/`*`Version`* ARN:

```
aws codepipeline list-tags-for-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:actiontype:Owner/Category/Provider/Version
```

Jika berhasil, perintah ini menampilkan informasi yang serupa dengan yang berikut:

```
{
    "tags": {
        "TestActionType": "UnitTest",
        "ApplicationName": "MyApplication"
    }
}
```

Mengedit tag untuk tindakan kustom

Ikuti langkah-langkah ini untuk menggunakan AWS CLI untuk mengedit tag untuk tindakan kustom. Anda dapat mengubah nilai untuk kunci yang ada atau menambahkan kunci lain. Anda juga dapat menghapus tag dari tindakan kustom, seperti yang ditunjukkan di bagian berikutnya.

Di terminal atau baris perintah, jalankan tag-resource perintah, tentukan Nama Sumber Daya Amazon (ARN) dari tindakan kustom tempat Anda ingin memperbarui tag dan menentukan kunci tag dan nilai tag:

```
aws codepipeline tag-resource --resource-arn arn:aws:codepipeline:us-
west-2:account-id:actiontype:Owner/Category/Provider/Version --tags
 key=TestActionType,value=IntegrationTest
```

Hapus tag dari tindakan kustom

Ikuti langkah-langkah ini untuk menggunakan AWS CLI untuk menghapus tag dari tindakan kustom. Saat Anda menghapus tanda dari sumber daya yang terkait, tanda akan dihapus.

> **Note**
>
> Jika Anda menghapus tindakan kustom, semua asosiasi tag akan dihapus dari tindakan kustom yang dihapus. Anda tidak perlu menghapus tag sebelum menghapus tindakan kustom.

Di terminal atau baris perintah, jalankan untag-resource perintah, tentukan ARN dari tindakan khusus tempat Anda ingin menghapus tag dan kunci tag dari tag yang ingin Anda hapus. Misalnya, untuk menghapus tag pada tindakan kustom dengan kunci tag*TestActionType*:

```
aws codepipeline untag-resource --resource-arn arn:aws:codepipeline:us-west-2:account-id:actiontype:Owner/Category/Provider/Version --tag-keys TestActionType
```

Jika berhasil, perintah ini tidak mengembalikan apa pun. Untuk memverifikasi tag yang terkait dengan tindakan kustom, jalankan list-tags-for-resource perintah.

Memanggil AWS Lambda fungsi dalam pipeline di CodePipeline

AWS Lambda adalah layanan komputasi yang memungkinkan Anda menjalankan kode tanpa perlu menyediakan atau mengelola server. Anda dapat membuat fungsi Lambda dan menambahkannya sebagai tindakan di pipeline Anda. Karena Lambda memungkinkan Anda menulis fungsi untuk melakukan hampir semua tugas, Anda dapat menyesuaikan cara kerja pipeline Anda.

> **Important**
>
> Jangan mencatat peristiwa JSON yang CodePipeline dikirim ke Lambda karena ini dapat mengakibatkan kredensi pengguna masuk Log. CloudWatch CodePipeline Peran tersebut menggunakan acara JSON untuk meneruskan kredensi sementara ke Lambda di lapangan. `artifactCredentials` Untuk contoh acara, lihatContoh acara JSON.

Berikut adalah beberapa cara fungsi Lambda dapat digunakan dalam jaringan pipa:

- Untuk membuat sumber daya sesuai permintaan dalam satu tahap pipa menggunakan CloudFormation dan menghapusnya di tahap lain.
- Untuk menerapkan versi aplikasi tanpa downtime dengan fungsi Lambda yang menukar nilai CNAME. AWS Elastic Beanstalk

- Untuk menyebarkan ke instans Amazon ECS Docker.
- Untuk mencadangkan sumber daya sebelum membangun atau menerapkan dengan membuat snapshot AMI.
- Untuk menambahkan integrasi dengan produk pihak ketiga ke pipeline Anda, seperti memposting pesan ke klien IRC.

> **Note**
>
> Membuat dan menjalankan fungsi Lambda dapat mengakibatkan biaya ke akun Anda AWS . Untuk informasi selengkapnya, silakan lihat Harga .

Topik ini mengasumsikan Anda sudah familiar AWS CodePipeline AWS Lambda dan tahu cara membuat pipeline, fungsi, dan kebijakan dan peran IAM yang menjadi sandarannya. Topik ini menunjukkan kepada Anda cara:

- Buat fungsi Lambda yang menguji apakah halaman web berhasil diterapkan.
- Konfigurasikan peran eksekusi CodePipeline dan Lambda dan izin yang diperlukan untuk menjalankan fungsi sebagai bagian dari pipeline.
- Edit pipeline untuk menambahkan fungsi Lambda sebagai tindakan.
- Uji tindakan dengan melepaskan perubahan secara manual.

> **Note**
>
> Saat menggunakan tindakan CodePipeline pemanggilan Lambda lintas wilayah, status eksekusi lambda menggunakan PutJobSuccessResultPutJobFailureResultdan harus dikirim ke Wilayah tempat fungsi Lambda hadir dan bukan ke AWS Wilayah tempat ada fungsi Lambda. CodePipeline

Topik ini mencakup fungsi sampel untuk menunjukkan fleksibilitas bekerja dengan fungsi Lambda di: CodePipeline

- Basic Lambda function
  - Membuat fungsi Lambda dasar untuk digunakan. CodePipeline

  - Mengembalikan hasil keberhasilan atau kegagalan ke CodePipeline tautan Detail untuk tindakan tersebut.
- Contoh fungsi Python yang menggunakan template AWS CloudFormation
  - Menggunakan parameter pengguna yang dikodekan JSON untuk meneruskan beberapa nilai konfigurasi ke fungsi (). `get_user_params`
  - Berinteraksi dengan artefak.zip dalam bucket artefak (). `get_template`
  - Menggunakan token lanjutan untuk memantau proses asinkron () yang berjalan lama. `continue_job_later` Ini memungkinkan tindakan untuk melanjutkan dan fungsi berhasil bahkan jika melebihi runtime lima belas menit (batas di Lambda).

Setiap fungsi sampel menyertakan informasi tentang izin yang harus Anda tambahkan ke peran. Untuk informasi tentang batasan AWS Lambda, lihat Batas dalam Panduan AWS Lambda Pengembang.

> **Important**
>
> Contoh kode, peran, dan kebijakan yang disertakan dalam topik ini hanya contoh, dan disediakan apa adanya.

Topik

- Langkah 1: Buat pipeline
- Langkah 2: Buat fungsi Lambda
- Langkah 3: Tambahkan fungsi Lambda ke pipeline di konsol CodePipeline
- Langkah 4: Uji pipa dengan fungsi Lambda
- Langkah 5: Langkah selanjutnya
- Contoh acara JSON
- Fungsi sampel tambahan

Langkah 1: Buat pipeline

Pada langkah ini, Anda membuat pipeline yang nantinya Anda tambahkan fungsi Lambda. Ini adalah pipeline yang sama yang Anda buatCodePipeline tutorial. Jika pipeline tersebut masih dikonfigurasi

untuk akun Anda dan berada di Wilayah yang sama di mana Anda berencana untuk membuat fungsi Lambda, Anda dapat melewati langkah ini.

Untuk membuat alur

1. Ikuti tiga langkah pertama Tutorial: Buat pipeline sederhana (ember S3) untuk membuat bucket Amazon S3, CodeDeploy sumber daya, dan pipeline dua tahap. Pilih opsi Amazon Linux untuk jenis instans Anda. Anda dapat menggunakan nama apa pun yang Anda inginkan untuk pipeline, tetapi langkah-langkah dalam topik ini digunakan MyLambdaTestPipeline.
2. Pada halaman status pipeline Anda, dalam CodeDeploy tindakan, pilih Detail. Pada halaman detail penerapan untuk grup penyebaran, pilih ID instans dari daftar.
3. Di konsol Amazon EC2, pada tab Detail untuk instance, salin alamat IP di IPv4 Alamat publik (misalnya,**`192.0.2.4`**). Anda menggunakan alamat ini sebagai target fungsi di AWS Lambda.

> **Note**
>
> Kebijakan peran layanan default untuk CodePipeline menyertakan izin Lambda yang diperlukan untuk menjalankan fungsi. Namun, jika Anda telah mengubah peran layanan default atau memilih peran lain, pastikan kebijakan untuk peran tersebut mengizinkan `lambda:InvokeFunction` dan `lambda:ListFunctions` izin. Jika tidak, saluran pipa yang menyertakan tindakan Lambda gagal.

Langkah 2: Buat fungsi Lambda

Pada langkah ini, Anda membuat fungsi Lambda yang membuat permintaan HTTP dan memeriksa baris teks pada halaman web. Sebagai bagian dari langkah ini, Anda juga harus membuat kebijakan IAM dan peran eksekusi Lambda. Untuk informasi selengkapnya, lihat Model Izin di Panduan AWS Lambda Pengembang.

Untuk membuat peran eksekusi

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Kebijakan, lalu pilih Buat Kebijakan. Pilih tab JSON, lalu tempelkan kebijakan berikut ke dalam bidang.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Action": [
        "codepipeline:PutJobSuccessResult",
        "codepipeline:PutJobFailureResult"
        ],
        "Effect": "Allow",
        "Resource": "*"
     }
  ]
}
```

3. Pilih Tinjau kebijakan.
4. Pada halaman Kebijakan ulasan, di Nama, ketikkan nama untuk kebijakan (misalnya,**`CodePipelineLambdaExecPolicy`**). Dalam Deskripsi, masukkan**`Enables Lambda to execute code`**.

   Pilih Buat Kebijakan.

   > **Note**
   >
   > Ini adalah izin minimum yang diperlukan untuk fungsi Lambda untuk berinteraksi dengan CodePipeline dan Amazon. CloudWatch Jika Anda ingin memperluas kebijakan ini untuk mengizinkan fungsi yang berinteraksi dengan AWS sumber daya lain, Anda harus mengubah kebijakan ini untuk mengizinkan tindakan yang diperlukan oleh fungsi Lambda tersebut.

5. Pada halaman dasbor kebijakan, pilih Peran, lalu pilih Buat peran.

6. Pada halaman Buat peran, pilih Layanan AWS. Pilih Lambda, lalu pilih Berikutnya: Izin.
7. Pada halaman Lampirkan kebijakan izin, pilih kotak centang di samping CodePipelineLambdaExecPolicy, lalu pilih Berikutnya: Tag. Pilih Berikutnya: Tinjauan.
8. Pada halaman Tinjauan, di Nama peran, masukkan nama, lalu pilih Buat peran.

Untuk membuat contoh fungsi Lambda untuk digunakan CodePipeline

1. Masuk ke Konsol Manajemen AWS dan buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pada halaman Functions, pilih Create function.

> **Note**
>
> Jika Anda melihat halaman Selamat Datang dan bukan halaman Lambda, pilih Mulai Sekarang.

3. Pilih halaman Buat fungsi, pilih Penulis dari scratch. Dalam nama Fungsi, masukkan nama untuk fungsi Lambda Anda (misalnya,**`MyLambdaFunctionForAWSCodePipeline`**). Di Runtime, pilih Node.js 20.x.
4. Di bawah Peran, pilih Pilih peran yang ada. Di Peran yang ada, pilih peran Anda, lalu pilih Buat fungsi.

   Halaman detail untuk fungsi yang Anda buat terbuka.
5. Salin kode berikut ke dalam kotak kode Fungsi:

> **Note**
>
> Objek acara, di bawah CodePipeline kunci.job, berisi rincian pekerjaan. Untuk contoh lengkap acara JSON CodePipeline kembali ke Lambda, lihat. Contoh acara 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. Tinggalkan Handler pada nilai default, dan biarkan Peran di default,**`CodePipelineLambdaExecRole`**.
7. Di Pengaturan dasar, untuk Timeout, masukkan **`20`** detik.
8. Pilih Simpan.

Langkah 3: Tambahkan fungsi Lambda ke pipeline di konsol CodePipeline

Pada langkah ini, Anda menambahkan tahap baru ke pipeline Anda, dan kemudian menambahkan tindakan Lambda yang memanggil fungsi Anda ke tahap itu.

Untuk menambahkan panggung

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pada halaman Selamat Datang, pilih pipeline yang Anda buat.
3. Pada halaman tampilan pipeline, pilih Edit.
4. Pada halaman Edit, pilih + Tambahkan tahap untuk menambahkan tahap setelah tahap penerapan dengan CodeDeploy tindakan. Masukkan nama untuk panggung (misalnya,**`LambdaStage`**), dan pilih Tambahkan tahap.

   > **Note**
   >
   > Anda juga dapat memilih untuk menambahkan tindakan Lambda Anda ke tahap yang ada. Untuk tujuan demonstrasi, kami menambahkan fungsi Lambda sebagai satu-satunya tindakan dalam satu tahap untuk memungkinkan Anda melihat kemajuannya dengan mudah saat artefak berkembang melalui pipeline.

5. Pilih + Tambahkan grup tindakan. Di Edit tindakan, di Nama tindakan, masukkan nama untuk tindakan Lambda Anda (misalnya,**`MyLambdaAction`**). Di Provider, pilih AWS Lambda. Dalam nama Fungsi, pilih atau masukkan nama fungsi Lambda Anda (misalnya,**`MyLambdaFunctionForAWSCodePipeline`**). Dalam parameter

Pengguna, tentukan alamat IP untuk instans Amazon EC2 yang Anda salin sebelumnya (misalnya,**`http://192.0.2.4`**), lalu pilih Selesai.

> **Note**
>
> Topik ini menggunakan alamat IP, tetapi dalam skenario dunia nyata, Anda dapat memberikan nama situs web terdaftar Anda sebagai gantinya (misalnya,**`http://www.example.com`**). Untuk informasi selengkapnya tentang data peristiwa dan penangan AWS Lambda, lihat Model Pemrograman di Panduan AWS Lambda Pengembang.

6. Pada halaman Edit tindakan, pilih Simpan.

Langkah 4: Uji pipa dengan fungsi Lambda

Untuk menguji fungsinya, lepaskan perubahan terbaru melalui pipa.

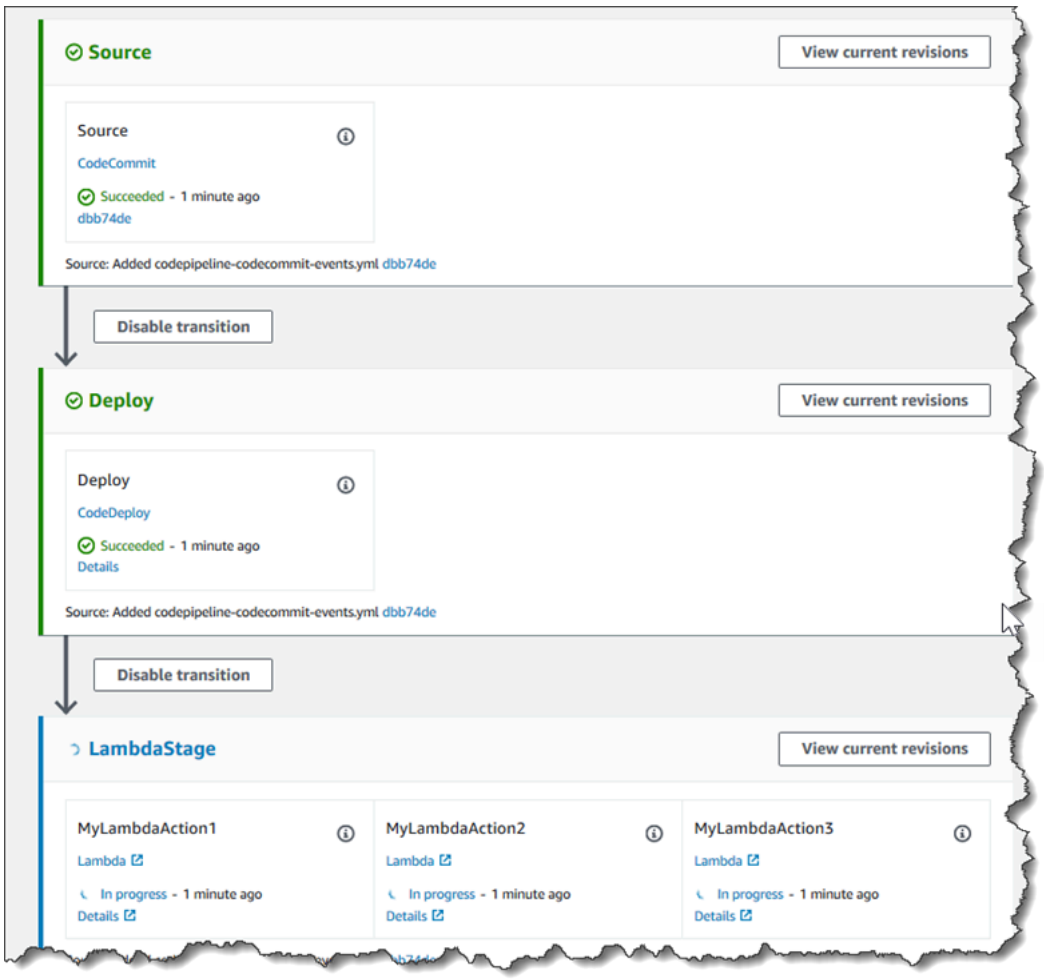
Untuk menggunakan konsol untuk menjalankan versi artefak terbaru melalui pipeline

1. Pada halaman detail pipeline, pilih Rilis perubahan. Ini menjalankan revisi terbaru yang tersedia di setiap lokasi sumber yang ditentukan dalam aksi sumber melalui pipeline.
2. Saat tindakan Lambda selesai, pilih tautan Detail untuk melihat aliran log untuk fungsi di Amazon CloudWatch, termasuk durasi acara yang ditagih. Jika fungsi gagal, CloudWatch log memberikan informasi tentang penyebabnya.

Langkah 5: Langkah selanjutnya

Sekarang setelah Anda berhasil membuat fungsi Lambda dan menambahkannya sebagai tindakan dalam pipeline, Anda dapat mencoba yang berikut ini:

- Tambahkan lebih banyak tindakan Lambda ke panggung Anda untuk memeriksa halaman web lainnya.
- Ubah fungsi Lambda untuk memeriksa string teks yang berbeda.
- Jelajahi fungsi Lambda dan buat serta tambahkan fungsi Lambda Anda sendiri ke saluran pipa.



Setelah Anda selesai bereksperimen dengan fungsi Lambda, pertimbangkan untuk menghapusnya dari pipeline Anda, menghapusnya, dan menghapus peran AWS Lambda dari IAM untuk menghindari kemungkinan biaya. Untuk informasi selengkapnya, lihat Edit pipa di CodePipelineHapus pipa di CodePipeline, dan Menghapus Peran atau Profil Instance.

Contoh acara JSON

Contoh berikut menunjukkan contoh peristiwa JSON dikirim ke CodePipeline Lambda oleh. Struktur acara ini mirip dengan respon terhadapGetJobDetails API, tetapi tanpa `actionTypeId` dan tipe `pipelineContext` data. Dua detail konfigurasi tindakan`UserParameters`, `FunctionName` dan, disertakan dalam acara JSON dan respons terhadap `GetJobDetails` API. Nilai-nilai dalam *`red italic text`* adalah contoh atau penjelasan, bukan nilai nyata.

```
{
    "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"
            }
        }
    }
}
```

Fungsi sampel tambahan

Contoh fungsi Lambda berikut menunjukkan fungsionalitas tambahan yang dapat Anda gunakan untuk saluran pipa Anda. CodePipeline Untuk menggunakan fungsi ini, Anda mungkin harus mengubah kebijakan untuk peran eksekusi Lambda, seperti yang tercantum dalam pendahuluan untuk setiap sampel.

Topik

- Contoh fungsi Python yang menggunakan template AWS CloudFormation

Contoh fungsi Python yang menggunakan template AWS CloudFormation

Contoh berikut menunjukkan fungsi yang membuat atau memperbarui tumpukan berdasarkan CloudFormation template yang disediakan. Template membuat bucket Amazon S3. Ini hanya untuk tujuan demonstrasi, untuk meminimalkan biaya. Idealnya, Anda harus menghapus tumpukan sebelum mengunggah apa pun ke ember. Jika Anda mengunggah file ke bucket, Anda tidak dapat menghapus bucket saat menghapus tumpukan. Anda harus menghapus semua yang ada di ember secara manual sebelum Anda dapat menghapus ember itu sendiri.

Contoh Python ini mengasumsikan Anda memiliki pipeline yang menggunakan bucket Amazon S3 sebagai aksi sumber, atau Anda memiliki akses ke bucket Amazon S3 berversi yang dapat Anda gunakan dengan pipeline. Anda membuat CloudFormation template, mengompresnya, dan mengunggahnya ke bucket itu sebagai file.zip. Anda kemudian harus menambahkan aksi sumber ke pipeline Anda yang mengambil file.zip ini dari bucket.

> **Note**
>
> Jika Amazon S3 adalah penyedia sumber untuk pipeline Anda, Anda dapat mem-zip file sumber atau file ke dalam satu .zip dan mengunggah.zip ke bucket sumber Anda. Anda juga

dapat mengunggah satu file yang tidak di-zip; namun, tindakan hilir yang mengharapkan file.zip akan gagal.

Sampel ini menunjukkan:

- Penggunaan parameter pengguna yang dikodekan JSON untuk meneruskan beberapa nilai konfigurasi ke fungsi (). `get_user_params`
- Interaksi dengan artefak .zip dalam bucket artefak (). `get_template`
- Penggunaan token lanjutan untuk memantau proses asinkron () yang berjalan lama. `continue_job_later` Ini memungkinkan tindakan untuk melanjutkan dan fungsi berhasil bahkan jika melebihi runtime lima belas menit (batas di Lambda).

Untuk menggunakan contoh fungsi Lambda ini, kebijakan untuk peran eksekusi Lambda harus memiliki `Allow` izin di, Amazon CloudFormation S3, dan CodePipeline, seperti yang ditunjukkan dalam kebijakan contoh ini:

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": "*"
        }
    ]
}
```

Untuk membuat CloudFormation template, buka editor teks biasa dan salin dan tempel kode berikut:

```
{
  "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"
    }
  }
}
```

Simpan ini sebagai file JSON dengan nama **`template.json`** dalam direktori bernama**`template-package`**. Buat file terkompresi (.zip) dari direktori ini dan file bernama**`template-package.zip`**, dan unggah file terkompresi ke bucket Amazon S3 berversi. Jika Anda sudah memiliki bucket yang dikonfigurasi untuk pipeline Anda, Anda dapat menggunakannya. Selanjutnya, edit pipeline

Anda untuk menambahkan aksi sumber yang mengambil file.zip. Beri nama output untuk tindakan ini*MyTemplate*. Untuk informasi selengkapnya, lihat Edit pipa di CodePipeline.

> **Note**
>
> Fungsi Lambda sampel mengharapkan nama file ini dan struktur terkompresi. Namun, Anda dapat mengganti CloudFormation template Anda sendiri untuk sampel ini. Jika Anda menggunakan template Anda sendiri, pastikan Anda mengubah kebijakan untuk peran eksekusi Lambda untuk mengizinkan fungsionalitas tambahan yang diperlukan oleh template Anda CloudFormation .

Untuk menambahkan kode berikut sebagai fungsi di Lambda

1. Buka konsol Lambda dan pilih Buat fungsi.
2. Pilih halaman Buat fungsi, pilih Penulis dari scratch. Di nama Fungsi, masukkan nama untuk fungsi Lambda Anda.
3. Di Runtime, pilih Python 2.7.
4. Di bawah Pilih atau buat peran eksekusi, pilih Gunakan peran yang ada. Di Peran yang ada, pilih peran Anda, lalu pilih Buat fungsi.

   Halaman detail untuk fungsi yang Anda buat terbuka.
5. Salin kode berikut ke dalam kotak kode Fungsi:

```
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. Biarkan Handler pada nilai default, dan biarkan Peran pada nama yang Anda pilih atau buat sebelumnya,**`CodePipelineLambdaExecRole`**.
7. Dalam pengaturan Dasar, untuk Timeout, ganti default 3 detik dengan**`20`**.
8. Pilih Simpan.
9. Dari CodePipeline konsol, edit pipeline untuk menambahkan fungsi sebagai tindakan dalam tahap di pipeline Anda. Pilih Edit untuk tahap pipeline yang ingin Anda ubah, dan pilih Tambahkan grup tindakan. Pada halaman Edit tindakan, dalam nama Tindakan, masukkan nama untuk tindakan Anda. Di penyedia Tindakan, pilih Lambda.

   Di bawah Input artefak, pilih`MyTemplate`. Di UserParameters, Anda harus menyediakan string JSON dengan tiga parameter:

   - Nama tumpukan
   - CloudFormation nama template dan path ke file
   - Artefak masukan

   Gunakan kurung kurawal ({}) dan pisahkan parameter dengan koma. Misalnya, untuk membuat tumpukan bernama*`MyTestStack`*, untuk pipeline dengan artefak masukan, masukkan: {"stack”:” *`MyTestStack`* “UserParameters, "file” :"template-package/template.json”, "artifact*`MyTemplate`*”:” “}. *`MyTemplate`*

> **Note**
>
> Meskipun Anda telah menentukan artefak input di UserParameters, Anda juga harus menentukan artefak masukan ini untuk tindakan dalam artefak Input.

10. Simpan perubahan Anda ke pipeline, lalu lepaskan perubahan secara manual untuk menguji tindakan dan fungsi Lambda.

Menambahkan tindakan persetujuan manual ke panggung

Di AWS CodePipeline, Anda dapat menambahkan tindakan persetujuan ke tahap dalam pipeline pada titik di mana Anda ingin eksekusi pipeline dihentikan sehingga seseorang dengan AWS Identity and Access Management izin yang diperlukan dapat menyetujui atau menolak tindakan tersebut.

Jika tindakan disetujui, eksekusi pipa dilanjutkan. Jika tindakan ditolak—atau jika tidak ada yang menyetujui atau menolak tindakan dalam waktu tujuh hari setelah pipeline mencapai tindakan dan penghentian—hasilnya sama dengan tindakan yang gagal, dan eksekusi pipeline tidak berlanjut.

Anda dapat menggunakan persetujuan manual untuk alasan ini:

- Anda ingin seseorang melakukan peninjauan kode atau mengubah tinjauan manajemen sebelum revisi diizinkan ke tahap pipa berikutnya.
- Anda ingin seseorang melakukan pengujian jaminan kualitas manual pada versi terbaru aplikasi, atau untuk mengkonfirmasi integritas artefak build, sebelum dirilis.
- Anda ingin seseorang meninjau teks baru atau yang diperbarui sebelum dipublikasikan ke situs web perusahaan.

Opsi konfigurasi untuk tindakan persetujuan manual di CodePipeline

CodePipeline menyediakan tiga opsi konfigurasi yang dapat Anda gunakan untuk memberi tahu pemberi persetujuan tentang tindakan persetujuan.

Mempublikasikan Pemberitahuan Persetujuan Anda dapat mengonfigurasi tindakan persetujuan untuk memublikasikan pesan ke topik Layanan Pemberitahuan Sederhana Amazon saat pipeline berhenti pada tindakan. Amazon SNS mengirimkan pesan ke setiap titik akhir yang berlangganan topik. Anda harus menggunakan topik yang dibuat di AWS Wilayah yang sama dengan pipeline yang akan menyertakan tindakan persetujuan. Saat Anda membuat topik, kami sarankan Anda

memberinya nama yang akan mengidentifikasi tujuannya, dalam format sepertiMy`FirstPipeline-us-east-2-approval`.

Saat mempublikasikan pemberitahuan persetujuan ke topik Amazon SNS, Anda dapat memilih dari format seperti penerima email atau SMS, antrian SQS, HTTP/HTTPS titik akhir, atau fungsi yang Anda AWS Lambda panggil menggunakan Amazon SNS. Untuk informasi tentang notifikasi topik Amazon SNS, lihat topik berikut:

- Apa itu Layanan Pemberitahuan Sederhana Amazon?
- Buat Topik di Amazon SNS
- Mengirim Pesan Amazon SNS ke Antrian Amazon SQS
- Berlangganan Antrian ke Topik Amazon SNS
- Mengirim Pesan Amazon SNS ke Endpoint HTTP/HTTPS
- Memanggil Fungsi Lambda Menggunakan Notifikasi Amazon SNS

Untuk struktur data JSON yang dihasilkan untuk pemberitahuan tindakan persetujuan, lihatFormat data JSON untuk pemberitahuan persetujuan manual di CodePipeline.

Tentukan URL untuk Tinjauan Sebagai bagian dari konfigurasi tindakan persetujuan, Anda dapat menentukan URL yang akan ditinjau. URL mungkin berupa tautan ke aplikasi web yang ingin Anda uji oleh pemberi persetujuan atau halaman dengan informasi lebih lanjut tentang permintaan persetujuan Anda. URL disertakan dalam notifikasi yang dipublikasikan ke topik Amazon SNS. Pemberi persetujuan dapat menggunakan konsol atau CLI untuk melihatnya.

Masukkan Komentar untuk Penyetuju Saat Anda membuat tindakan persetujuan, Anda juga dapat menambahkan komentar yang ditampilkan kepada mereka yang menerima pemberitahuan atau mereka yang melihat tindakan di konsol atau respons CLI.

Tidak Ada Opsi Konfigurasi Anda juga dapat memilih untuk tidak mengonfigurasi salah satu dari tiga opsi ini. Anda mungkin tidak membutuhkannya jika, misalnya, Anda dapat memberi tahu seseorang secara langsung bahwa tindakan tersebut siap untuk ditinjau, atau Anda hanya ingin saluran pipa berhenti sampai Anda memutuskan untuk menyetujui tindakan itu sendiri.

Ikhtisar penyiapan dan alur kerja untuk tindakan persetujuan di CodePipeline

Berikut ini adalah ikhtisar untuk menyiapkan dan menggunakan persetujuan manual.

1. Anda memberikan izin IAM yang diperlukan untuk menyetujui atau menolak tindakan persetujuan untuk satu atau beberapa peran IAM di organisasi Anda.
2. (Opsional) Jika Anda menggunakan notifikasi Amazon SNS, Anda memastikan bahwa peran layanan yang Anda gunakan dalam CodePipeline operasi memiliki izin untuk mengakses sumber daya Amazon SNS.
3. (Opsional) Jika Anda menggunakan notifikasi Amazon SNS, Anda membuat topik Amazon SNS dan menambahkan satu atau beberapa pelanggan atau titik akhir ke dalamnya.
4. Saat Anda menggunakan AWS CLI untuk membuat pipeline atau setelah Anda menggunakan CLI atau konsol untuk membuat pipeline, Anda menambahkan tindakan persetujuan ke tahap dalam pipeline.

   Jika Anda menggunakan notifikasi, Anda menyertakan Nama Sumber Daya Amazon (ARN) dari topik Amazon SNS dalam konfigurasi tindakan. (ARN adalah pengidentifikasi unik untuk sumber daya Amazon. ARNs untuk Amazon SNS topik terstruktur seperti. *`arn:aws:sns:us-east-2:80398EXAMPLE:MyApprovalTopic`* Untuk informasi selengkapnya, lihat Amazon Resource Layanan AWS Names (ARNs) dan ruang nama di Referensi Umum Amazon Web Services.)
5. Pipa berhenti ketika mencapai tindakan persetujuan. Jika ARN topik Amazon SNS disertakan dalam konfigurasi tindakan, pemberitahuan akan dipublikasikan ke topik Amazon SNS, dan pesan dikirimkan ke pelanggan mana pun ke topik atau titik akhir berlangganan, dengan tautan untuk meninjau tindakan persetujuan di konsol.
6. Pemberi persetujuan memeriksa URL target dan mengulas komentar, jika ada.
7. Menggunakan konsol, CLI, atau SDK, pemberi persetujuan memberikan komentar ringkasan dan mengirimkan tanggapan:
   - Disetujui: Eksekusi pipa dilanjutkan.
   - Ditolak: Status tahap diubah menjadi “Gagal” dan eksekusi pipeline tidak dilanjutkan.

   Jika tidak ada tanggapan yang dikirimkan dalam waktu tujuh hari, tindakan ditandai sebagai “Gagal.”

Berikan izin persetujuan kepada pengguna IAM di CodePipeline

Sebelum pengguna IAM di organisasi Anda dapat menyetujui atau menolak tindakan persetujuan, mereka harus diberikan izin untuk mengakses saluran pipa dan memperbarui status tindakan persetujuan. Anda dapat memberikan izin untuk mengakses semua pipeline dan tindakan

persetujuan di akun Anda dengan melampirkan kebijakan `AWSCodePipelineApproverAccess` terkelola ke pengguna, peran, atau grup IAM; atau Anda dapat memberikan izin terbatas dengan menentukan sumber daya individual yang dapat diakses oleh pengguna, peran, atau grup IAM.

> **Note**
>
> Izin yang dijelaskan dalam topik ini memberikan akses yang sangat terbatas. Agar pengguna, peran, atau grup dapat melakukan lebih dari sekadar menyetujui atau menolak tindakan persetujuan, Anda dapat melampirkan kebijakan terkelola lainnya. Untuk informasi tentang kebijakan terkelola yang tersedia CodePipeline, lihatAWS kebijakan terkelola untuk AWS CodePipeline.

Berikan izin persetujuan untuk semua jalur pipa dan tindakan persetujuan

Untuk pengguna yang perlu melakukan tindakan persetujuan CodePipeline, gunakan kebijakan `AWSCodePipelineApproverAccess` terkelola.

Untuk memberikan akses dan menambahkan izin bagi pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti instruksi di Buat rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti instruksi dalam Buat peran untuk penyedia identitas pihak ketiga (federasi) dalam Panduan Pengguna IAM.

- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Buat peran untuk pengguna IAM dalam Panduan Pengguna IAM.
  - (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

Tentukan izin persetujuan untuk jalur pipa tertentu dan tindakan persetujuan

Untuk pengguna yang perlu melakukan tindakan persetujuan CodePipeline, gunakan kebijakan kustom berikut. Dalam kebijakan di bawah ini, tentukan sumber daya individu yang dapat diakses pengguna. Misalnya, kebijakan berikut memberi pengguna wewenang untuk menyetujui atau menolak hanya tindakan yang disebutkan `MyApprovalAction` dalam `MyFirstPipeline` pipeline di Wilayah AS Timur (Ohio) (us-east-2):

> **Note**
>
> `codepipeline:ListPipelines`Izin diperlukan hanya jika pengguna IAM perlu mengakses CodePipeline dasbor untuk melihat daftar saluran pipa ini. Jika akses konsol tidak diperlukan, Anda dapat menghilangkannya`codepipeline:ListPipelines`.

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan dokumen kebijakan JSON berikut:

```
{
   "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. Pilih Berikutnya.

> **Note**
>
> Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Berikan izin Amazon SNS ke peran layanan CodePipeline

Jika Anda berencana menggunakan Amazon SNS untuk mempublikasikan pemberitahuan ke topik saat tindakan persetujuan memerlukan peninjauan, peran layanan yang Anda gunakan dalam

CodePipeline operasi harus diberikan izin untuk mengakses sumber daya Amazon SNS. Anda dapat menggunakan konsol IAM untuk menambahkan izin ini ke peran layanan Anda.

Dalam kebijakan di bawah ini, tentukan kebijakan penerbitan dengan SNS. Untuk kebijakan berikut, Anda dapat menamainya`SNSPublish`. Gunakan kebijakan berikut dengan melampirkannya ke peran layanan Anda.

> **Important**
>
> Pastikan Anda masuk dengan informasi akun yang sama Konsol Manajemen AWS dengan yang Anda gunakanMemulai dengan CodePipeline.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "*"
        }
    ]
}
```

Cara menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pada panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul halaman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.
3. Di bagian atas halaman, pilih Buat kebijakan.
4. Di bagian Editor kebijakan, pilih opsi JSON.
5. Masukkan atau tempel dokumen kebijakan JSON. Untuk detail bahasa kebijakan IAM, lihat Referensi kebijakan JSON IAM.

6. Selesaikan peringatan keamanan, kesalahan, atau peringatan umum yang dihasilkan selama validasi kebijakan, lalu pilih Berikutnya.

   > **Note**
   >
   > Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda melakukan perubahan atau memilih Berikutnya di editor Visual, IAM dapat merestrukturisasi kebijakan Anda untuk mengoptimalkannya bagi editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. (Opsional) Saat membuat atau mengedit kebijakan Konsol Manajemen AWS, Anda dapat membuat templat kebijakan JSON atau YAMAL yang dapat Anda gunakan dalam CloudFormation templat.

   Untuk melakukannya, di editor Kebijakan pilih Tindakan, lalu pilih Buat CloudFormation templat. Untuk mempelajari selengkapnya CloudFormation, lihat referensi jenis AWS Identity and Access Management sumber daya di Panduan AWS CloudFormation Pengguna.

8. Setelah selesai menambahkan izin ke kebijakan, pilih Berikutnya.
9. Pada halaman Tinjau dan buat, masukkan Nama kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
10. (Opsional) Tambahkan metadata ke kebijakan dengan melampirkan tanda sebagai pasangan nilai kunci. Untuk informasi selengkapnya tentang penggunaan tag di IAM, lihat Tag untuk AWS Identity and Access Management sumber daya di Panduan Pengguna IAM.
11. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Tambahkan tindakan persetujuan manual ke pipeline di CodePipeline

Anda dapat menambahkan tindakan persetujuan ke tahap dalam CodePipeline pipeline pada titik di mana Anda ingin pipeline berhenti sehingga seseorang dapat menyetujui atau menolak tindakan secara manual.

> **Note**
>
> Tindakan persetujuan tidak dapat ditambahkan ke tahap Sumber. Tahapan sumber hanya dapat berisi tindakan sumber.

Jika Anda ingin menggunakan Amazon SNS untuk mengirim notifikasi saat tindakan persetujuan siap ditinjau, Anda harus terlebih dahulu menyelesaikan prasyarat berikut:

- Berikan izin ke peran CodePipeline layanan Anda untuk mengakses sumber daya Amazon SNS. Untuk informasi, lihat Berikan izin Amazon SNS ke peran layanan CodePipeline.
- Berikan izin kepada satu atau beberapa identitas IAM di organisasi Anda untuk memperbarui status tindakan persetujuan. Untuk informasi, lihat Berikan izin persetujuan kepada pengguna IAM di CodePipeline.

Dalam contoh ini, Anda membuat tahap persetujuan baru dan menambahkan tindakan persetujuan manual ke panggung. Anda juga dapat menambahkan tindakan persetujuan manual ke tahap yang ada yang berisi tindakan lain.

Menambahkan tindakan persetujuan manual ke CodePipeline pipeline (konsol)

Anda dapat menggunakan CodePipeline konsol untuk menambahkan tindakan persetujuan ke CodePipeline pipeline yang ada. Anda harus menggunakan AWS CLI jika ingin menambahkan tindakan persetujuan saat membuat pipeline baru.

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Dalam Nama, pilih pipeline.
3. Pada halaman detail pipeline, pilih Edit.
4. Jika Anda ingin menambahkan tindakan persetujuan ke tahap baru, pilih + Tambahkan tahap pada titik di pipeline tempat Anda ingin menambahkan permintaan persetujuan, dan masukkan nama untuk tahap tersebut. Pada halaman Add stage, dalam nama Stage, masukkan nama panggung baru Anda. Misalnya, tambahkan tahap baru dan beri nama`Manual_Approval`.

   Jika Anda ingin menambahkan tindakan persetujuan ke tahap yang ada, pilih Edit tahap.
5. Pada tahap di mana Anda ingin menambahkan tindakan persetujuan, pilih + Tambahkan grup tindakan.
6. Pada halaman Edit tindakan, lakukan hal berikut:

   1. Di Nama tindakan, masukkan nama untuk mengidentifikasi tindakan.
   2. Di penyedia Tindakan, di bawah Persetujuan, pilih Persetujuan manual.
   3. (Opsional) Dalam topik SNS ARN, pilih nama topik yang akan digunakan untuk mengirim pemberitahuan untuk tindakan persetujuan.

4. (Opsional) Di URL untuk ditinjau, masukkan URL halaman atau aplikasi yang ingin diperiksa oleh pemberi persetujuan. Pemberi persetujuan dapat mengakses URL ini melalui tautan yang disertakan dalam tampilan konsol pipeline.
5. (Opsional) Di Komentar, masukkan informasi lain yang ingin Anda bagikan dengan pengulas.
6. Pilih Simpan.

Tambahkan tindakan persetujuan manual ke CodePipeline pipeline (CLI)

Anda dapat menggunakan CLI untuk menambahkan tindakan persetujuan ke pipeline yang ada atau saat Anda membuat pipeline. Anda melakukan ini dengan menyertakan tindakan persetujuan, dengan jenis persetujuan Manual, dalam tahap yang Anda buat atau edit.

Untuk informasi selengkapnya tentang membuat dan mengedit pipeline, lihat Buat pipeline, tahapan, dan tindakan danEdit pipa di CodePipeline.

Untuk menambahkan tahapan ke pipeline yang hanya menyertakan tindakan persetujuan, Anda akan menyertakan sesuatu yang mirip dengan contoh berikut saat membuat atau memperbarui pipeline.

> **Note**
>
> `configuration`Bagian ini opsional. Ini hanya sebagian, bukan seluruh struktur, dari file. Untuk informasi selengkapnya, lihat CodePipeline referensi struktur pipa.

```
{
    "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
        }
    ]
}
```

Jika tindakan persetujuan berada dalam tahap dengan tindakan lain, bagian file JSON Anda yang berisi tahapan mungkin terlihat mirip dengan contoh berikut.

> **Note**
>
> `configuration`Bagian ini opsional. Ini hanya sebagian, bukan seluruh struktur, dari file. Untuk informasi selengkapnya, lihat CodePipeline referensi struktur pipa.

```
,
{
    "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
        }
    ]
}
```

Menyetujui atau menolak tindakan persetujuan di CodePipeline

Ketika pipeline menyertakan tindakan persetujuan, eksekusi pipeline berhenti pada titik di mana tindakan telah ditambahkan. Pipeline tidak akan dilanjutkan kecuali seseorang menyetujui tindakan secara manual. Jika pemberi persetujuan menolak tindakan tersebut, atau jika tidak ada tanggapan persetujuan yang diterima dalam waktu tujuh hari setelah pipa berhenti untuk tindakan persetujuan, status pipeline menjadi “Gagal.”

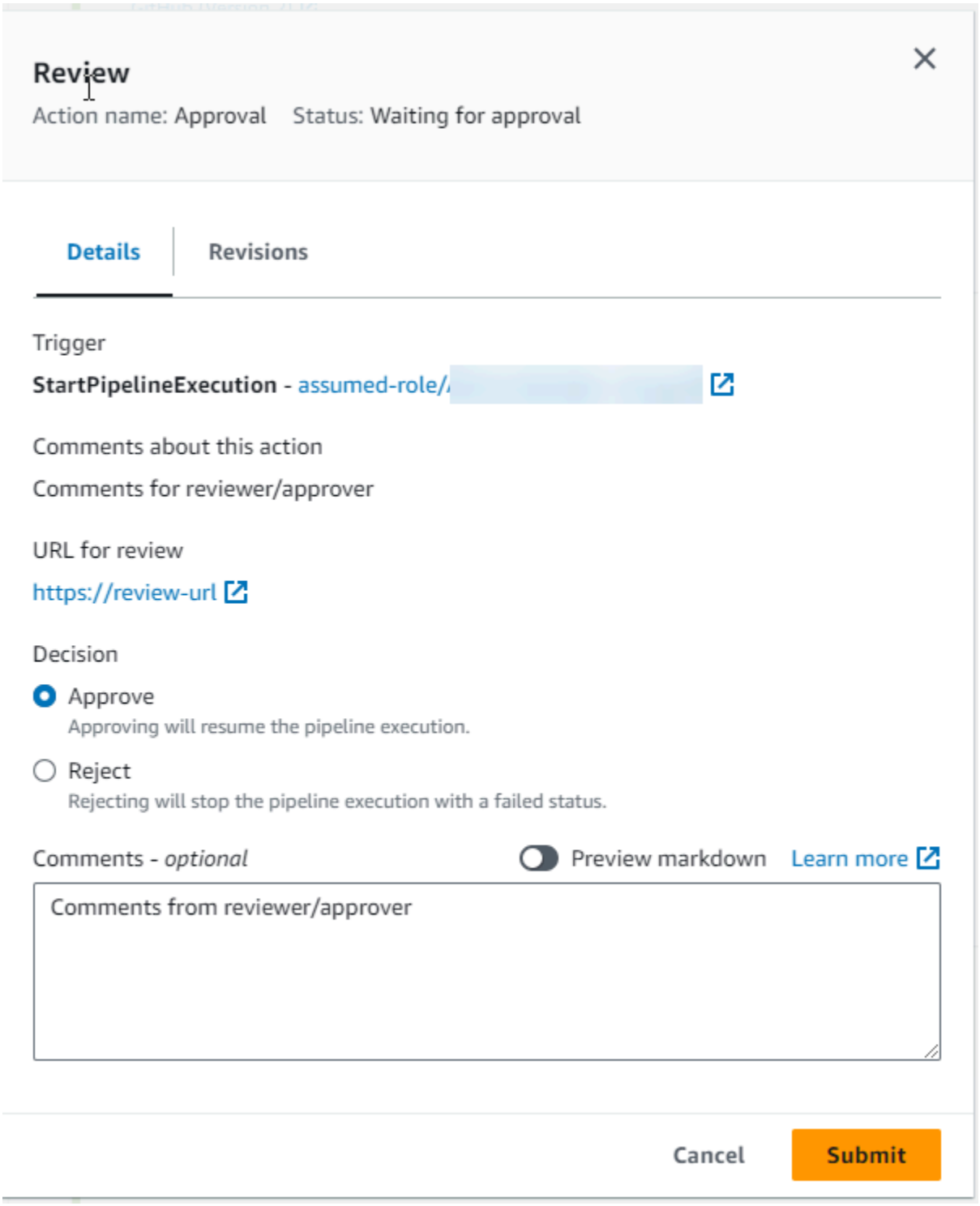
Jika orang yang menambahkan tindakan persetujuan ke pipeline mengonfigurasi notifikasi, Anda mungkin menerima email dengan informasi pipeline dan status untuk persetujuan.

Menyetujui atau menolak tindakan persetujuan (konsol)

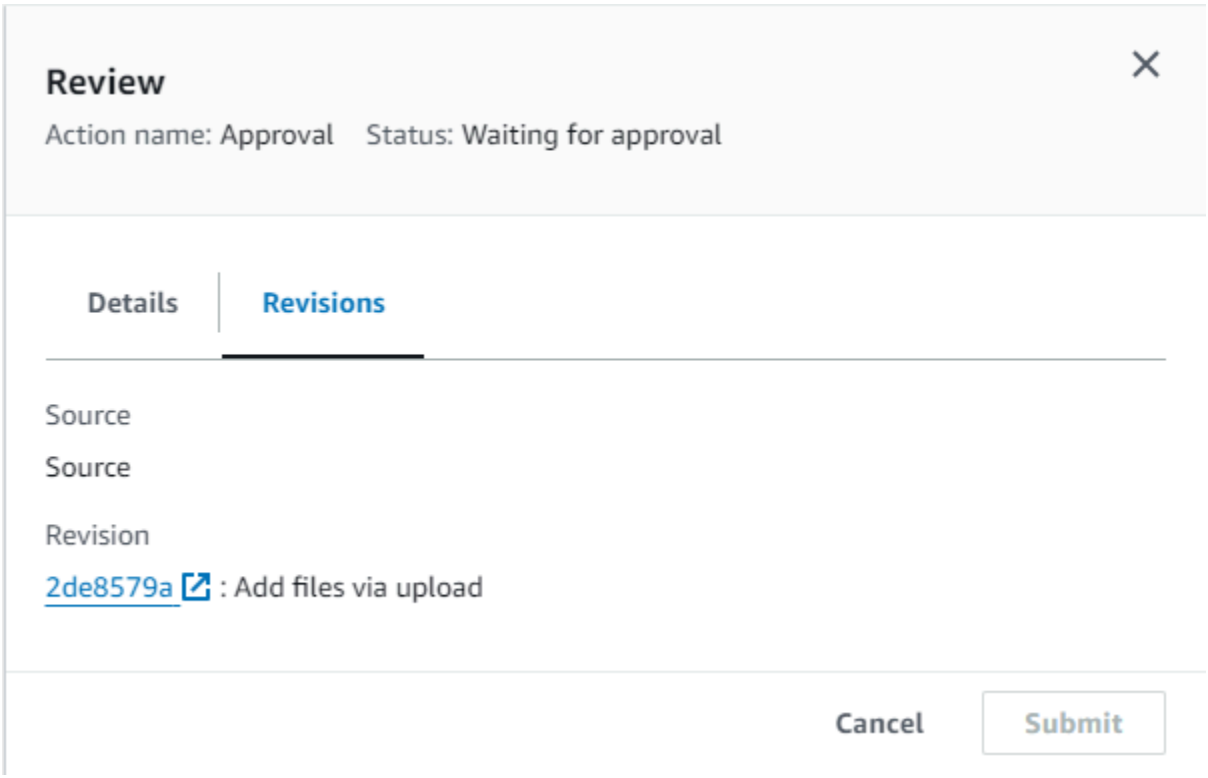
Jika Anda menerima pemberitahuan yang menyertakan tautan langsung ke tindakan persetujuan, pilih tautan Setujui atau tolak, masuk ke konsol, lalu lanjutkan dengan langkah 7 di sini. Jika tidak, ikuti semua langkah ini.

1. Buka CodePipeline konsol di https://console.aws.amazon.com/codepipeline/.
2. Pada halaman All Pipelines, pilih nama pipeline.
3. Temukan panggung dengan tindakan persetujuan. Pilih Tinjau.

   Kotak dialog Review ditampilkan. Tab Detail menampilkan konten ulasan dan komentar.



Tab Revisi menunjukkan revisi sumber untuk eksekusi.



4. Pada tab Detail, lihat komentar dan URL, jika ada. Pesan juga menampilkan URL konten untuk Anda tinjau, jika ada yang disertakan.
5. Jika URL disediakan, pilih URL untuk tautan peninjauan dalam tindakan untuk membuka halaman web target, lalu tinjau konten.
6. Di jendela Tinjau, masukkan komentar ulasan, seperti alasan Anda menyetujui atau menolak tindakan, lalu pilih Menyetujui atau Tolak.
7. Pilih Kirim.

Menyetujui atau menolak permintaan persetujuan (CLI)

Untuk menggunakan CLI untuk menanggapi tindakan persetujuan, Anda harus terlebih dahulu menggunakan get-pipeline-state perintah untuk mengambil token yang terkait dengan eksekusi terbaru dari tindakan persetujuan.

1. Di terminal (Linux, macOS, atau Unix) atau command prompt (Windows), jalankan get-pipeline-stateperintah pada pipeline yang berisi tindakan persetujuan. Misalnya, untuk pipeline bernama*MyFirstPipeline*, masukkan yang berikut ini:

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

2. Dalam menanggapi perintah, cari `token` nilai, yang muncul `latestExecution` di `actionStates` bagian untuk tindakan persetujuan, seperti yang ditunjukkan di sini:

```
{
    "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. Dalam editor teks biasa, buat file tempat Anda menambahkan yang berikut ini, dalam format JSON:

   - Nama pipa yang berisi tindakan persetujuan.
   - Nama panggung yang berisi tindakan persetujuan.
   - Nama tindakan persetujuan.
   - Nilai token yang Anda kumpulkan pada langkah sebelumnya.
   - Tanggapan Anda terhadap tindakan (Disetujui atau Ditolak). Respons harus dikapitalisasi.
   - Komentar ringkasan Anda.

   Untuk *MyFirstPipeline* contoh sebelumnya, file Anda akan terlihat seperti ini:

```
{
```

```
  "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. Simpan file dengan nama seperti**approvalstage-approved.json**.
5. Jalankan put-approval-resultperintah, tentukan nama file JSON persetujuan, mirip dengan yang berikut ini:

> **Important**
>
> Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

```
aws codepipeline put-approval-result --cli-input-json file://approvalstage-approved.json
```

Format data JSON untuk pemberitahuan persetujuan manual di CodePipeline

Untuk tindakan persetujuan yang menggunakan notifikasi Amazon SNS, data JSON tentang tindakan dibuat dan dipublikasikan ke Amazon SNS saat pipeline berhenti. Anda dapat menggunakan output JSON untuk mengirim pesan ke antrian Amazon SQS atau menjalankan fungsi di. AWS Lambda

> **Note**
>
> Panduan ini tidak membahas cara mengonfigurasi notifikasi menggunakan JSON. Untuk selengkapnya, lihat Mengirim Pesan Amazon SNS ke Antrian Amazon SQS dan Memanggil Fungsi Lambda Menggunakan Notifikasi Amazon SNS di Panduan Pengembang Amazon SNS.

Contoh berikut menunjukkan struktur output JSON yang tersedia dengan CodePipeline persetujuan.

```
{
    "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."
    }
}
```

Menambahkan tindakan Lintas wilayah di CodePipeline

AWS CodePipeline menyertakan sejumlah tindakan yang membantu Anda mengonfigurasi pembuatan, pengujian, dan penerapan sumber daya untuk proses rilis otomatis Anda. Anda dapat menambahkan tindakan ke pipeline yang berada di AWS Wilayah yang berbeda dari pipeline Anda. Jika Layanan AWS adalah penyedia tindakan, dan type/provider tipe tindakan ini berada di AWS Wilayah yang berbeda dari pipeline Anda, ini adalah tindakan Lintas wilayah.

> **Note**
>
> Tindakan lintas wilayah didukung dan hanya dapat dibuat di AWS Wilayah yang CodePipeline didukung. Untuk daftar AWS Wilayah yang didukung CodePipeline, lihatKuota di AWS CodePipeline.

Anda dapat menggunakan konsol, AWS CLI, atau CloudFormation untuk menambahkan tindakan lintas wilayah di pipeline.

> **Note**
>
> Jenis tindakan tertentu CodePipeline mungkin hanya tersedia di AWS Wilayah tertentu. Perhatikan juga bahwa mungkin ada AWS Wilayah di mana tipe tindakan tersedia, tetapi AWS penyedia khusus untuk jenis tindakan tersebut tidak tersedia.

Saat membuat atau mengedit pipeline, Anda harus memiliki bucket artefak di Wilayah pipeline dan kemudian Anda harus memiliki satu bucket artefak per Wilayah tempat Anda berencana untuk menjalankan suatu tindakan. Untuk informasi tentang parameter `ArtifactStores`, lihat CodePipeline referensi struktur pipa.

> **Note**
>
> CodePipeline menangani penyalinan artefak dari satu AWS Wilayah ke Wilayah lain saat melakukan tindakan lintas wilayah.

Jika Anda menggunakan konsol untuk membuat pipeline atau tindakan lintas wilayah, bucket artefak default dikonfigurasi oleh CodePipeline di Wilayah tempat Anda memiliki tindakan. Saat Anda menggunakan AWS CLI, CloudFormation, atau SDK untuk membuat pipeline atau tindakan lintas wilayah, Anda menyediakan bucket artefak untuk setiap Wilayah tempat Anda memiliki tindakan.

> **Note**
>
> Anda harus membuat bucket artefak dan kunci enkripsi di AWS Wilayah yang sama dengan tindakan Lintas wilayah dan di akun yang sama dengan pipeline Anda.

Anda tidak dapat membuat tindakan lintas wilayah untuk jenis tindakan berikut:

- Tindakan sumber
- Tindakan pihak ketiga
- Tindakan kustom

> **Note**
>
> Saat menggunakan tindakan CodePipeline pemanggilan Lambda lintas wilayah, status eksekusi lambda menggunakan PutJobSuccessResultPutJobFailureResultdan harus dikirim ke Wilayah tempat fungsi Lambda hadir dan bukan ke AWS Wilayah tempat ada fungsi Lambda. CodePipeline

Ketika pipeline menyertakan aksi Lintas wilayah sebagai bagian dari tahap, hanya CodePipeline mereplikasi artefak masukan dari aksi Lintas wilayah dari Wilayah pipa ke Wilayah aksi.

> **Note**
>
> Wilayah pipeline dan Wilayah tempat sumber daya deteksi perubahan CloudWatch Peristiwa Anda dipertahankan tetap sama. Wilayah tempat pipeline Anda di-host tidak berubah.

Mengelola tindakan lintas wilayah dalam pipeline (konsol)

Anda dapat menggunakan CodePipeline konsol untuk menambahkan tindakan Lintas wilayah ke pipeline yang ada. Untuk membuat pipeline baru dengan tindakan lintas wilayah menggunakan wizard Buat pipeline, lihatBuat pipeline khusus (konsol).

Di konsol, Anda membuat tindakan lintas wilayah dalam tahap pipeline dengan memilih penyedia tindakan dan bidang Wilayah, yang mencantumkan sumber daya yang telah Anda buat di wilayah tersebut untuk penyedia tersebut. Saat menambahkan tindakan Lintas wilayah, CodePipeline gunakan bucket artefak terpisah di wilayah tindakan. Untuk informasi lebih lanjut tentang ember artefak lintas wilayah, lihat. CodePipeline referensi struktur pipa

Menambahkan tindakan Lintas wilayah ke tahap pipeline (konsol)

Gunakan konsol untuk menambahkan tindakan Lintas wilayah ke pipeline.

> **Note**
>
> Jika pipeline berjalan saat perubahan disimpan, eksekusi itu tidak selesai.

Untuk menambahkan tindakan Lintas wilayah

1. Masuk ke konsol di http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Pilih pipeline Anda, lalu pilih Edit.
3. Di bagian bawah diagram, pilih + Tambahkan tahap jika Anda menambahkan tahap baru, atau pilih Edit tahap jika Anda ingin menambahkan tindakan ke tahap yang ada.
4. Pada Edit:<Stage>, pilih + Tambahkan grup tindakan untuk menambahkan aksi serial. Atau pilih + Tambahkan tindakan untuk menambahkan aksi paralel.
5. Pada halaman Edit tindakan:
   a. Di Nama tindakan, masukkan nama untuk tindakan Lintas wilayah.
   b. Di Penyedia tindakan, pilih penyedia tindakan.
   c. Di Wilayah, pilih AWS Wilayah tempat Anda membuat atau rencanakan untuk membuat sumber daya untuk tindakan tersebut. Ketika Wilayah dipilih, sumber daya yang tersedia untuk Wilayah tersebut dicantumkan untuk dipilih. Bidang Region menentukan tempat AWS sumber daya dibuat untuk tipe tindakan dan jenis penyedia ini. Bidang ini hanya menampilkan tindakan di mana penyedia tindakan adalah Layanan AWS. Bidang Region default sama Wilayah AWS dengan pipeline Anda.
   d. Dalam artefak Input pilih input yang sesuai dari tahap sebelumnya. Misalnya, jika tahap sebelumnya adalah tahap sumber, pilih SourceArtifact.
   e. Lengkapi semua bidang yang diperlukan untuk penyedia tindakan yang Anda konfigurasikan.
   f. Dalam artefak Output pilih output yang sesuai ke tahap berikutnya. Misalnya, jika tahap selanjutnya adalah tahap penyebaran, pilih BuildArtifact.
   g. Pilih Simpan.
6. Pada Edit:<Stage>, pilih Selesai.
7. Pilih Simpan.

Mengedit tindakan Lintas wilayah dalam tahap pipeline (konsol)

Gunakan konsol untuk mengedit tindakan Lintas wilayah yang ada dalam pipeline.

> **Note**
>
> Jika pipeline berjalan saat perubahan disimpan, eksekusi itu tidak selesai.

Untuk mengedit tindakan Lintas wilayah

1. Masuk ke konsol di https://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pilih pipeline Anda, lalu pilih Edit.
3. Pilih Edit tahap.
4. Pada Edit:<Stage>, pilih ikon untuk mengedit tindakan yang ada.
5. Pada halaman Edit tindakan, buat perubahan pada bidang, yang sesuai.
6. Pada Edit:<Stage>, pilih Selesai.
7. Pilih Simpan.

Menghapus tindakan Lintas wilayah dari tahap pipeline (konsol)

Gunakan konsol untuk menghapus tindakan Lintas wilayah yang ada dari pipeline.

> **Note**
>
> Jika pipeline berjalan saat perubahan disimpan, eksekusi itu tidak selesai.

Untuk menghapus tindakan Lintas wilayah

1. Masuk ke konsol di http://console.aws.amazon. com/codesuite/codepipeline/home.
2. Pilih pipeline Anda, lalu pilih Edit.
3. Pilih Edit tahap.
4. Pada Edit:<Stage>, pilih ikon untuk menghapus tindakan yang ada.
5. Pada Edit:<Stage>, pilih Selesai.
6. Pilih Simpan.

Tambahkan tindakan Lintas wilayah ke pipeline (CLI)

Anda dapat menggunakan AWS CLI untuk menambahkan tindakan Lintas wilayah ke pipeline yang ada.

Untuk membuat tindakan Lintas wilayah dalam tahap pipeline dengan AWS CLI, Anda menambahkan tindakan konfigurasi bersama dengan `region` bidang opsional. Anda juga harus sudah membuat ember artefak di wilayah aksi. Alih-alih menyediakan `artifactStore` parameter pipeline wilayah

tunggal, Anda menggunakan `artifactStores` parameter tersebut untuk menyertakan daftar bucket artefak masing-masing Region.

> **Note**
>
> Dalam panduan ini dan contohnya, *`RegionA`* adalah Wilayah tempat pipa dibuat. Ini memiliki akses ke bucket *`RegionA`* Amazon S3 yang digunakan untuk menyimpan artefak pipa dan peran layanan yang digunakan oleh. CodePipeline *`RegionB`*adalah wilayah tempat CodeDeploy aplikasi, grup penyebaran, dan peran layanan yang digunakan oleh CodeDeploy dibuat.

Prasyarat

Anda harus telah membuat yang berikut:

- Sebuah pipa masuk*`RegionA`*.
- Ember artefak Amazon S3 masuk. *`RegionB`*
- Sumber daya untuk tindakan Anda, seperti CodeDeploy aplikasi dan grup penerapan untuk tindakan penerapan Lintas wilayah, di. *`RegionB`*

Tambahkan tindakan Lintas wilayah ke pipeline (CLI)

Gunakan AWS CLI untuk menambahkan tindakan Lintas wilayah ke pipeline.

Untuk menambahkan tindakan Lintas wilayah

1. Untuk pipeline*`RegionA`*, jalankan get-pipeline perintah untuk menyalin struktur pipa ke file JSON. Misalnya, untuk pipeline bernama`MyFirstPipeline`, jalankan perintah berikut:

   ```
   aws codepipeline get-pipeline --name MyFirstPipeline >pipeline.json
   ```

   Perintah ini tidak mengembalikan apa pun, tetapi file yang Anda buat akan muncul di direktori tempat Anda menjalankan perintah.
2. Tambahkan `region` bidang untuk menambahkan tahap baru dengan tindakan Lintas wilayah yang menyertakan Wilayah dan sumber daya untuk tindakan Anda. Contoh JSON berikut menambahkan tahap Deploy dengan tindakan penerapan lintas wilayah di mana penyedia berada CodeDeploy, di wilayah baru. `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. Dalam struktur pipa, hapus `artifactStore` bidang dan tambahkan `artifactStores` peta untuk tindakan Lintas wilayah baru Anda. Pemetaan harus menyertakan entri untuk setiap AWS Wilayah di mana Anda memiliki tindakan. Untuk setiap entri dalam pemetaan, sumber daya harus berada di AWS Wilayah masing-masing. Dalam contoh di bawah ini, `ID-A` adalah ID kunci enkripsi untuk*RegionA*, dan `ID-B` merupakan ID kunci enkripsi untuk*RegionB*.

```
"artifactStores":{
   "RegionA":{
      "encryptionKey":{
         "id":"ID-A",
         "type":"KMS"
      },
      "location":"Location1",
      "type":"S3"
   },
   "RegionB":{
      "encryptionKey":{
```

```
          "id":"ID-B",
          "type":"KMS"
       },
       "location":"Location2",
       "type":"S3"
    }
}
```

Contoh JSON berikut menunjukkan bucket us-west-2 sebagai `my-storage-bucket` dan menambahkan bucket us-east-1 baru bernama. `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. Jika Anda bekerja dengan struktur pipa yang diambil menggunakan get-pipeline perintah, hapus `metadata` baris dari file JSON. Jika tidak, update-pipeline perintah tidak dapat menggunakannya. Hapus `"metadata": { }` garis dan`"created"`,`"pipelineARN"`, dan `"updated"` bidang.

   Misalnya, hapus baris berikut dari struktur:

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

   Simpan file tersebut.

5. Untuk menerapkan perubahan Anda, jalankan update-pipeline perintah, dengan menentukan file JSON pipeline:

> **Important**
>
> Pastikan untuk menyertakan `file://` sebelum nama file. Diperlukan dalam perintah ini.

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

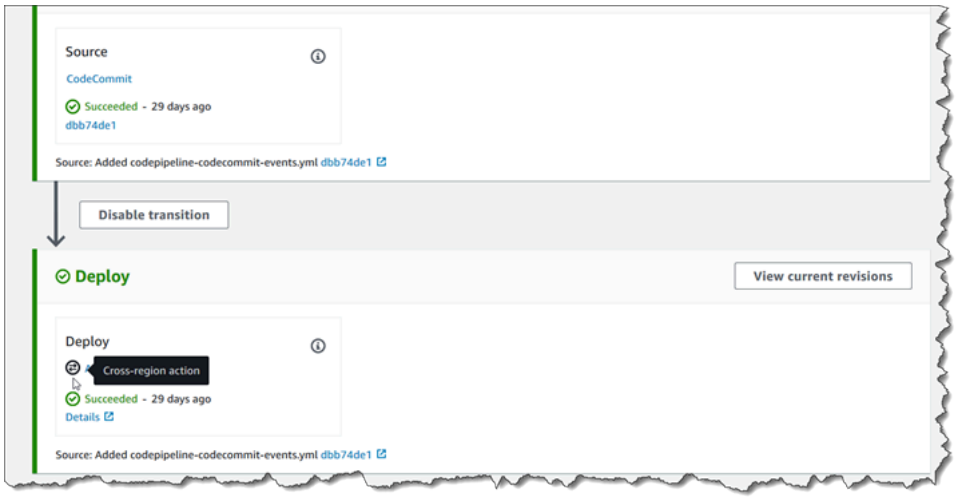
Perintah ini mengembalikan seluruh struktur pipa yang diedit. Output Anda serupa dengan yang berikut ini.

```
{
    "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**
>
> update-pipelinePerintah menghentikan pipa. Jika revisi sedang dijalankan melalui pipeline saat Anda menjalankan update-pipeline perintah, proses itu dihentikan. Anda harus memulai pipeline secara manual untuk menjalankan revisi itu melalui pipeline yang diperbarui. Gunakan **`start-pipeline-execution`** perintah untuk memulai pipeline Anda secara manual.

6. Setelah memperbarui pipeline, tindakan lintas wilayah akan ditampilkan di konsol.



Menambahkan tindakan Lintas wilayah ke pipeline ()CloudFormation

Anda dapat menggunakan CloudFormation untuk menambahkan tindakan Lintas wilayah ke pipeline yang ada.

Untuk menambahkan tindakan Lintas wilayah dengan CloudFormation

1. Tambahkan `Region` parameter ke `ActionDeclaration` sumber daya di template Anda, seperti yang ditunjukkan dalam contoh ini:

```
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. Di bawah`Mappings`, tambahkan peta wilayah seperti yang ditunjukkan dalam contoh ini untuk pemetaan bernama `SecondRegionMap` yang memetakan nilai untuk kunci `RegionA` dan`RegionB`. Di bawah `Pipeline` sumber daya, di bawah `artifactStore` bidang, tambahkan `artifactStores` peta untuk tindakan Lintas wilayah baru Anda sebagai berikut:

```
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
```

Contoh YAMG berikut menunjukkan *RegionA* bucket sebagai `us-west-2` dan menambahkan *RegionB* bucket baru,`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. Simpan template yang diperbarui ke komputer lokal Anda, lalu buka CloudFormation konsol.
4. Pilih tumpukan Anda, lalu pilih Create Change Set for Current Stack.
5. Unggah template, lalu lihat perubahan yang tercantum di dalamnya CloudFormation. Ini adalah perubahan yang harus dilakukan pada tumpukan. Anda harus melihat sumber daya baru Anda dalam daftar.
6. Pilih Eksekusi.

Bekerja dengan variabel

Beberapa tindakan dalam CodePipeline menghasilkan variabel. Untuk menggunakan variabel:

- Anda menetapkan namespace ke tindakan untuk membuat variabel yang dihasilkannya tersedia untuk konfigurasi tindakan hilir.
- Anda mengonfigurasi tindakan hilir untuk menggunakan variabel yang dihasilkan oleh tindakan.

  Anda dapat melihat detail untuk setiap eksekusi tindakan untuk melihat nilai untuk setiap variabel keluaran yang dihasilkan oleh tindakan dalam waktu eksekusi.

Untuk melihat step-by-step contoh penggunaan variabel:

- Untuk tutorial dengan tindakan Lambda yang menggunakan variabel dari tindakan upstream (CodeCommit) dan menghasilkan variabel keluaran, lihat. Tutorial: Menggunakan variabel dengan tindakan panggilan Lambda
- Untuk tutorial dengan CloudFormation tindakan yang mereferensikan variabel keluaran tumpukan dari CloudFormation tindakan upstream, lihatTutorial: Buat pipeline yang menggunakan variabel dari AWS CloudFormation tindakan penerapan.
- Untuk contoh tindakan persetujuan manual dengan teks pesan yang mereferensikan variabel keluaran yang diselesaikan ke ID CodeCommit komit dan pesan komit, lihatContoh: Gunakan variabel dalam persetujuan manual.
- Untuk contoh CodeBuild tindakan dengan variabel lingkungan yang menyelesaikan ke nama GitHub cabang, lihat. Contoh: Gunakan BranchName variabel dengan variabel CodeBuild lingkungan
- CodeBuild tindakan menghasilkan sebagai variabel semua variabel lingkungan yang diekspor sebagai bagian dari build. Untuk informasi selengkapnya, lihat CodeBuild variabel keluaran aksi. Untuk daftar variabel lingkungan yang dapat Anda gunakan CodeBuild, lihat Variabel lingkungan di lingkungan build di Panduan AWS CodeBuild Pengguna.

Topik

- Konfigurasikan tindakan untuk variabel
- Lihat variabel keluaran
- Contoh: Gunakan variabel dalam persetujuan manual
- Contoh: Gunakan BranchName variabel dengan variabel CodeBuild lingkungan

Konfigurasikan tindakan untuk variabel

Saat menambahkan tindakan ke pipeline, Anda dapat menetapkannya namespace dan mengonfigurasinya untuk menggunakan variabel dari tindakan sebelumnya.

Konfigurasikan tindakan dengan variabel (konsol)

Contoh ini membuat pipeline dengan aksi CodeCommit sumber dan aksi CodeBuild build. CodeBuild Tindakan dikonfigurasi untuk mengkonsumsi variabel yang dihasilkan oleh CodeCommit tindakan.

Jika namespace tidak ditentukan, variabel tidak tersedia untuk referensi dalam konfigurasi tindakan. Saat Anda menggunakan konsol untuk membuat pipeline, namespace untuk setiap tindakan dibuat secara otomatis.

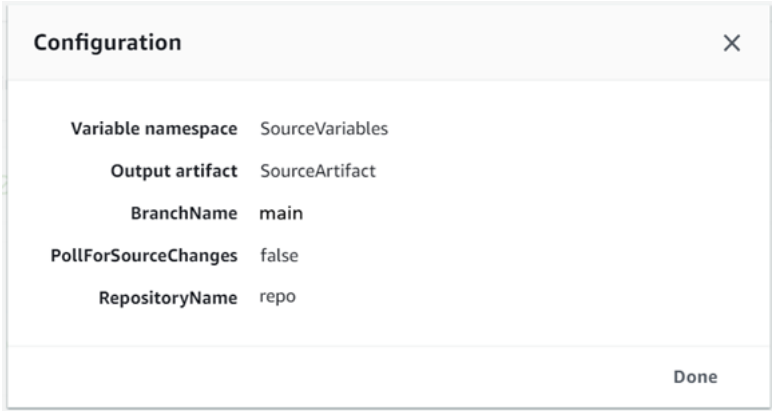
Untuk membuat pipeline dengan variabel

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pilih Buat pipeline. Masukkan nama untuk pipeline Anda, lalu pilih Berikutnya.
3. Di Sumber, di Penyedia, pilih CodeCommit. Pilih CodeCommit repositori dan cabang untuk tindakan sumber, lalu pilih Berikutnya.
4. Di Build, di Provider, pilih CodeBuild. Pilih nama proyek CodeBuild build yang ada atau pilih Buat proyek. Pada Create build project, buat project build, lalu pilih Return to CodePipeline.

   Di bawah variabel Lingkungan, pilih Tambahkan variabel lingkungan. Misalnya, masukkan ID eksekusi dengan sintaks variabel `#{codepipeline.PipelineExecutionId}` dan komit ID dengan sintaks `#{SourceVariables.CommitId}` variabel.

   > **Note**
   >
   > Anda dapat memasukkan sintaks variabel di bidang konfigurasi tindakan apa pun di wizard.

5. Pilih Buat.
6. Setelah pipeline dibuat, Anda dapat melihat namespace yang dibuat oleh wizard. Pada pipeline, pilih ikon untuk tahap yang ingin Anda lihat namespace. Dalam contoh ini, namespace yang dibuat secara otomatis dari aksi sumber`SourceVariables`, ditampilkan.



Untuk mengedit namespace untuk tindakan yang ada

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Pilih pipeline yang ingin Anda edit, lalu pilih Edit. Untuk tahap sumber, pilih Edit tahap. Tambahkan CodeCommit tindakan.
3. Pada tindakan Edit, lihat bidang namespace Variable. Jika tindakan yang ada dibuat sebelumnya atau tanpa menggunakan wizard, Anda harus menambahkan namespace. Di Namespace variabel, masukkan nama namespace, lalu pilih Simpan.

Untuk melihat variabel keluaran

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.
2. Setelah pipeline dibuat dan berjalan dengan sukses, Anda dapat melihat variabel di halaman Detail eksekusi tindakan. Untuk informasi, lihat Lihat variabel (konsol).

Konfigurasikan tindakan untuk variabel (CLI)

Bila Anda menggunakan create-pipeline perintah untuk membuat pipeline atau update-pipeline perintah untuk mengedit pipeline, Anda dapat reference/use variabel dalam konfigurasi tindakan.

Jika namespace tidak ditentukan, variabel yang dihasilkan oleh tindakan tidak tersedia untuk direferensikan dalam konfigurasi tindakan hilir apa pun.

Untuk mengkonfigurasi tindakan dengan namespace

1. Ikuti langkah-langkah Buat pipeline, tahapan, dan tindakan untuk membuat pipeline menggunakan CLI. Mulai file input untuk memberikan create-pipeline perintah dengan `--cli-input-json` parameter. Dalam struktur pipa, tambahkan `namespace` parameter dan tentukan nama, seperti`SourceVariables`.

   ```
   . . .
   {
             "inputArtifacts": [],
             "name": "Source",
             "region": "us-west-2",
             "namespace": "SourceVariables",
             "actionTypeId": {
               "category": "Source",
               "owner": "AWS",
               "version": "1",
               "provider": "CodeCommit"
             },
             "outputArtifacts": [

   . . .
   ```

2. Simpan file dengan nama seperti**`MyPipeline.json`**.
3. Di terminal (Linux, macOS, atau Unix) atau command prompt (Windows), jalankan create-pipelineperintah dan buat pipeline.

   Panggil file yang Anda buat saat menjalankan create-pipelineperintah. Contoh:

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

Untuk mengonfigurasi tindakan hilir untuk menggunakan variabel

1. Edit file input untuk memberikan update-pipeline perintah dengan `--cli-input-json` parameter. Dalam tindakan hilir, tambahkan variabel ke konfigurasi untuk tindakan itu. Variabel terdiri dari namespace dan kunci, dipisahkan oleh titik. Misalnya, untuk menambahkan variabel untuk ID eksekusi pipeline dan ID komit sumber, tentukan namespace `codepipeline` untuk variabel tersebut. `#{codepipeline.PipelineExecutionId}` Tentukan namespace `SourceVariables` untuk variabel. `#{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. Simpan file dengan nama seperti**`MyPipeline.json`**.
3. Di terminal (Linux, macOS, atau Unix) atau command prompt (Windows), jalankan create-pipelineperintah dan buat pipeline.

   Panggil file yang Anda buat saat menjalankan create-pipelineperintah. Contoh:

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

Lihat variabel keluaran

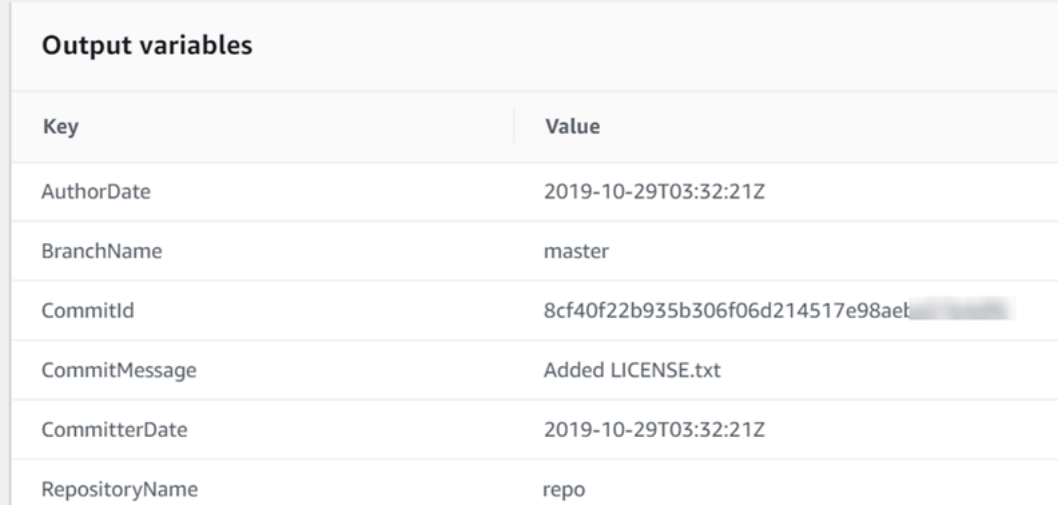
Anda dapat melihat detail eksekusi tindakan untuk melihat variabel untuk tindakan tersebut, khusus untuk setiap eksekusi.

Lihat variabel (konsol)

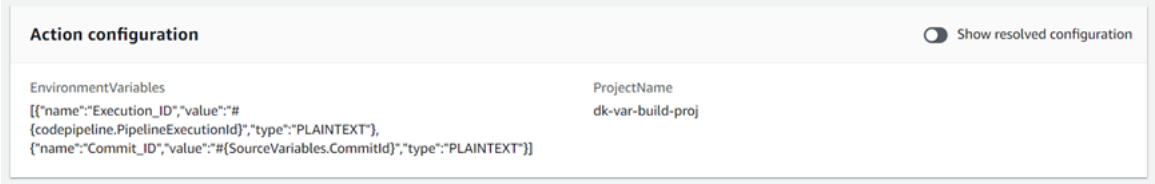
Anda dapat menggunakan konsol untuk melihat variabel untuk tindakan.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

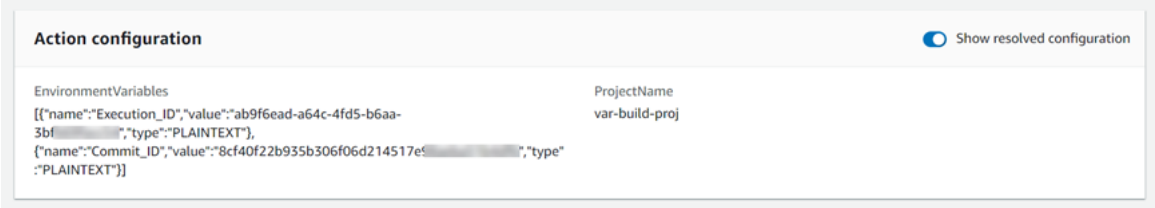
   Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipa.
3. Pilih Lihat riwayat.
4. Setelah pipeline berjalan dengan sukses, Anda dapat melihat variabel yang dihasilkan oleh aksi sumber. Pilih Lihat riwayat. Pilih Sumber dalam daftar tindakan untuk eksekusi pipeline untuk melihat detail eksekusi tindakan untuk CodeCommit tindakan tersebut. Pada layar detail tindakan, lihat variabel di bawah Variabel keluaran.



5. Setelah pipeline berjalan dengan sukses, Anda dapat melihat variabel yang digunakan oleh aksi build. Pilih Lihat riwayat. Dalam daftar tindakan untuk eksekusi pipeline, pilih Build untuk melihat detail eksekusi tindakan untuk CodeBuild tindakan tersebut. Pada halaman detail tindakan, lihat variabel di bawah konfigurasi Tindakan. Namespace yang dibuat secara otomatis ditampilkan.



Secara default, konfigurasi Action menampilkan sintaks variabel. Anda dapat memilih Tampilkan konfigurasi yang diselesaikan untuk mengaktifkan daftar untuk menampilkan nilai yang dihasilkan selama eksekusi tindakan.



Lihat variabel (CLI)

Anda dapat menggunakan list-action-executions perintah untuk melihat variabel untuk tindakan.

1. Gunakan perintah berikut ini.

```
aws codepipeline list-action-executions
```

Output menunjukkan `outputVariables` parameter seperti yang ditunjukkan di sini.

```
"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. Gunakan perintah berikut ini.

```
aws codepipeline get-pipeline --name <pipeline-name>
```

Dalam konfigurasi tindakan untuk CodeBuild tindakan, Anda dapat melihat variabel:

```
{
    "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
        }
    ]
},
```

Contoh: Gunakan variabel dalam persetujuan manual

Saat menentukan namespace untuk suatu tindakan, dan tindakan tersebut menghasilkan variabel keluaran, Anda dapat menambahkan persetujuan manual yang menampilkan variabel dalam pesan

persetujuan. Contoh ini menunjukkan cara menambahkan sintaks variabel ke pesan persetujuan manual.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan. Pilih pipeline yang ingin Anda tambahkan persetujuan.
2. Untuk mengedit pipeline Anda, pilih Edit. Tambahkan persetujuan manual setelah tindakan sumber. Di Nama tindakan, masukkan nama tindakan persetujuan.
3. Di penyedia Tindakan, pilih Persetujuan manual.
4. Di URL untuk ditinjau, tambahkan sintaks variabel `CommitId` ke CodeCommit URL Anda. Pastikan Anda menggunakan namespace yang ditetapkan untuk tindakan sumber Anda. Misalnya, sintaks variabel untuk CodeCommit tindakan dengan namespace `SourceVariables` default adalah. `#{SourceVariables.CommitId}`

   Di Komentar, masuk`CommitMessage`, masukkan pesan komit:

   ```
   Please approve this change. Commit message: #{SourceVariables.CommitMessage}
   ```
5. Setelah pipeline berhasil berjalan, Anda dapat melihat nilai variabel dalam pesan persetujuan.

Contoh: Gunakan BranchName variabel dengan variabel CodeBuild lingkungan

Saat menambahkan CodeBuild tindakan ke pipeline, Anda dapat menggunakan variabel CodeBuild lingkungan untuk mereferensikan variabel `BranchName` keluaran dari tindakan sumber hulu. Dengan variabel keluaran dari action in CodePipeline, Anda dapat membuat variabel CodeBuild lingkungan Anda sendiri untuk digunakan dalam perintah build Anda.

Contoh ini menunjukkan cara menambahkan sintaks variabel keluaran dari aksi GitHub sumber ke variabel CodeBuild lingkungan. Sintaks variabel keluaran dalam contoh ini mewakili variabel keluaran aksi GitHub sumber untuk`BranchName`. Setelah tindakan berjalan dengan sukses, variabel memutuskan untuk menunjukkan nama GitHub cabang.

1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan. Pilih pipeline yang ingin Anda tambahkan persetujuan.

2. Untuk mengedit pipeline Anda, pilih Edit. Pada panggung yang berisi CodeBuild tindakan Anda, pilih Edit tahap.
3. Pilih ikon untuk mengedit CodeBuild tindakan Anda.
4. Pada halaman Edit tindakan, di bawah variabel Lingkungan, masukkan yang berikut ini:

   - Di Nama, masukkan nama untuk variabel lingkungan Anda.
   - Di Value, masukkan sintaks variabel untuk variabel keluaran pipeline Anda, yang mencakup namespace yang ditetapkan untuk tindakan sumber Anda. Misalnya, sintaks variabel keluaran untuk GitHub tindakan dengan namespace `SourceVariables` default adalah. `#{SourceVariables.BranchName}`
   - Di Type, pilih Plaintext.
5. Setelah pipeline berjalan dengan sukses, Anda dapat melihat bagaimana variabel keluaran yang diselesaikan adalah nilai dalam variabel lingkungan. Pilih salah satu cara berikut:

   - CodePipeline konsol: Pilih pipeline Anda, lalu pilih History. Pilih eksekusi pipeline terbaru.
     - Di bawah Timeline, pilih pemilih untuk Sumber. Ini adalah tindakan sumber yang menghasilkan variabel GitHub output. Pilih Lihat detail eksekusi. Di bawah variabel Output, lihat daftar variabel keluaran yang dihasilkan oleh tindakan ini.
     - Di bawah Timeline, pilih pemilih untuk Build. Ini adalah tindakan build yang menentukan variabel CodeBuild lingkungan untuk proyek build Anda. Pilih Lihat detail eksekusi. Di bawah konfigurasi Action, lihat variabel CodeBuild lingkungan Anda. Pilih Tampilkan konfigurasi yang diselesaikan. Nilai variabel lingkungan Anda adalah variabel `BranchName` keluaran yang diselesaikan dari aksi GitHub sumber. Dalam contoh ini, nilai yang diselesaikan adalah`main`.

       Untuk informasi selengkapnya, lihat Lihat variabel (konsol).
   - CodeBuild console: Pilih project build Anda dan pilih link untuk build run Anda. Di bawah variabel Lingkungan, variabel keluaran yang diselesaikan adalah nilai untuk variabel CodeBuild lingkungan. Dalam contoh ini, variabel lingkungan Nama adalah `BranchName` dan Nilai adalah variabel `BranchName` keluaran yang diselesaikan dari tindakan GitHub sumber. Dalam contoh ini, nilai yang diselesaikan adalah`main`.

Build logs
Phase details
Reports
Environment variables
Build details
Resource utilization
Name
Value
Type
BranchName
main
PLAINTEXT

Bekerja dengan transisi panggung di CodePipeline

Transisi adalah tautan antara tahapan pipa yang dapat dinonaktifkan atau diaktifkan. Mereka diaktifkan secara default. Saat Anda mengaktifkan kembali transisi yang dinonaktifkan, revisi terbaru berjalan melalui tahapan pipa yang tersisa kecuali lebih dari 30 hari telah berlalu. Eksekusi pipeline tidak akan dilanjutkan untuk transisi yang telah dinonaktifkan lebih dari 30 hari kecuali perubahan baru terdeteksi atau Anda menjalankan ulang pipeline secara manual.

Anda dapat menggunakan AWS CodePipeline konsol atau AWS CLI untuk menonaktifkan atau mengaktifkan transisi antar tahapan dalam pipeline.

> **Note**
>
> Anda dapat menggunakan tindakan persetujuan untuk menjeda proses pipeline hingga disetujui secara manual untuk melanjutkan. Untuk informasi selengkapnya, lihat Menambahkan tindakan persetujuan manual ke panggung.

Topik

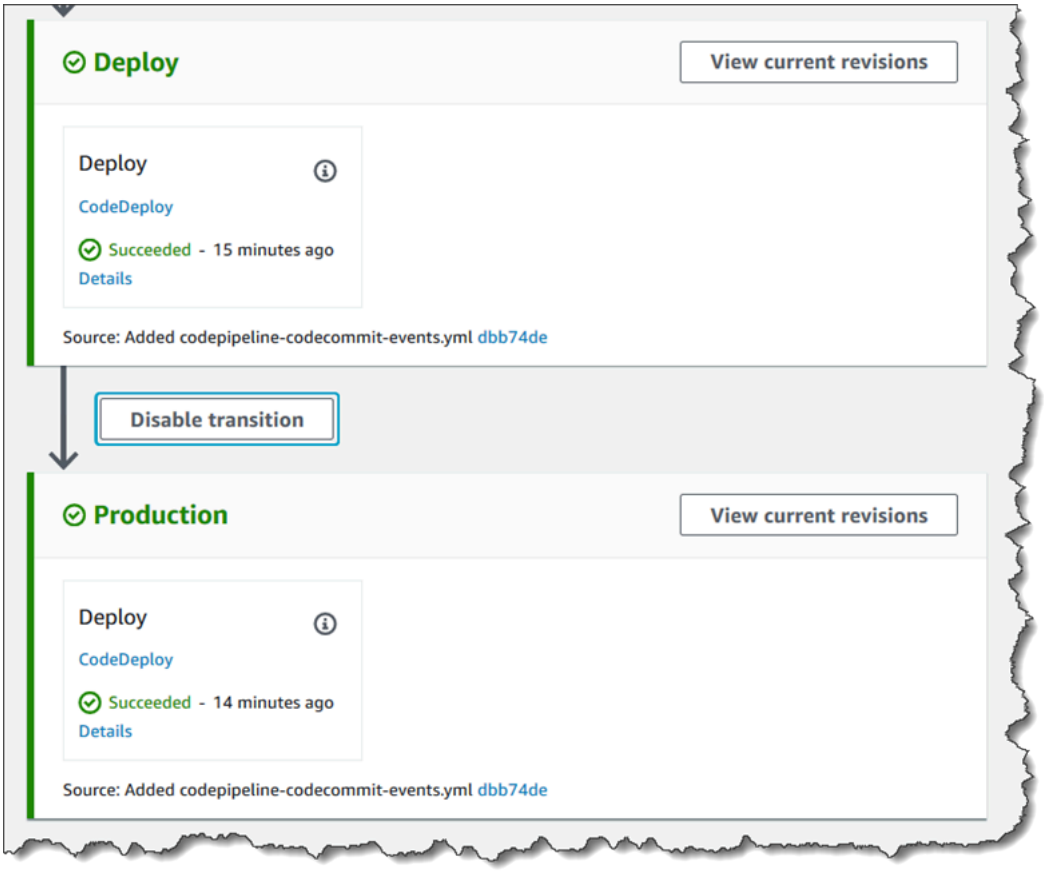
- Nonaktifkan atau aktifkan transisi (konsol)
- Nonaktifkan atau aktifkan transisi (CLI)

Nonaktifkan atau aktifkan transisi (konsol)

Untuk menonaktifkan atau mengaktifkan transisi dalam pipeline

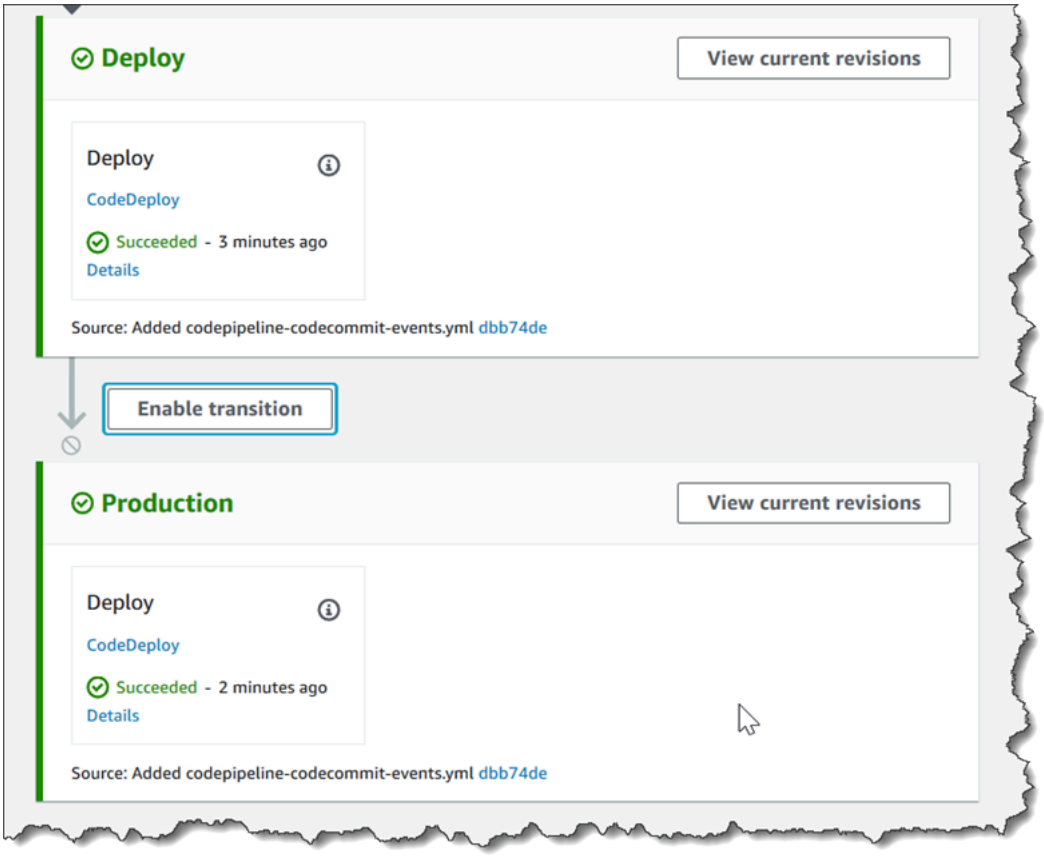
1. Masuk ke Konsol Manajemen AWS dan buka CodePipeline konsol di http://console.aws.amazon.com/codesuite/codepipeline/home.

   Nama-nama semua pipeline yang terkait dengan AWS akun Anda ditampilkan.
2. Di Nama, pilih nama pipeline yang ingin Anda aktifkan atau nonaktifkan transisi. Ini membuka tampilan rinci dari pipa, termasuk transisi antara tahapan pipa.
3. Temukan panah setelah tahap terakhir yang ingin Anda jalankan, lalu pilih tombol di sebelahnya. Misalnya, dalam pipeline berikut, jika Anda ingin tindakan dalam tahap Pementasan berjalan, tetapi bukan tindakan di tahap bernama Produksi, pilih tombol Nonaktifkan transisi di antara dua tahap tersebut:



4. Dalam kotak dialog Nonaktifkan transisi, masukkan alasan untuk menonaktifkan transisi, lalu pilih Nonaktifkan.

   Tombol berubah untuk menunjukkan bahwa transisi dinonaktifkan antara tahap sebelum panah dan tahap mengikuti panah. Setiap revisi yang sudah berjalan pada tahap-tahap yang datang setelah transisi yang dinonaktifkan berlanjut melalui pipa, tetapi setiap revisi berikutnya tidak berlanjut melewati transisi yang dinonaktifkan.



5. Pilih tombol Aktifkan transisi di sebelah panah. Dalam kotak dialog Aktifkan transisi, pilih Aktifkan. Pipa segera memungkinkan transisi antara dua tahap. Jika ada revisi yang telah dijalankan melalui tahap awal setelah transisi dinonaktifkan, dalam beberapa saat, pipeline mulai menjalankan revisi terbaru melalui tahapan setelah transisi yang sebelumnya dinonaktifkan. Pipa menjalankan revisi melalui semua tahapan yang tersisa dalam pipa.

> **Note**
>
> Mungkin perlu beberapa detik agar perubahan muncul di CodePipeline konsol setelah Anda mengaktifkan transisi.

Nonaktifkan atau aktifkan transisi (CLI)

Untuk menonaktifkan transisi antar tahapan dengan menggunakan AWS CLI, jalankan disable-stage-transition perintah. Untuk mengaktifkan transisi yang dinonaktifkan, jalankan enable-stage-transition perintah.

Untuk menonaktifkan transisi

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan disable-stage-transitionperintah, menentukan nama pipeline, nama tahap yang ingin Anda nonaktifkan transisi, jenis transisi, dan alasan Anda menonaktifkan transisi ke tahap itu. Tidak seperti menggunakan konsol, Anda juga harus menentukan apakah Anda menonaktifkan transisi ke tahap (inbound) atau transisi keluar dari panggung setelah semua tindakan selesai (outbound).

   Misalnya, untuk menonaktifkan transisi ke tahap bernama *Staging* dalam pipeline bernama*MyFirstPipeline*, Anda akan mengetik perintah yang mirip dengan berikut ini:

   ```
   aws codepipeline disable-stage-transition --pipeline-name MyFirstPipeline --stage-
   name Staging --transition-type Inbound --reason "My Reason"
   ```

   Perintah tidak mengembalikan apa pun.
2. Untuk memverifikasi transisi telah dinonaktifkan, lihat pipeline di CodePipeline konsol atau jalankan get-pipeline-state perintah. Untuk informasi selengkapnya, lihat Lihat saluran pipa (konsol) dan Lihat detail dan riwayat saluran pipa (CLI).

Untuk mengaktifkan transisi

1. Buka terminal (Linux, macOS, atau Unix) atau command prompt (Windows) dan gunakan AWS CLI untuk menjalankan enable-stage-transitionperintah, menentukan nama pipeline, nama tahap yang ingin Anda aktifkan transisi, dan jenis transisi.

   Misalnya, untuk mengaktifkan transisi ke tahap bernama *Staging* dalam pipeline bernama*MyFirstPipeline*, Anda akan mengetik perintah yang mirip dengan berikut ini:

   ```
   aws codepipeline enable-stage-transition --pipeline-name MyFirstPipeline --stage-
   name Staging  --transition-type Inbound
   ```

   Perintah tidak mengembalikan apa pun.
2. Untuk memverifikasi transisi telah dinonaktifkan, lihat pipeline di CodePipeline konsol atau jalankan get-pipeline-state perintah. Lihat informasi yang lebih lengkap di Lihat saluran pipa (konsol) dan Lihat detail dan riwayat saluran pipa (CLI).

Memantau pipeline

Pemantauan merupakan bagian penting dari menjaga keandalan, ketersediaan, dan kinerja AWS CodePipeline. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik, jika terjadi. Sebelum Anda mulai memantau, Anda harus membuat rencana pemantauan yang menjawab pertanyaan-pertanyaan berikut:

- Apa tujuan pemantauan Anda?
- Sumber daya manakah yang akan Anda pantau?
- Seberapa seringkah Anda akan memantau sumber daya ini?
- Alat pemantauan apa yang tersedia untuk Anda gunakan?
- Siapa yang akan melakukan tugas pemantauan?
- Siapa yang harus diberi tahu jika terjadi kesalahan?

Anda dapat menggunakan alat-alat berikut untuk memantau CodePipeline saluran pipa Anda dan sumber dayanya:

- EventBridge event bus event - Anda dapat memantau CodePipeline peristiwa di EventBridge, yang mendeteksi perubahan dalam pipeline, panggung, atau status eksekusi tindakan Anda. EventBridge merutekan data tersebut ke target seperti AWS Lambda dan Amazon Simple Notification Service. EventBridge acara sama dengan yang muncul di Amazon CloudWatch Events.
- Pemberitahuan untuk peristiwa pipeline di konsol Alat Pengembang — Anda dapat memantau CodePipeline peristiwa dengan notifikasi yang Anda atur di konsol, lalu membuat topik dan langganan Amazon Simple Notification Service. Untuk informasi selengkapnya, lihat Apa itu notifikasi di Panduan Pengguna Konsol Alat Pengembang.
- AWS CloudTrail— Gunakan CloudTrail untuk menangkap panggilan API yang dilakukan oleh atau atas nama CodePipeline di AWS akun Anda dan mengirimkan file log ke bucket Amazon S3. Anda dapat memilih untuk CloudWatch mempublikasikan notifikasi Amazon SNS saat file log baru dikirimkan sehingga Anda dapat mengambil tindakan cepat.
- Konsol dan CLI — Anda dapat menggunakan CodePipeline konsol dan CLI untuk melihat detail tentang status pipeline atau eksekusi pipeline tertentu.

Topik

- Memantau CodePipeline peristiwa
- Referensi bucket placeholder acara
- Pencatatan panggilan CodePipeline API dengan AWS CloudTrail
- CodePipeline CloudWatch metrik

Memantau CodePipeline peristiwa

Anda dapat memantau CodePipeline peristiwa di EventBridge, yang mengirimkan aliran data real-time dari aplikasi Anda sendiri, aplikasi software-as-a-service (SaaS), dan. Layanan AWS EventBridge merutekan data tersebut ke target seperti AWS Lambda dan Amazon Simple Notification Service. Peristiwa ini sama dengan yang muncul di Amazon CloudWatch Events, yang memberikan aliran peristiwa sistem yang mendekati waktu nyata yang menggambarkan perubahan AWS sumber daya. Untuk informasi selengkapnya, lihat Apa itu Amazon EventBridge? di Panduan EventBridge Pengguna Amazon.

> **Note**
>
> Amazon EventBridge adalah cara yang lebih disukai untuk mengelola acara Anda. Amazon CloudWatch Events dan EventBridge merupakan layanan dan API dasar yang sama, tetapi EventBridge menyediakan lebih banyak fitur. Perubahan yang Anda buat di CloudWatch Acara atau EventBridge akan muncul di setiap konsol.

Acara terdiri dari aturan. Aturan dikonfigurasi dengan memilih yang berikut:

- Pola Acara. Setiap aturan dinyatakan sebagai pola peristiwa dengan sumber dan jenis peristiwa untuk memantau, dan target acara. Untuk memantau peristiwa, Anda membuat aturan dengan layanan yang Anda pantau sebagai sumber acara, seperti CodePipeline. Misalnya, Anda dapat membuat aturan dengan pola peristiwa yang digunakan CodePipeline sebagai sumber peristiwa untuk memicu aturan ketika ada perubahan dalam status pipeline, tahapan, atau tindakan.
- Target. Aturan baru menerima layanan yang dipilih sebagai target acara. Anda mungkin ingin menyiapkan layanan target untuk mengirim pemberitahuan, menangkap informasi status, mengambil tindakan korektif, memulai peristiwa, atau mengambil tindakan lain. Ketika Anda menambahkan target Anda, Anda juga harus memberikan izin EventBridge untuk mengizinkannya memanggil layanan target yang dipilih.

Setiap jenis peristiwa perubahan status eksekusi memancarkan pemberitahuan dengan konten pesan tertentu, di mana:

- `version`Entri awal menunjukkan nomor versi untuk acara tersebut.
- `version`Entri di bawah pipeline `detail` menunjukkan nomor versi struktur pipa.
- `execution-id`Entri di bawah pipeline `detail` menunjukkan ID eksekusi untuk eksekusi pipeline yang menyebabkan perubahan status. Lihat panggilan GetPipelineExecution API di Referensi AWS CodePipeline API.
- `pipeline-execution-attempt`Entri menunjukkan jumlah upaya, atau percobaan ulang, untuk ID eksekusi tertentu.

CodePipeline melaporkan suatu peristiwa EventBridge kapan pun keadaan sumber daya dalam Akun AWS perubahan Anda. Acara dipancarkan dengan jaminan, at-least-once berdasarkan sumber daya berikut:

- Eksekusi pipa
- Eksekusi panggung
- Eksekusi aksi

Peristiwa dipancarkan EventBridge dengan pola acara dan skema yang dirinci di atas. Untuk acara yang diproses, seperti peristiwa yang Anda terima melalui notifikasi yang telah dikonfigurasi di konsol Alat Pengembang, pesan acara menyertakan bidang pola peristiwa dengan beberapa variasi. Misalnya, `detail-type` bidang dikonversi ke`detailType`. Untuk informasi selengkapnya, lihat panggilan PutEvents API di Referensi Amazon EventBridge API.

Contoh berikut menunjukkan acara untuk CodePipeline. Jika memungkinkan, setiap contoh menunjukkan skema untuk peristiwa yang dipancarkan bersama dengan skema untuk acara yang diproses.

Topik

- Jenis detail
- Peristiwa tingkat saluran pipa
- Acara tingkat panggung
- Acara tingkat aksi
- Buat Aturan yang Mengirim Pemberitahuan pada Acara Pipeline

Jenis detail

Saat mengatur acara untuk dipantau, Anda dapat memilih jenis detail untuk acara tersebut.

Anda dapat mengonfigurasi notifikasi yang akan dikirim saat status berubah untuk:

- Pipa tertentu atau semua saluran pipa Anda. Anda mengontrol ini dengan menggunakan `"detail-type":"CodePipeline Pipeline Execution State Change"`.
- Tahapan yang ditentukan atau semua tahapan Anda, dalam pipa tertentu atau semua saluran pipa Anda. Anda mengontrol ini dengan menggunakan `"detail-type":"CodePipeline Stage Execution State Change"`.
- Tindakan tertentu atau semua tindakan, dalam tahap tertentu atau semua tahapan, dalam pipa tertentu atau semua saluran pipa Anda. Anda mengontrol ini dengan menggunakan `"detail-type":"CodePipeline Action Execution State Change"`.

> **Note**
>
> Peristiwa yang dipancarkan oleh EventBridge berisi `detail-type` parameter, yang dikonversi ke `detailType` saat peristiwa diproses.

Jenis detail	Status	Deskripsi
CodePipeline Perubahan Status Eksekusi Pipeline	MEMBATALKAN	Eksekusi pipa dibatalkan karena struktur pipa diperbarui.
	FAILED	Eksekusi pipeline tidak berhasil diselesaikan.
	DILANJUTKAN	Eksekusi pipeline yang gagal telah dicoba ulang sebagai respons terhadap panggilan `RetryStageExecution` API.
	DIMULAI	Eksekusi pipeline saat ini sedang berjalan.
	DIHENTIKAN	Proses penghentian selesai, dan eksekusi pipa dihentikan.

Jenis detail	Status	Deskripsi
	BERHENTI	Eksekusi pipeline berhenti karena permintaan untuk berhenti dan menunggu atau menghentikan dan meninggalkan eksekusi pipa.
	SUCCEEDED	Eksekusi pipa selesai dengan sukses.
	DIGANTIKAN	Sementara eksekusi pipeline ini menunggu tahap berikutnya selesai, eksekusi pipeline yang lebih baru maju dan dilanjutkan melalui pipeline sebagai gantinya.
CodePipeline Perubahan Status Eksekusi Tahap	MEMBATALKAN	Panggung dibatalkan karena struktur pipa diperbarui.
	FAILED	Panggung tidak selesai dengan sukses.
	DILANJUTKAN	Tahap yang gagal telah dicoba ulang sebagai respons terhadap panggilan `RetryStageExecution` API.
	DIMULAI	Panggung sedang berjalan.
	DIHENTIKAN	Proses penghentian selesai, dan eksekusi panggung dihentikan.
	BERHENTI	Eksekusi tahap berhenti karena permintaan untuk berhenti dan menunggu atau berhenti dan meninggalkan eksekusi pipa.
	SUCCEEDED	Panggung selesai dengan sukses.
CodePipeline Perubahan Status Eksekusi Tindakan	MENELANTARKAN	Tindakan ini ditinggalkan karena permintaan untuk menghentikan dan meninggalkan eksekusi pipa.
	MEMBATALKAN	Tindakan dibatalkan karena struktur pipa diperbarui.
	FAILED	Untuk tindakan persetujuan, status GAGAL berarti tindakan ditolak oleh pengulas atau gagal karena konfigurasi tindakan yang salah.
	DIMULAI	Tindakan sedang berjalan.

Jenis detail	Status	Deskripsi
	SUCCEEDED	Tindakan itu selesai dengan sukses.

Peristiwa tingkat saluran pipa

Peristiwa tingkat pipeline dipancarkan ketika ada perubahan status untuk eksekusi pipeline.

Topik

- Acara PIPELINE STARTED
- Acara STOPLINE STOPING
- Acara PIPELINE BERHASIL
- Pipeline BERHASIL (contoh dengan tag Git)
- Acara PIPELINE GAGAL
- Pipeline FAILED (contoh dengan tag Git)

Acara PIPELINE STARTED

Ketika eksekusi pipeline dimulai, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini untuk pipa yang dinamai `"myPipeline"` di `us-east-1` Wilayah. `id`Bidang mewakili ID peristiwa, dan `account` bidang tersebut mewakili ID akun tempat pipeline dibuat.

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": {}
}
```

Acara STOPLINE STOPING

Ketika eksekusi pipeline berhenti, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini untuk pipa yang dinamai `myPipeline` di `us-west-2` Wilayah.

```
{
    "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"
    }
}
```

Acara PIPELINE BERHASIL

Ketika eksekusi pipeline berhasil, ia memancarkan peristiwa yang mengirimkan notifikasi dengan konten berikut. Contoh ini untuk pipa yang dinamai `myPipeline` di `us-east-1` Wilayah.

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 BERHASIL (contoh dengan tag Git)

Ketika eksekusi pipeline memiliki tahap yang telah dicoba ulang dan berhasil, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini adalah untuk pipeline

bernama `myPipeline` di `eu-central-1` Wilayah di mana `execution-trigger` dikonfigurasi untuk tag Git.

> **Note**
>
> `execution-trigger`Bidang akan memiliki salah satu `tag-name` atau`branch-name`, tergantung pada jenis peristiwa yang memicu 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
    }
}
```

Acara PIPELINE GAGAL

Ketika eksekusi pipeline gagal, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini untuk pipa yang dinamai `"myPipeline"` di `us-west-2` Wilayah.

Emitted event

```
{
    "version": "0",
    "id": "01234567-EXAMPLE",
    "detail-type": "CodePipeline Pipeline Execution State Change",
    "source": "aws.codepipeline",
    "account": "123456789012",
    "time": "2020-01-31T18:55:43Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "FAILED",
        "version": 4.0,
        "pipeline-execution-attempt": 1.0
    }
}
```

Processed event

```
{
    "account": "123456789012",
    "detailType": "CodePipeline Pipeline Execution State Change",
    "region": "us-west-2",
    "source": "aws.codepipeline",
    "time": "2021-06-24T00:46:16Z",
    "notificationRuleArn": "arn:aws:codestar-notifications:us-
west-2:123456789012:notificationrule/a69c62c21EXAMPLE",
    "detail": {
        "pipeline": "myPipeline",
        "execution-id": "12345678-1234-5678-abcd-12345678abcd",
        "start-time": "2023-10-26T13:49:39.208Z",
        "state": "FAILED",
```

```
        "version": 1.0,
        "pipeline-execution-attempt": 1.0
    },
    "resources": [
        "arn:aws:codepipeline:us-west-2:123456789012:myPipeline"
    ],
    "additionalAttributes": {
        "failedActionCount": 1,
        "failedActions": [
            {
                "action": "Deploy",
                "additionalInformation": "Deployment <ID> failed"
            }
        ],
        "failedStage": "Deploy"
    }
```

Pipeline FAILED (contoh dengan tag Git)

Kecuali gagal pada tahap sumber, untuk konfigurasi pipeline dengan pemicu, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini adalah untuk pipeline bernama `myPipeline` di `eu-central-1` Wilayah di mana `execution-trigger` dikonfigurasi untuk tag Git.

> **Note**
>
> `execution-trigger`Bidang akan memiliki salah satu `tag-name` atau`branch-name`, tergantung pada jenis peristiwa yang memicu 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"
    }
```

Acara tingkat panggung

Peristiwa tingkat tahap dipancarkan ketika ada perubahan status untuk eksekusi tahap.

Topik

- Acara STAGE STARTED
- Acara STAGE STOPING
- Acara STAGE STOPTED
- Tahap DILANJUTKAN setelah acara coba lagi tahap

Acara STAGE STARTED

Ketika eksekusi tahap dimulai, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini untuk pipa yang disebutkan "`myPipeline`" di `us-east-1` Wilayah, untuk panggung`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"
                }
            }
        ]
    }
}
```

Acara STAGE STOPING

Ketika eksekusi panggung berhenti, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini untuk pipa yang disebutkan `myPipeline` di `us-west-2` Wilayah, untuk panggung`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
    }
}
```

Acara STAGE STOPTED

Ketika eksekusi panggung dihentikan, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini untuk pipa yang disebutkan `myPipeline` di `us-west-2` Wilayah, untuk panggung`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
    }
}
```

Tahap DILANJUTKAN setelah acara coba lagi tahap

Ketika eksekusi panggung dilanjutkan dan memiliki tahap yang telah dicoba lagi, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut.

Ketika sebuah panggung telah dicoba lagi, `stage-last-retry-attempt-time` bidang akan ditampilkan, seperti yang ditunjukkan pada contoh. Bidang ditampilkan pada semua acara panggung jika percobaan ulang dilakukan.

> **Note**
>
> `stage-last-retry-attempt-time`Bidang akan hadir di semua acara tahap berikutnya setelah tahap telah dicoba lagi.

```
{
    "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
    }
}
```

Acara tingkat aksi

Peristiwa tingkat aksi dipancarkan ketika ada perubahan status untuk eksekusi tindakan.

Topik

- Acara Aksi DIMULAI
- Acara Aksi BERHASIL
- Acara Aksi GAGAL

- Acara Aksi ABANDONED

Acara Aksi DIMULAI

Ketika eksekusi tindakan dimulai, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini untuk pipeline yang dinamai `myPipeline` di `us-east-1` Wilayah, untuk tindakan `myAction` penerapan.

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": {}
}
```

Acara Aksi BERHASIL

Ketika eksekusi tindakan berhasil, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini untuk pipeline yang dinamai `"myPipeline"` di `us-west-2` Wilayah, untuk aksi sumber`"Source"`. Untuk jenis acara ini, ada dua `region` bidang yang berbeda. `region`Bidang acara menentukan Wilayah untuk acara pipeline. `region`Bidang di bawah `detail` bagian menentukan Wilayah untuk tindakan.

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": {}
}
```

Acara Aksi GAGAL

Ketika eksekusi tindakan gagal, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini untuk pipa yang dinamai `"myPipeline"` di `us-west-2` Wilayah, untuk tindakan`"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"
    }
}
```

Acara Aksi ABANDONED

Ketika eksekusi tindakan ditinggalkan, ia memancarkan peristiwa yang mengirimkan pemberitahuan dengan konten berikut. Contoh ini untuk pipa yang dinamai "`myPipeline`" di `us-west-2` Wilayah, untuk tindakan"`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
    }
}
```

Buat Aturan yang Mengirim Pemberitahuan pada Acara Pipeline

Aturan mengawasi peristiwa tertentu dan kemudian mengarahkannya ke AWS target yang Anda pilih. Anda dapat membuat aturan yang melakukan AWS tindakan secara otomatis ketika AWS tindakan lain terjadi, atau aturan yang melakukan AWS tindakan secara teratur pada jadwal yang ditetapkan.

Topik

- Mengirim Pemberitahuan Saat Status Pipeline Berubah (Konsol)
- Kirim Pemberitahuan Saat Status Pipeline Berubah (CLI)

Mengirim Pemberitahuan Saat Status Pipeline Berubah (Konsol)

Langkah-langkah ini menunjukkan cara menggunakan EventBridge konsol untuk membuat aturan untuk mengirim pemberitahuan perubahan CodePipeline.

Untuk membuat EventBridge aturan yang menargetkan pipeline Anda dengan sumber Amazon S3

1. Buka EventBridge konsol Amazon di https://console.aws.amazon.com/events/.
2. Di panel navigasi, pilih Aturan. Biarkan bus default dipilih atau pilih bus acara. Pilih Buat aturan.
3. Di Nama, masukkan nama untuk aturan Anda.
4. Di bawah Jenis aturan, pilih Aturan dengan pola acara. Pilih Berikutnya.
5. Di bawah Pola acara, pilih AWS layanan.
6. Dari daftar drop-down Jenis Acara, pilih tingkat perubahan status untuk notifikasi.
   - Untuk aturan yang berlaku untuk peristiwa tingkat pipeline, pilih Perubahan Status Eksekusi CodePipeline Pipeline.
   - Untuk aturan yang berlaku untuk peristiwa tingkat tahap, pilih Perubahan Status Eksekusi CodePipeline Tahap.
   - Untuk aturan yang berlaku untuk peristiwa tingkat tindakan, pilih Perubahan Status Eksekusi CodePipeline Tindakan.
7. Tentukan perubahan status yang berlaku untuk aturan:
   - Untuk aturan yang berlaku untuk semua perubahan status, pilih Status apa pun.
   - Untuk aturan yang berlaku untuk beberapa perubahan status saja, pilih Status tertentu, lalu pilih satu atau beberapa nilai status dari daftar.
8. Untuk pola acara yang lebih rinci daripada yang diizinkan oleh penyeleksi, Anda juga dapat menggunakan opsi Edit pola di jendela Pola acara untuk menunjuk pola acara dalam format JSON.

   > **Note**
   >
   > Jika tidak ditentukan lain, maka pola acara dibuat untuk semua pipelines/stages/actions dan status.

   Untuk pola acara yang lebih rinci, Anda dapat menyalin dan menempelkan contoh pola peristiwa berikut ke dalam jendela pola Peristiwa.

- Example

  Gunakan contoh pola peristiwa ini untuk menangkap tindakan penerapan dan pembuatan yang gagal di semua pipeline.

```
{
"source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Deploy", "Build"]
    }
  }
}
```

- Example

  Gunakan contoh pola peristiwa ini untuk menangkap semua tindakan persetujuan yang ditolak atau gagal di semua pipeline.

```
{
 "source": [
    "aws.codepipeline"
  ],
  "detail-type": [
    "CodePipeline Action Execution State Change"
  ],
  "detail": {
    "state": [
      "FAILED"
    ],
    "type": {
      "category": ["Approval"]
    }
  }
```

```
}
```

- Example

  Gunakan contoh pola peristiwa ini untuk menangkap semua peristiwa dari pipeline yang ditentukan.

```
{
"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. Pilih Berikutnya.
10. Di jenis Target, pilih AWS layanan.
11. Di Pilih target, pilih CodePipeline. Di ARN Pipeline, masukkan ARN pipa agar pipa dimulai dengan aturan ini.

> **Note**
>
> Untuk mendapatkan ARN pipeline, jalankan perintah. get-pipeline ARN pipa muncul di output. Itu dibangun dalam format ini:
> arn:aws:codepipeline::: *region account pipeline-name*
> Contoh pipa ARN:
> arn:aws:codepipeline:us-east- 2:80398 CONTOH: MyFirstPipeline

12. Untuk membuat atau menentukan peran layanan IAM yang memberikan EventBridge izin untuk memanggil target yang terkait dengan EventBridge aturan Anda (dalam hal ini, targetnya adalah): CodePipeline

    - Pilih Buat peran baru untuk sumber daya khusus ini untuk membuat peran layanan yang memberikan EventBridge izin untuk memulai eksekusi pipeline Anda.

- Pilih Gunakan peran yang ada untuk memasukkan peran layanan yang memberikan EventBridge izin untuk memulai eksekusi pipeline Anda.

13. Pilih Berikutnya.
14. Pada halaman Tag, pilih Berikutnya.
15. Pada halaman Tinjau dan buat, tinjau konfigurasi aturan. Jika Anda puas dengan aturan, pilih Create rule (Buat aturan).

Kirim Pemberitahuan Saat Status Pipeline Berubah (CLI)

Langkah-langkah ini menunjukkan cara menggunakan CLI untuk membuat aturan CloudWatch Acara untuk mengirim pemberitahuan perubahan. CodePipeline

Untuk menggunakan aturan AWS CLI untuk membuat aturan, panggil put-rule perintah, dengan menentukan:

- Nama yang secara unik mengidentifikasi aturan yang Anda buat. Nama ini harus unik di semua pipeline yang Anda buat CodePipeline terkait dengan AWS akun Anda.
- Pola acara untuk bidang sumber dan detail yang digunakan oleh aturan. Untuk informasi selengkapnya, lihat Amazon EventBridge dan Pola Acara.

Untuk membuat EventBridge aturan dengan CodePipeline sebagai sumber acara

1. Panggil put-rule perintah untuk membuat aturan yang menentukan pola acara. (Lihat tabel sebelumnya untuk status valid.)

   Perintah contoh berikut digunakan --event-pattern untuk membuat aturan yang disebut “`MyPipelineStateChanges`” yang memancarkan CloudWatch peristiwa ketika eksekusi pipeline gagal untuk pipeline bernama “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. Panggil put-targets perintah dan sertakan parameter berikut:

   - `--rule`Parameter digunakan dengan yang `rule_name` Anda buat dengan menggunakanput-rule.

- `--targets`Parameter digunakan dengan `Id` daftar target dalam daftar target dan topik ARN Amazon SNS.

Contoh perintah berikut menentukan bahwa untuk aturan yang dipanggil`MyPipelineStateChanges`, target `Id` terdiri dari nomor satu, menunjukkan bahwa dalam daftar target untuk aturan, ini adalah target 1. Perintah sampel juga menentukan contoh ARN untuk topik Amazon SNS.

```
aws events put-targets --rule MyPipelineStateChanges --targets
 Id=1,Arn=arn:aws:sns:us-west-2:11111EXAMPLE:MyNotificationTopic
```

3. Tambahkan izin EventBridge untuk menggunakan layanan target yang ditunjuk untuk memanggil notifikasi. Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis sumber daya untuk Amazon. EventBridge

Referensi bucket placeholder acara

Bagian ini hanya referensi. Untuk informasi tentang membuat pipeline dengan sumber daya deteksi peristiwa, lihatTindakan sumber dan metode deteksi perubahan.

Tindakan sumber yang disediakan oleh Amazon S3 dan CodeCommit menggunakan sumber daya deteksi perubahan berbasis peristiwa untuk memicu pipeline Anda saat perubahan dilakukan di bucket sumber atau repositori. Sumber daya ini adalah aturan CloudWatch Peristiwa yang dikonfigurasi untuk merespons peristiwa di sumber pipeline, seperti perubahan kode ke CodeCommit repositori. Saat Anda menggunakan CloudWatch Acara untuk sumber Amazon S3, Anda harus mengaktifkannya CloudTrail agar peristiwa dicatat. CloudTrail membutuhkan ember S3 di mana ia dapat mengirim intisari. Anda dapat mengakses file log untuk sumber daya CloudWatch Acara dari bucket khusus, tetapi Anda tidak dapat mengakses data dari bucket placeholder.

- Jika Anda menggunakan CLI atau CloudFormation untuk menyiapkan sumber daya CloudWatch Acara, Anda dapat menemukan CloudTrail file di bucket yang Anda tentukan saat menyiapkan pipeline.
- Jika Anda menggunakan konsol untuk menyiapkan pipeline dengan sumber S3, konsol akan menggunakan bucket CloudTrail placeholder saat membuat resource CloudWatch Acara untuk Anda. CloudTrail intisari disimpan di ember placeholder di Wilayah AWS tempat pipa dibuat.

Anda dapat mengubah konfigurasi jika ingin menggunakan bucket selain bucket placeholder.

> **Note**
>
> Data yang ditulis ke bucket CloudTrail placeholder secara otomatis kedaluwarsa setelah satu hari dan tidak disimpan.

Untuk informasi selengkapnya tentang menemukan dan mengelola file CloudTrail log Anda, lihat Mendapatkan dan Melihat File CloudTrail Log Anda.

Topik

- Nama bucket placeholder acara menurut Wilayah

Nama bucket placeholder acara menurut Wilayah

Tabel ini mencantumkan nama bucket placeholder S3 yang berisi file log yang melacak peristiwa deteksi perubahan untuk pipeline dengan tindakan sumber Amazon S3.

Nama Wilayah	Nama bucket placeholder	Pengenal wilayah
AS Timur (Ohio)	codepipeline-cloudtrail-placeholder-bucket-kami-timur-2	us-east-2
AS Timur (Virginia Utara)	codepipeline-cloudtrail-placeholder-bucket-kami-timur-1	us-east-1
AS Barat (California Utara)	codepipeline-cloudtrail-placeholder-bucket-kami-barat-1	us-west-1
AS Barat (Oregon)	codepipeline-cloudtrail-placeholder-bucket-kami-barat-2	us-west-2
Kanada (Pusat)	codepipeline-cloudtrail-placeholder-bucket-ca-pusat-1	ca-central-1
Eropa (Frankfurt)	codepipeline-cloudtrail-placeholder-bucket-eu-sentral-1	eu-central-1

Nama Wilayah	Nama bucket placeholder	Pengenal wilayah
Eropa (Irlandia)	codepipeline-cloudtrail-placeholder-bucket-eu-barat-1	eu-west-1
Eropa (London)	codepipeline-cloudtrail-placeholder-bucket-eu-barat-2	eu-west-2
Eropa (Paris)	codepipeline-cloudtrail-placeholder-bucket-eu-barat-3	eu-west-3
Eropa (Stockholm)	codepipeline-cloudtrail-placeholder-bucket-eu-utara-1	eu-north-1
Asia Pasifik (Hong Kong)	codepipeline-cloudtrail-placeholder-bucket-ap-timur-1	ap-east-1
Asia Pasifik (Hyderabad)	codepipeline-cloudtrail-placeholder-bucket-ap-selatan-2	ap-south-2
Asia Pasifik (Jakarta)	codepipeline-cloudtrail-placeholder-bucket-ap-tenggara 3	ap-southeast-3
Asia Pasifik (Melbourne)	codepipeline-cloudtrail-placeholder-bucket-ap-tenggara 4	ap-southeast-4
Asia Pasifik (Mumbai)	codepipeline-cloudtrail-placeholder-bucket-ap-selatan-1	ap-south-1
Asia Pasifik (Osaka)	codepipeline-cloudtrail-placeholder-bucket-ap-timur laut-3-prod	ap-northeast-3
Asia Pasifik (Tokyo)	codepipeline-cloudtrail-placeholder-bucket-ap-timur laut-1	ap-northeast-1

Nama Wilayah	Nama bucket placeholder	Pengenal wilayah
Asia Pasifik (Seoul)	codepipeline-cloudtrail-placeholder-bucket-ap-timur laut-2	ap-northeast-2
Asia Pasifik (Singapura)	codepipeline-cloudtrail-placeholder-bucket-ap-tenggara	ap-southeast-1
Asia Pasifik (Sydney)	codepipeline-cloudtrail-placeholder-bucket-ap-tenggara 2	ap-southeast-2
Asia Pasifik (Tokyo)	codepipeline-cloudtrail-placeholder-bucket-ap-timur laut-1	ap-northeast-1
Kanada (Pusat)	codepipeline-cloudtrail-placeholder-bucket-ca-pusat-1	ca-central-1
Eropa (Frankfurt)	codepipeline-cloudtrail-placeholder-bucket-eu-sentral-1	eu-central-1
Eropa (Irlandia)	codepipeline-cloudtrail-placeholder-bucket-eu-barat-1	eu-west-1
Eropa (London)	codepipeline-cloudtrail-placeholder-bucket-eu-barat-2	eu-west-2
Eropa (Milan)	codepipeline-cloudtrail-placeholder-bucket-eu-selatan-1	eu-south-1
Eropa (Paris)	codepipeline-cloudtrail-placeholder-bucket-eu-barat-3	eu-west-3
Eropa (Spanyol)	codepipeline-cloudtrail-placeholder-bucket-eu-selatan-2	eu-south-2
Eropa (Stockholm)	codepipeline-cloudtrail-placeholder-bucket-eu-utara-1	eu-north-1

Nama Wilayah	Nama bucket placeholder	Pengenal wilayah
Eropa (Zürich) *	codepipeline-cloudtrail-placeholder-bucket-eu-sentral-2	eu-central-2
Israel (Tel Aviv)	codepipeline-cloudtrail-placeholder-bucket-il-pusat-1	il-central-1
Timur Tengah (Bahrain) *	codepipeline-cloudtrail-placeholder-bucket-saya-selatan-1	me-south-1
Timur Tengah (UEA)	codepipeline-cloudtrail-placeholder-bucket-saya-pusat-1	me-central-1
Amerika Selatan (Sao Paulo)	codepipeline-cloudtrail-placeholder-bucket-sa-timur-1	sa-east-1

Pencatatan panggilan CodePipeline API dengan AWS CloudTrail

AWS CodePipeline terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau Layanan AWS dalam CodePipeline; CloudTrail menangkap semua panggilan API untuk CodePipeline sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari CodePipeline konsol dan panggilan kode ke operasi CodePipeline API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk. CodePipeline Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat CodePipeline, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

CodePipeline informasi di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas terjadi CodePipeline, aktivitas tersebut dicatat dalam suatu CloudTrail peristiwa bersama dengan Layanan AWS peristiwa lain dalam sejarah Peristiwa. Anda dapat melihat, mencari, dan mengunduh acara

terbaru di AWS akun Anda. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan acara yang sedang berlangsung di Anda Akun AWS , termasuk acara untuk CodePipeline, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua AWS Wilayah. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi lainnya Layanan AWS untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran Umum untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah dan Menerima File CloudTrail Log dari Beberapa Akun

Semua CodePipeline tindakan dicatat oleh CloudTrail dan didokumentasikan dalam Referensi CodePipeline API. Misalnya, panggilan ke`CreatePipeline`, `GetPipelineExecution` dan `UpdatePipeline` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan dibuat dengan kredensi root atau AWS Identity and Access Management (IAM).
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan tersebut dibuat oleh Layanan AWS lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail .

Memahami entri file CodePipeline log

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta,

tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log untuk peristiwa pipeline pembaruan, di mana pipeline bernama MyFirstPipeline telah diedit oleh pengguna bernama JaneDoe - CodePipeline dengan ID akun 80398EXAMPLE. Pengguna mengubah nama tahap sumber pipa dari `Source` ke`MySourceStage`. Karena elemen `requestParameters` dan `responseElements` elemen dalam CloudTrail log berisi seluruh struktur pipa yang diedit, elemen-elemen tersebut telah disingkat dalam contoh berikut. Penekanan telah ditambahkan ke `requestParameters` bagian pipa di mana perubahan terjadi, nomor versi sebelumnya dari pipa, dan `responseElements` bagian, yang menunjukkan nomor versi bertambah 1. Bagian yang diedit ditandai dengan elips (...) untuk mengilustrasikan di mana lebih banyak data muncul dalam entri log nyata.

```
{
 "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 metrik

Anda dapat menggunakan metrik CloudWatch untuk mengukur CodePipeline saluran pipa. Anda dapat menggunakan metrik berikut untuk mengukur durasi pipeline dan eksekusi pipeline yang gagal.

Anda dapat memantau jaringan pipa Anda di dua tingkat:

Tingkat pipa

Metrik ini berada di tingkat pipa. Pilih Dengan Pipeline di CloudWatch. Pipa yang tersedia akan dicantumkan di CloudWatch.

AWS tingkat akun

Metrik ini untuk semua saluran pipa di akun. Untuk melihat metrik di tingkat AWS akun, pilih Metrik Akun di. CloudWatch

Untuk informasi selengkapnya tentang melihat metrik, lihat Melihat metrik yang tersedia di CloudWatch Panduan Pengguna Amazon.

Topik

- PipelineDuration
- FailedPipelineExecutions

PipelineDuration

`PipelineDuration`Metrik mengukur durasi eksekusi pipa. Metrik ini hanya tersedia untuk tingkat pipa.

Unit: detik

FailedPipelineExecutions

`FailedPipelineExecutions`Metrik mengukur jumlah eksekusi pipa yang gagal. Metrik ini tersedia di tingkat pipa dan tingkat akun.

Metrik ini mencakup hitungan terpisah untuk eksekusi pipeline dengan percobaan ulang tindakan yang gagal. Dengan kata lain, ketika tindakan gagal dicoba lagi dalam pipeline, ini akan dihitung sebagai metrik eksekusi pipeline terpisah. Misalnya, untuk pipeline dengan aksi sumber S3 di mana aksi sumber S3 gagal pertama kali, yang membuat satu eksekusi, dan kemudian tindakan sumber dicoba ulang dan gagal lagi. Untuk contoh ini, metriknya adalah sebagai berikut:

```
FailedPipelineExecutionAttempts: 2
```

Unit: Hitungan

Pemecahan masalah CodePipeline

Informasi berikut dapat membantu Anda memecahkan masalah umum di AWS CodePipeline.

Topik

- Kesalahan saluran pipa: Pipeline yang dikonfigurasi dengan AWS Elastic Beanstalk menampilkan pesan kesalahan: “Penerapan gagal. Peran yang diberikan tidak memiliki izin yang memadai: Layanan: AmazonElasticLoadBalancing
- Kesalahan penerapan: Saluran pipa yang dikonfigurasi dengan tindakan AWS Elastic Beanstalk penerapan hang alih-alih gagal jika izin "" DescribeEvents tidak ada
- Kesalahan saluran pipa: Tindakan sumber mengembalikan pesan izin yang tidak memadai: “Tidak dapat mengakses CodeCommit repository-name repositori. Pastikan bahwa peran IAM pipeline memiliki izin yang cukup untuk mengakses repositori.”
- Kesalahan saluran pipa: Tindakan build atau pengujian Jenkins berjalan untuk waktu yang lama dan kemudian gagal karena kurangnya kredensyal atau izin
- Kesalahan pipa: Pipeline yang dibuat di satu AWS Wilayah menggunakan bucket yang dibuat di AWS Wilayah lain mengembalikan InternalError "" dengan kode "JobFailed”
- Kesalahan penerapan: File ZIP yang berisi file WAR berhasil digunakan AWS Elastic Beanstalk, tetapi URL aplikasi melaporkan kesalahan 404 tidak ditemukan
- Nama folder artefak pipa tampaknya terpotong
- Tambahkan CodeBuild GitClone izin untuk koneksi ke Bitbucket,, GitHub Enterprise Server GitHub, atau .com GitLab
- Tambahkan CodeBuild GitClone izin untuk tindakan CodeCommit sumber
- <source artifact name>Kesalahan saluran pipa: Penerapan dengan tindakan CodeDeployTo ECS mengembalikan pesan kesalahan: “Pengecualian saat mencoba membaca file artefak definisi tugas dari:”
- GitHub (melalui OAuth aplikasi) tindakan sumber: Daftar repositori menunjukkan repositori yang berbeda
- GitHub (melalui GitHub Aplikasi) tindakan sumber: Tidak dapat menyelesaikan koneksi untuk repositori
- Kesalahan Amazon S3: peran CodePipeline layanan <ARN>mendapatkan akses S3 ditolak untuk bucket S3 < > BucketName

- Saluran pipa dengan Amazon S3, Amazon ECR, CodeCommit atau sumber tidak lagi dimulai secara otomatis
- Kesalahan koneksi saat menghubungkan ke GitHub: “Masalah terjadi, pastikan cookie diaktifkan di browser Anda” atau “Pemilik organisasi harus menginstal GitHub aplikasi”
- Pipelines dengan mode eksekusi diubah menjadi mode QUEUED atau PARALLEL gagal saat batas run tercapai
- Pipelines dalam mode PARALLEL memiliki definisi pipeline yang sudah ketinggalan zaman jika diedit saat mengubah ke mode QUEUED atau SUPERSEDED
- Pipelines diubah dari mode PARALLEL akan menampilkan mode eksekusi sebelumnya
- Saluran pipa dengan koneksi yang menggunakan pemfilteran pemicu berdasarkan jalur file mungkin tidak dimulai pada pembuatan cabang
- Saluran pipa dengan koneksi yang menggunakan pemfilteran pemicu berdasarkan jalur file mungkin tidak dimulai saat batas file tercapai
- CodeCommit atau revisi sumber S3 dalam mode PARALLEL mungkin tidak cocok dengan acara EventBridge
- Tindakan Penerapan EC2 gagal dengan pesan kesalahan No such file
- Tindakan Penerapan EKS gagal dengan pesan cluster unreachable kesalahan
- Butuh bantuan dengan masalah yang berbeda?

Kesalahan saluran pipa: Pipeline yang dikonfigurasi dengan AWS Elastic Beanstalk menampilkan pesan kesalahan: “Penerapan gagal. Peran yang diberikan tidak memiliki izin yang memadai: Layanan: AmazonElasticLoadBalancing

Masalah: Peran layanan untuk CodePipeline tidak memiliki izin yang cukup untuk AWS Elastic Beanstalk, termasuk, namun tidak terbatas pada, beberapa operasi di Elastic Load Balancing. Peran layanan untuk CodePipeline diperbarui pada 6 Agustus 2015 untuk mengatasi masalah ini. Pelanggan yang membuat peran layanan mereka sebelum tanggal ini harus mengubah pernyataan kebijakan untuk peran layanan mereka untuk menambahkan izin yang diperlukan.

Kemungkinan perbaikan: Solusi termudah adalah mengedit pernyataan kebijakan untuk peran layanan Anda seperti yang dijelaskan dalamMenambahkan izin ke peran CodePipeline layanan.

Setelah Anda menerapkan kebijakan yang telah diedit, ikuti langkah-langkah Mulai pipa secara manual untuk menjalankan ulang saluran pipa yang menggunakan Elastic Beanstalk secara manual.

Bergantung pada kebutuhan keamanan Anda, Anda juga dapat mengubah izin dengan cara lain.

Kesalahan penerapan: Saluran pipa yang dikonfigurasi dengan tindakan AWS Elastic Beanstalk penerapan hang alih-alih gagal jika izin "" DescribeEvents tidak ada

Masalah: Peran layanan untuk CodePipeline harus mencakup `"elasticbeanstalk:DescribeEvents"` tindakan untuk setiap jaringan pipa yang digunakan AWS Elastic Beanstalk. Tanpa izin ini, tindakan AWS Elastic Beanstalk penerapan hang tanpa gagal atau menunjukkan kesalahan. Jika tindakan ini hilang dari peran layanan Anda, maka CodePipeline tidak memiliki izin untuk menjalankan tahap penerapan pipeline AWS Elastic Beanstalk atas nama Anda.

Kemungkinan perbaikan: Tinjau peran CodePipeline layanan Anda. Jika `"elasticbeanstalk:DescribeEvents"` tindakan tidak ada, gunakan langkah-langkah Menambahkan izin ke peran CodePipeline layanan untuk menambahkannya menggunakan fitur Edit Policy di konsol IAM.

Setelah Anda menerapkan kebijakan yang telah diedit, ikuti langkah-langkah Mulai pipa secara manual untuk menjalankan ulang saluran pipa yang menggunakan Elastic Beanstalk secara manual.

Kesalahan saluran pipa: Tindakan sumber mengembalikan pesan izin yang tidak memadai: “Tidak dapat mengakses CodeCommit **`repository-name`** repositori. Pastikan bahwa peran IAM pipeline memiliki izin yang cukup untuk mengakses repositori.”

Masalah: Peran layanan untuk CodePipeline tidak memiliki izin yang cukup untuk CodeCommit dan kemungkinan dibuat sebelum dukungan untuk menggunakan CodeCommit repositori ditambahkan pada 18 April 2016. Pelanggan yang membuat peran layanan mereka sebelum tanggal ini harus mengubah pernyataan kebijakan untuk peran layanan mereka untuk menambahkan izin yang diperlukan.

Kemungkinan perbaikan: Tambahkan izin yang diperlukan CodeCommit untuk kebijakan peran CodePipeline layanan Anda. Untuk informasi selengkapnya, lihat Menambahkan izin ke peran CodePipeline layanan.

Kesalahan saluran pipa: Tindakan build atau pengujian Jenkins berjalan untuk waktu yang lama dan kemudian gagal karena kurangnya kredensyal atau izin

Masalah: Jika server Jenkins diinstal pada instans Amazon EC2, instance mungkin tidak dibuat dengan peran instans yang memiliki izin yang diperlukan. CodePipeline Jika Anda menggunakan pengguna IAM di server Jenkins, instans lokal, atau instans Amazon EC2 yang dibuat tanpa peran IAM yang diperlukan, pengguna IAM tidak memiliki izin yang diperlukan, atau server Jenkins tidak dapat mengakses kredensialnya melalui profil yang dikonfigurasi di server.

Kemungkinan perbaikan: Pastikan peran instans Amazon EC2 atau pengguna IAM dikonfigurasi dengan kebijakan terkelola atau dengan izin `AWSCodePipelineCustomActionAccess` yang setara. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk AWS CodePipeline.

Jika Anda menggunakan pengguna IAM, pastikan AWS profil yang dikonfigurasi pada instance menggunakan pengguna IAM yang dikonfigurasi dengan izin yang benar. Anda mungkin harus memberikan kredensyal pengguna IAM yang Anda konfigurasikan untuk integrasi antara Jenkins dan CodePipeline langsung ke UI Jenkins. Ini bukan praktik terbaik yang direkomendasikan. Jika Anda harus melakukannya, pastikan server Jenkins diamankan dan menggunakan HTTPS, bukan HTTP.

Kesalahan pipa: Pipeline yang dibuat di satu AWS Wilayah menggunakan bucket yang dibuat di AWS Wilayah lain mengembalikan InternalError "" dengan kode "JobFailed”

Masalah: Pengunduhan artefak yang disimpan di bucket Amazon S3 akan gagal jika pipeline dan bucket dibuat di AWS Wilayah yang berbeda.

Kemungkinan perbaikan: Pastikan bucket Amazon S3 tempat artefak Anda disimpan berada di Wilayah AWS yang sama dengan pipeline yang telah Anda buat.

Kesalahan penerapan: File ZIP yang berisi file WAR berhasil digunakan AWS Elastic Beanstalk, tetapi URL aplikasi melaporkan kesalahan 404 tidak ditemukan

Masalah: File WAR berhasil disebarkan ke AWS Elastic Beanstalk lingkungan, tetapi URL aplikasi mengembalikan kesalahan 404 Tidak Ditemukan.

Kemungkinan perbaikan: AWS Elastic Beanstalk dapat membongkar file ZIP, tetapi bukan file WAR yang terkandung dalam file ZIP. Alih-alih menentukan file WAR di `buildspec.yml` file Anda, tentukan folder yang berisi konten yang akan digunakan. Contoh:

```
version: 0.2

phases:
  post_build:
    commands:
      - mvn package
      - mv target/my-web-app ./
artifacts:
  files:
    - my-web-app/**/*
 discard-paths: yes
```

Sebagai contoh, lihat AWS Elastic Beanstalk Contoh untuk CodeBuild.

Nama folder artefak pipa tampaknya terpotong

Masalah: Saat Anda melihat nama artefak pipeline CodePipeline, nama tampaknya terpotong. Ini dapat membuat nama tampak serupa atau tampaknya tidak lagi berisi seluruh nama pipeline.

Penjelasan: CodePipeline memotong nama artefak untuk memastikan bahwa jalur Amazon S3 lengkap tidak melebihi batas ukuran kebijakan saat CodePipeline menghasilkan kredensi sementara untuk pekerja kerja.

Meskipun nama artefak tampaknya terpotong, CodePipeline peta ke ember artefak dengan cara yang tidak terpengaruh oleh artefak dengan nama terpotong. Pipa dapat berfungsi secara normal. Ini bukan masalah dengan folder atau artefak. Ada batas 100 karakter untuk nama pipeline. Meskipun nama folder artefak mungkin tampak dipersingkat, itu masih unik untuk pipeline Anda.

Tambahkan CodeBuild GitClone izin untuk koneksi ke Bitbucket,, GitHub Enterprise Server GitHub, atau .com GitLab

Saat Anda menggunakan aksi AWS CodeConnections dalam sumber dan CodeBuild tindakan, ada dua cara artefak input dapat diteruskan ke build:

- Default: Tindakan sumber menghasilkan file zip yang berisi kode yang CodeBuild diunduh.
- Klon lengkap: Kode sumber dapat langsung diunduh ke lingkungan build.

  Mode klon Penuh memungkinkan Anda untuk berinteraksi dengan kode sumber sebagai repositori Git yang berfungsi. Untuk menggunakan mode ini, Anda harus memberikan izin CodeBuild lingkungan Anda untuk menggunakan koneksi.

Untuk menambahkan izin ke kebijakan peran CodeBuild layanan, Anda membuat kebijakan yang dikelola pelanggan yang dilampirkan ke peran layanan Anda CodeBuild . Langkah-langkah berikut membuat kebijakan di mana `UseConnection` izin ditentukan di `action` bidang, ARN koneksi ditentukan di `Resource` bidang, dan ID repositori sumber dibatasi melalui. `Condition`

Untuk menggunakan konsol untuk menambahkan UseConnection izin

1. Untuk menemukan ARN koneksi dan ID repositori sumber untuk pipeline Anda, buka pipeline Anda, klik ikon (i) pada tindakan sumber Anda dan beralih ke tab Input.

   Contoh koneksi ARN adalah:

   ```
   arn:aws:codeconnections:eu-central-1:123456789123:connection/
   sample-1908-4932-9ecc-2ddacee15095
   ```

   Contoh ID repositori sumber:

   ```
   owner/test-app
   ```

   Anda menambahkan ARN koneksi dan ID repositori ke kebijakan peran layanan Anda CodeBuild .
2. Untuk menemukan peran CodeBuild layanan Anda, pilih proyek build yang digunakan dalam pipeline Anda dan navigasikan ke tab Detail build.

3. Pilih tautan Peran layanan. Ini membuka konsol IAM tempat Anda dapat menambahkan kebijakan baru yang memberikan akses ke koneksi Anda.
4. Di konsol IAM, pilih Tambahkan izin, lalu pilih Buat kebijakan sebaris.

   Gunakan templat kebijakan sampel berikut. Tambahkan ARN koneksi Anda di `Resource` bidang dan ID repositori Anda di `Condition` bidang, seperti `codeconnections:FullRepositoryId` yang ditunjukkan dalam contoh ini:

   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"
                   }
               }
           }
       ]
   }
   ```

   Gunakan `Condition` bidang untuk mencatat izin kebijakan Anda lebih lanjut berdasarkan persyaratan spesifikasi build Anda (lihat dokumentasi `CodeConnection` kondisi).

   Pada tab JSON, tempel kebijakan Anda.
5. Pilih Berikutnya. Masukkan nama untuk kebijakan (misalnya, **`connection-permissions`**), lalu pilih Buat kebijakan.

   Anda akan melihat **`connection-permissions`** kebijakan yang dilampirkan pada kebijakan Izin peran Anda.

Tambahkan CodeBuild GitClone izin untuk tindakan CodeCommit sumber

Ketika pipeline Anda memiliki aksi CodeCommit sumber, ada dua cara Anda dapat meneruskan artefak input ke build:

- Default — Tindakan sumber menghasilkan file zip yang berisi kode yang CodeBuild diunduh.
- Klon penuh - Kode sumber dapat langsung diunduh ke lingkungan build.

  Opsi klon Penuh memungkinkan Anda untuk berinteraksi dengan kode sumber sebagai repositori Git yang berfungsi. Untuk menggunakan mode ini, Anda harus menambahkan izin untuk CodeBuild lingkungan Anda untuk menarik dari repositori Anda.

Untuk menambahkan izin ke kebijakan peran CodeBuild layanan, Anda membuat kebijakan yang dikelola pelanggan yang dilampirkan ke peran layanan Anda CodeBuild . Langkah-langkah berikut membuat kebijakan yang menentukan `codecommit:GitPull` izin di `action` bidang.

Untuk menggunakan konsol untuk menambahkan GitPull izin

1. Untuk menemukan peran CodeBuild layanan Anda, buka proyek build yang digunakan dalam pipeline Anda dan navigasikan ke tab Detail build.
2. Pilih tautan Peran layanan. Ini membuka konsol IAM tempat Anda dapat menambahkan kebijakan baru yang memberikan akses ke repositori Anda.
3. Pada konsol IAM, pilih Lampirkan kebijakan, lalu pilih Buat kebijakan.
4. Pada tab JSON, tempel kebijakan sampel berikut.

   ```
   {
       "Action": [
           "codecommit:GitPull"
       ],
       "Resource": "*",
       "Effect": "Allow"
   },
   ```

5. Pilih Tinjau kebijakan. Masukkan nama untuk kebijakan (misalnya, **`codecommit-gitpull`**), lalu pilih Buat kebijakan.

6. Kembali ke halaman tempat Anda melampirkan izin, menyegarkan daftar kebijakan, dan pilih kebijakan yang baru saja Anda buat. Pilih Lampirkan kebijakan.

<source artifact name>Kesalahan saluran pipa: Penerapan dengan tindakan CodeDeployTo ECS mengembalikan pesan kesalahan: “Pengecualian saat mencoba membaca file artefak definisi tugas dari:”

Masalah:

File definisi tugas adalah artefak yang diperlukan untuk tindakan CodePipeline penerapan ke Amazon ECS melalui CodeDeploy (tindakan). `CodeDeployToECS` Ukuran ZIP artefak maksimum dalam aksi `CodeDeployToECS` penerapan adalah 3 MB. Pesan galat berikut dikembalikan ketika file tidak ditemukan atau ukuran artefak melebihi 3 MB:

Pengecualian saat mencoba membaca file artefak definisi tugas dari: <source artifact name>

Kemungkinan perbaikan: Pastikan file definisi tugas disertakan sebagai artefak. Jika file sudah ada, pastikan ukuran terkompresi kurang dari 3 MB.

GitHub (melalui OAuth aplikasi) tindakan sumber: Daftar repositori menunjukkan repositori yang berbeda

Masalah:

Setelah otorisasi berhasil untuk tindakan GitHub (melalui OAuth aplikasi) di CodePipeline konsol, Anda dapat memilih dari daftar GitHub repositori Anda. Jika daftar tidak menyertakan repositori yang Anda harapkan untuk dilihat, maka Anda dapat memecahkan masalah akun yang digunakan untuk otorisasi.

Kemungkinan perbaikan: Daftar repositori yang disediakan di CodePipeline konsol didasarkan pada GitHub organisasi tempat akun resmi milik. Verifikasi bahwa akun yang Anda gunakan untuk mengotorisasi GitHub adalah akun yang terkait dengan GitHub organisasi tempat repositori Anda dibuat.

GitHub (melalui GitHub Aplikasi) tindakan sumber: Tidak dapat menyelesaikan koneksi untuk repositori

Masalah:

Karena koneksi ke GitHub repositori menggunakan AWS Connector for GitHub, Anda memerlukan izin pemilik organisasi atau izin admin ke repositori untuk membuat koneksi.

Kemungkinan perbaikan: Untuk informasi tentang tingkat izin untuk GitHub repositori, lihat https://docs.github.com/en/ free-pro-team @ -. latest/github/setting-up-and-managing-organizations-and-teams/permission levels-for-an-organization

Kesalahan Amazon S3: peran CodePipeline layanan <ARN>mendapatkan akses S3 ditolak untuk bucket S3 < > BucketName

Masalah:

Saat sedang berlangsung, CodeCommit tindakan dalam CodePipeline memeriksa apakah ember artefak pipa ada. Jika tindakan tidak memiliki izin untuk memeriksa, `AccessDenied` kesalahan terjadi di Amazon S3 dan pesan kesalahan berikut ditampilkan di: CodePipeline

CodePipeline peran layanan “arn:aws:iam: ::role/service-role/*`AccountID`*" mendapatkan akses S3 ditolak untuk ember S3 "" *`RoleID BucketName`*

CloudTrail Log untuk tindakan juga mencatat `AccessDenied` kesalahan.

Kemungkinan perbaikan: Lakukan hal berikut:

- Untuk kebijakan yang dilampirkan pada peran CodePipeline layanan Anda, tambahkan `s3:ListBucket` ke daftar tindakan dalam kebijakan Anda. Untuk petunjuk tentang cara melihat kebijakan peran layanan Anda, lihatLihat ARN pipeline dan peran layanan ARN (konsol). Edit pernyataan kebijakan untuk peran layanan Anda seperti yang dijelaskan dalamMenambahkan izin ke peran CodePipeline layanan.
- Untuk kebijakan berbasis sumber daya yang dilampirkan pada bucket artefak Amazon S3 untuk pipeline Anda, juga disebut kebijakan bucket artefak, tambahkan pernyataan untuk mengizinkan `s3:ListBucket` izin digunakan oleh peran layanan Anda. CodePipeline

Untuk menambahkan kebijakan Anda ke bucket artefak

1. Ikuti langkah-langkah Lihat ARN pipeline dan peran layanan ARN (konsol) untuk memilih bucket artefak Anda di halaman Pengaturan pipeline dan kemudian melihatnya di konsol Amazon S3.
2. Pilih Izin.
3. Di bagian Kebijakan bucket, pilih Edit.
4. Di bidang teks Kebijakan, masukkan kebijakan bucket baru, atau edit kebijakan yang ada seperti yang ditunjukkan pada contoh berikut. Kebijakan bucket adalah file JSON, jadi Anda harus memasukkan JSON yang valid.

   Contoh berikut menunjukkan pernyataan kebijakan bucket untuk bucket artefak tempat ID peran contoh untuk peran layanan berada*AROAEXAMPLEID*.

```
{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::BucketName",
    "Condition": {
        "StringLike": {
            "aws:userid": "AROAEXAMPLEID:*"
        }
    }
}
```

   Contoh berikut menunjukkan pernyataan kebijakan bucket yang sama setelah izin ditambahkan.

   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
                }
            }
        }
    ]
}
```

Untuk informasi selengkapnya, lihat langkah-langkahnyahttps://aws.amazon.com/blogs/security/writing-iam-policies-how-to-grant-access-to-an-amazon-s3-bucket/.

5. Pilih Simpan.

Setelah menerapkan kebijakan yang telah diedit, ikuti langkah-langkah Mulai pipa secara manual untuk menjalankan ulang pipeline secara manual.

Saluran pipa dengan Amazon S3, Amazon ECR, CodeCommit atau sumber tidak lagi dimulai secara otomatis

Masalah:

Setelah membuat perubahan pada pengaturan konfigurasi untuk tindakan yang menggunakan aturan peristiwa (EventBridgeatau CloudWatch Peristiwa) untuk deteksi perubahan, konsol mungkin tidak mendeteksi perubahan di mana pengidentifikasi pemicu sumber serupa dan memiliki karakter awal yang identik. Karena aturan acara baru tidak dibuat oleh konsol, pipeline tidak lagi dimulai secara otomatis.

Contoh perubahan kecil di akhir nama parameter untuk CodeCommit akan mengubah nama CodeCommit cabang Anda `MyTestBranch-1` menjadi`MyTestBranch-2`. Karena perubahan ada di akhir nama cabang, aturan acara untuk tindakan sumber mungkin tidak memperbarui atau membuat aturan untuk pengaturan sumber baru.

Ini berlaku untuk tindakan sumber yang menggunakan peristiwa CWE untuk deteksi perubahan sebagai berikut:

Tindakan sumber	Parameter/pengidentifikasi pemicu (konsol)
Amazon ECR	Nama repositori Tag gambar
Amazon S3	Bucket Kunci objek S3
CodeCommit	Nama repositori Nama cabang

Kemungkinan perbaikan:

Lakukan salah satu tindakan berikut:

- Ubah pengaturan konfigurasi CodeCommit /S3/ECR sehingga perubahan dilakukan pada bagian awal dari nilai parameter.

Contoh: Ubah nama cabang Anda `release-branch` menjadi`2nd-release-branch`. Hindari perubahan di akhir nama, seperti`release-branch-2`.

- Ubah pengaturan konfigurasi CodeCommit /S3/ECR untuk setiap pipeline.

  Contoh: Ubah nama cabang Anda `myRepo/myBranch` menjadi`myDeployRepo/myDeployBranch`. Hindari perubahan di akhir nama, seperti`myRepo/myBranch2`.
- Alih-alih konsol, gunakan CLI atau CloudFormation untuk membuat dan memperbarui aturan peristiwa deteksi perubahan Anda. Untuk petunjuk tentang membuat aturan acara untuk tindakan sumber S3, lihatMenghubungkan ke tindakan sumber Amazon S3 yang menggunakan dan EventBridge AWS CloudTrail. Untuk petunjuk cara membuat aturan acara untuk tindakan Amazon ECR, lihatTindakan sumber dan sumber daya Amazon ECR EventBridge. Untuk petunjuk tentang membuat aturan acara untuk CodeCommit tindakan, lihatCodeCommit tindakan sumber dan EventBridge.

Setelah Anda mengedit konfigurasi tindakan di konsol, terima sumber daya deteksi perubahan yang diperbarui yang dibuat oleh konsol.

Kesalahan koneksi saat menghubungkan ke GitHub: “Masalah terjadi, pastikan cookie diaktifkan di browser Anda” atau “Pemilik organisasi harus menginstal GitHub aplikasi”

Masalah:

Untuk membuat koneksi untuk tindakan GitHub sumber di CodePipeline, Anda harus menjadi pemilik GitHub organisasi. Untuk repositori yang tidak berada di bawah organisasi, Anda harus menjadi pemilik repositori. Ketika koneksi dibuat oleh orang lain selain pemilik organisasi, permintaan dibuat untuk pemilik organisasi, dan salah satu kesalahan berikut akan ditampilkan:

Masalah terjadi, pastikan cookie diaktifkan di browser Anda

ATAU

Pemilik organisasi harus menginstal GitHub aplikasi

Kemungkinan perbaikan: Untuk repositori dalam GitHub organisasi, pemilik organisasi harus membuat koneksi ke repositori. GitHub Untuk repositori yang tidak berada di bawah organisasi, Anda harus menjadi pemilik repositori.

Pipelines dengan mode eksekusi diubah menjadi mode QUEUED atau PARALLEL gagal saat batas run tercapai

Masalah: Jumlah maksimum eksekusi bersamaan untuk pipeline dalam mode ANTRIAN adalah 50 eksekusi. Ketika batas ini tercapai, pipeline gagal tanpa pesan status.

Kemungkinan perbaikan: Saat mengedit definisi pipeline untuk mode eksekusi, buat pengeditan secara terpisah dari tindakan edit lainnya.

Untuk informasi selengkapnya tentang mode eksekusi ANTRIAN atau PARALLEL, lihat. CodePipeline konsep

Pipelines dalam mode PARALLEL memiliki definisi pipeline yang sudah ketinggalan zaman jika diedit saat mengubah ke mode QUEUED atau SUPERSEDED

Masalah: Untuk pipeline dalam mode parallel, saat mengedit mode eksekusi pipeline ke QUEUED atau SUPERSEDED, definisi pipeline untuk mode PARALLEL tidak akan diperbarui. Definisi pipeline yang diperbarui saat memperbarui mode PARALLEL tidak digunakan dalam mode SUPERSEDED atau ANTRIAN

Kemungkinan perbaikan: Untuk pipeline dalam mode parallel, saat mengedit mode eksekusi pipeline ke QUEUED atau SUPERSEDED, hindari memperbarui definisi pipeline secara bersamaan.

Untuk informasi selengkapnya tentang mode eksekusi ANTRIAN atau PARALLEL, lihat. CodePipeline konsep

Pipelines diubah dari mode PARALLEL akan menampilkan mode eksekusi sebelumnya

Masalah: Untuk saluran pipa dalam mode PARALLEL, saat mengedit mode eksekusi pipa ke ANTRIAN atau DIGANTI, status pipa tidak akan menampilkan status yang diperbarui sebagai PARALLEL. Jika pipeline berubah dari PARALLEL ke QUEUED atau SUPERSEDED, status untuk pipeline dalam mode SUPERSEDED atau QUEUED akan menjadi status terakhir yang diketahui di salah satu mode tersebut. Jika pipeline tidak pernah dijalankan dalam mode itu sebelumnya, maka statusnya akan kosong.

Kemungkinan perbaikan: Untuk pipeline dalam mode parallel, saat mengedit mode eksekusi pipeline ke QUEUED atau SUPERSEDED, perhatikan bahwa tampilan mode eksekusi tidak akan menampilkan status PARALLEL.

Untuk informasi selengkapnya tentang mode eksekusi ANTRIAN atau PARALLEL, lihat. CodePipeline konsep

Saluran pipa dengan koneksi yang menggunakan pemfilteran pemicu berdasarkan jalur file mungkin tidak dimulai pada pembuatan cabang

Deskripsi: Untuk pipeline dengan tindakan sumber yang menggunakan koneksi, seperti aksi BitBucket sumber, Anda dapat mengatur pemicu dengan konfigurasi Git yang memungkinkan Anda memfilter menurut jalur file untuk memulai pipeline. Dalam kasus tertentu, untuk saluran pipa dengan pemicu yang difilter pada jalur file, pipeline mungkin tidak dimulai ketika cabang dengan filter jalur file pertama kali dibuat, karena ini tidak memungkinkan CodeConnections koneksi untuk menyelesaikan file yang berubah. Ketika konfigurasi Git untuk pemicu diatur untuk memfilter pada jalur file, pipeline tidak akan dimulai ketika cabang dengan filter baru saja dibuat di repositori sumber, Untuk informasi lebih lanjut tentang pemfilteran pada jalur file, lihat. Tambahkan pemicu dengan tipe acara permintaan push atau pull kode

Hasil: Misalnya, pipeline CodePipeline yang memiliki filter jalur file pada cabang “B” tidak akan dipicu saat cabang “B” dibuat. Jika tidak ada filter jalur file, pipeline akan tetap dimulai.

Saluran pipa dengan koneksi yang menggunakan pemfilteran pemicu berdasarkan jalur file mungkin tidak dimulai saat batas file tercapai

Deskripsi: Untuk pipeline dengan tindakan sumber yang menggunakan koneksi, seperti aksi BitBucket sumber, Anda dapat mengatur pemicu dengan konfigurasi Git yang memungkinkan Anda memfilter menurut jalur file untuk memulai pipeline. CodePipeline mengambil hingga 100 file pertama; oleh karena itu, ketika konfigurasi Git untuk pemicu diatur untuk memfilter pada jalur file, pipeline mungkin tidak dimulai jika ada lebih dari 100 file, Untuk informasi lebih lanjut tentang pemfilteran pada jalur file, lihat. Tambahkan pemicu dengan tipe acara permintaan push atau pull kode

Hasil: Misalnya, jika diff berisi 150 file, CodePipeline lihat 100 file pertama (tanpa urutan tertentu) untuk memeriksa filter jalur file yang ditentukan. Jika file yang cocok dengan filter jalur file tidak termasuk di antara 100 file yang diambil oleh CodePipeline, pipeline tidak akan dipanggil.

CodeCommit atau revisi sumber S3 dalam mode PARALLEL mungkin tidak cocok dengan acara EventBridge

Deskripsi: Untuk eksekusi pipeline dalam mode PARALLEL, eksekusi mungkin dimulai dengan perubahan terbaru, seperti komit CodeCommit repositori, yang mungkin tidak sama dengan perubahan untuk acara tersebut. EventBridge Dalam beberapa kasus, di mana sepersekian detik mungkin antara komit atau tag gambar yang memulai pipeline, saat CodePipeline menerima acara dan memulai eksekusi itu, komit atau tag gambar lain telah didorong, CodePipeline (misalnya, CodeCommit tindakan) akan mengkloning komit HEAD pada saat itu.

Hasil: Untuk pipeline dalam mode PARALLEL dengan sumber CodeCommit atau S3, terlepas dari perubahan yang memicu eksekusi pipeline, aksi sumber akan selalu mengkloning HEAD pada saat dimulai. Misalnya, untuk pipeline dalam mode PARALLEL, komit didorong, yang memulai pipeline untuk eksekusi 1, dan eksekusi pipeline kedua menggunakan komit kedua.

Tindakan Penerapan EC2 gagal dengan pesan kesalahan **`No such file`**

Deskripsi: Setelah tindakan penerapan EC2 membuka ritsleting artefak di direktori target pada instance, tindakan menjalankan skrip. Jika skrip ada di direktori target tetapi tindakan tidak dapat menjalankan skrip, tindakan gagal pada instance itu, dan instance yang tersisa gagal diterapkan.

Kesalahan yang mirip dengan pesan kesalahan berikut ditampilkan di log untuk penerapan di mana direktori target berada `/home/ec2-user/deploy/` dan jalur repositori sumber berada. `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
  ```

Hasil: Tindakan penerapan gagal dalam pipeline.

Kemungkinan perbaikan: Untuk memecahkan masalah, gunakan langkah-langkah berikut.

1. Lihat log untuk memverifikasi instance mana yang menyebabkan skrip gagal.
2. Ubah directory (`cd`) ke direktori target pada instance Anda. Uji jalankan skrip pada instance.
3. Di repositori sumber Anda, edit file skrip untuk menghapus komentar atau perintah apa pun yang mungkin menyebabkan masalah.

Tindakan Penerapan EKS gagal dengan pesan **`cluster unreachable`** kesalahan

Deskripsi: Setelah tindakan penerapan EKS berjalan, tindakan gagal dengan pesan `cluster unreachable` kesalahan. Pesan menunjukkan masalah akses pada klaster karena izin yang hilang. Berdasarkan jenis file (Helm chart atau file manifes Kubernetes), pesan kesalahan ditampilkan sebagai berikut.

- Untuk tindakan penerapan EKS yang menggunakan bagan Helm, kesalahan yang mirip dengan pesan kesalahan berikut akan ditampilkan.

  ```
  error message:

  helm upgrade --install my-release test-chart --wait
  Error: Kubernetes cluster unreachable: the server has asked for the client to provide
   credentials
  ```

- Untuk tindakan penerapan EKS yang menggunakan file manifes Kubernetes, kesalahan yang mirip dengan pesan kesalahan berikut akan ditampilkan.

  ```
  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
  ```
  ```

Hasil: Tindakan penerapan gagal dalam pipeline.

Kemungkinan perbaikan: Jika menggunakan peran yang ada, peran CodePipeline layanan harus diperbarui dengan izin yang diperlukan untuk menggunakan tindakan penerapan EKS. Selain itu, untuk mengizinkan akses peran CodePipeline layanan ke kluster Anda, Anda harus menambahkan entri akses ke klaster dan menentukan peran layanan untuk entri akses.

1. Verifikasi bahwa peran CodePipeline layanan memiliki izin yang diperlukan untuk tindakan penerapan EKS. Untuk referensi izin, lihatIzin kebijakan peran layanan.
2. Tambahkan entri akses ke klaster Anda dan tentukan peran CodePipeline layanan untuk akses tersebut. Sebagai contoh, lihat Langkah 4: Buat entri akses untuk peran CodePipeline layanan.

Butuh bantuan dengan masalah yang berbeda?

Coba sumber daya lain ini:

- Hubungi AWS Support.
- Ajukan pertanyaan di CodePipelineforum.
- Minta peningkatan kuota. Untuk informasi selengkapnya, lihat Kuota di AWS CodePipeline.

> **Note**
>
> Diperlukan waktu hingga dua minggu untuk memproses permintaan peningkatan kuota.

Keamanan di AWS CodePipeline

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang sesuai untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menggambarkan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang berjalan Layanan AWS di dalamnya AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara berkala menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku AWS CodePipeline, lihat Layanan AWS cakupan berdasarkan program kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh Layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor-faktor lain, termasuk sensitivitas data Anda, persyaratan perusahaan Anda, dan hukum dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan CodePipeline. Topik berikut menunjukkan cara mengonfigurasi CodePipeline untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan Layanan AWS yang lain yang membantu Anda memantau dan mengamankan CodePipeline sumber daya Anda.

Topik

- Perlindungan data di AWS CodePipeline
- Identitas dan manajemen akses untuk AWS CodePipeline
- Penebangan dan pemantauan di CodePipeline
- Validasi kepatuhan untuk AWS CodePipeline
- Ketahanan di AWS CodePipeline
- Keamanan infrastruktur di AWS CodePipeline
- Praktik terbaik keamanan

Perlindungan data di AWS CodePipeline

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di AWS CodePipeline. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensil dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-3.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan CodePipeline atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDKs. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan

atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Praktik terbaik keamanan berikut juga membahas perlindungan data di CodePipeline:

- Konfigurasikan enkripsi sisi server untuk artefak yang disimpan di Amazon S3 CodePipeline
- Gunakan AWS Secrets Manager untuk melacak kata sandi database atau kunci API pihak ketiga

Privasi lalu lintas antarjaringan

Amazon VPC adalah Layanan AWS yang dapat Anda gunakan untuk meluncurkan AWS sumber daya di jaringan virtual (virtual private cloud) yang Anda tentukan. CodePipelinemendukung endpoint Amazon VPC yang didukung oleh AWS PrivateLink, sebuah AWS teknologi yang memfasilitasi komunikasi pribadi antara Layanan AWS menggunakan antarmuka network elastis dengan alamat IP pribadi. Ini berarti Anda dapat terhubung langsung CodePipeline melalui titik akhir pribadi di VPC Anda, menjaga semua lalu lintas di dalam VPC dan jaringan Anda. AWS Sebelumnya, aplikasi yang berjalan di dalam VPC memerlukan akses internet untuk terhubung. CodePipeline Dengan VPC, Anda memiliki kontrol atas pengaturan jaringan Anda, seperti:

- Rentang alamat IP,
- Subnet,
- Tabel rute, dan
- Gerbang jaringan.

Untuk menghubungkan VPC Anda CodePipeline, Anda menentukan titik akhir VPC antarmuka untuk. CodePipeline Jenis titik akhir ini memungkinkan Anda untuk menghubungkan Layanan AWS VPC Anda ke. Endpoint menyediakan konektivitas yang andal dan dapat diskalakan CodePipeline tanpa memerlukan gateway internet, instance terjemahan alamat jaringan (NAT), atau koneksi VPN. Untuk informasi tentang pengaturan VPC, lihat Panduan Pengguna VPC.

Enkripsi saat diam

Data dalam CodePipeline dienkripsi saat istirahat menggunakan. AWS KMS keys Artefak kode disimpan dalam bucket S3 milik pelanggan dan dienkripsi dengan kunci atau yang dikelola pelanggan. Kunci yang dikelola AWS Untuk informasi selengkapnya, lihat Konfigurasikan enkripsi sisi server untuk artefak yang disimpan di Amazon S3 CodePipeline.

Enkripsi saat bergerak

Semua service-to-service komunikasi dienkripsi dalam perjalanan menggunakan SSL/TLS.

Pengelolaan kunci enkripsi

Jika Anda memilih opsi default untuk mengenkripsi artefak kode, CodePipeline gunakan file. Kunci yang dikelola AWS Anda tidak dapat mengubah atau menghapus ini Kunci yang dikelola AWS. Jika Anda menggunakan kunci yang dikelola pelanggan AWS KMS untuk mengenkripsi atau mendekripsi artefak di bucket S3, Anda dapat mengubah atau memutar kunci yang dikelola pelanggan ini seperlunya.

> **Important**
>
> CodePipeline hanya mendukung tombol KMS simetris. Jangan gunakan kunci KMS asimetris untuk mengenkripsi data di bucket S3 Anda.

Konfigurasikan enkripsi sisi server untuk artefak yang disimpan di Amazon S3 CodePipeline

Ada dua cara untuk mengonfigurasi enkripsi sisi server untuk artefak Amazon S3:

- CodePipeline membuat bucket artefak S3 dan default Kunci yang dikelola AWS saat Anda membuat pipeline menggunakan wizard Create Pipeline. Kunci yang dikelola AWS Ini dienkripsi bersama dengan data objek dan dikelola oleh. AWS
- Anda dapat membuat dan mengelola kunci yang dikelola pelanggan Anda sendiri.

> **Important**
>
> CodePipeline hanya mendukung tombol KMS simetris. Jangan gunakan kunci KMS asimetris untuk mengenkripsi data di bucket S3 Anda.

Jika Anda menggunakan tombol S3 default, Anda tidak dapat mengubah atau menghapus ini Kunci yang dikelola AWS. Jika Anda menggunakan kunci yang dikelola pelanggan AWS KMS untuk mengenkripsi atau mendekripsi artefak di bucket S3, Anda dapat mengubah atau memutar kunci yang dikelola pelanggan ini seperlunya.

Amazon S3 mendukung kebijakan bucket yang dapat Anda gunakan jika Anda memerlukan enkripsi sisi server untuk semua objek yang disimpan di bucket. Misalnya, kebijakan bucket berikut menolak izin mengunggah objek (`s3:PutObject`) kepada semua orang jika permintaan tidak mencakup header `x-amz-server-side-encryption` yang meminta enkripsi sisi server dengan 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"
                }
            }
        }
    ]
}
```

Untuk informasi selengkapnya tentang enkripsi sisi server dan AWS KMS, lihat Melindungi Data Menggunakan Enkripsi Sisi Server dan Melindungi data menggunakan enkripsi sisi server dengan kunci KMS yang disimpan di (SSE-KMS). AWS Key Management Service

Untuk informasi selengkapnya AWS KMS, lihat Panduan AWS Key Management Service Pengembang.

Topik

- Lihat Anda Kunci yang dikelola AWS
- Konfigurasikan enkripsi sisi server untuk bucket S3 menggunakan atau CloudFormation AWS CLI

Lihat Anda Kunci yang dikelola AWS

Saat Anda menggunakan wizard Create Pipeline untuk membuat pipeline pertama, bucket S3 akan dibuat untuk Anda di Wilayah yang sama dengan pipeline yang Anda buat. Ember digunakan untuk menyimpan artefak pipa. Saat pipa berjalan, artefak dimasukkan ke dalam dan diambil dari ember S3. Secara default, CodePipeline menggunakan enkripsi sisi server dengan AWS KMS menggunakan untuk Amazon Kunci yang dikelola AWS S3 (kuncinya). `aws/s3` Kunci yang dikelola AWS Ini dibuat dan disimpan di AWS akun Anda. Saat artefak diambil dari bucket S3, CodePipeline gunakan proses SSE-KMS yang sama untuk mendekripsi artefak.

Untuk melihat informasi tentang Kunci yang dikelola AWS

1. Masuk ke Konsol Manajemen AWS dan buka AWS KMS konsol.
2. Jika halaman selamat datang muncul, pilih Mulai sekarang.
3. Di panel navigasi layanan, pilih tombol AWS terkelola.
4. Pilih Wilayah untuk pipeline Anda. Misalnya, jika pipa dibuat di`us-east-2`, pastikan filter disetel ke US East (Ohio).

   Untuk informasi selengkapnya tentang Wilayah dan titik akhir yang tersedia CodePipeline, lihat AWS CodePipeline titik akhir dan kuota.
5. Dalam daftar, pilih kunci dengan alias yang digunakan untuk pipeline Anda (secara default, aws/s3). Informasi dasar tentang kunci ditampilkan.

Konfigurasikan enkripsi sisi server untuk bucket S3 menggunakan atau CloudFormation AWS CLI

Bila Anda menggunakan CloudFormation atau AWS CLI untuk membuat pipeline, Anda harus mengkonfigurasi enkripsi sisi server secara manual. Gunakan contoh kebijakan bucket di atas, lalu buat kunci terkelola pelanggan Anda sendiri. Anda juga dapat menggunakan kunci Anda sendiri alih-alih tombol Kunci yang dikelola AWS. Beberapa alasan untuk memilih kunci Anda sendiri meliputi:

- Anda ingin memutar kunci pada jadwal untuk memenuhi persyaratan bisnis atau keamanan untuk organisasi Anda.
- Anda ingin membuat pipeline yang menggunakan sumber daya yang terkait dengan AWS akun lain. Ini membutuhkan penggunaan kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Buat pipeline CodePipeline yang menggunakan sumber daya dari AWS akun lain.

Praktik terbaik kriptografi mencegah penggunaan ulang kunci enkripsi secara ekstensif. Sebagai praktik terbaik, putar kunci Anda secara teratur. Untuk membuat materi kriptografi baru untuk AWS KMS kunci Anda, Anda dapat membuat kunci yang dikelola pelanggan, dan kemudian mengubah aplikasi atau alias Anda untuk menggunakan kunci yang dikelola pelanggan baru. Atau, Anda dapat mengaktifkan rotasi kunci otomatis untuk kunci terkelola pelanggan yang ada.

Untuk memutar kunci terkelola pelanggan Anda, lihat Memutar kunci.

> **Important**
>
> CodePipeline hanya mendukung tombol KMS simetris. Jangan gunakan kunci KMS asimetris untuk mengenkripsi data di bucket S3 Anda.

Gunakan AWS Secrets Manager untuk melacak kata sandi database atau kunci API pihak ketiga

Sebaiknya gunakan AWS Secrets Manager untuk memutar, mengelola, dan mengambil kredenal database, kunci API, dan rahasia lainnya sepanjang siklus hidupnya. Secrets Manager memungkinkan Anda mengganti kredensi hardcode dalam kode Anda (termasuk kata sandi) dengan panggilan API ke Secrets Manager untuk mengambil rahasia secara terprogram. Untuk informasi selengkapnya, lihat Apa itu AWS Secrets Manager? di Panduan Pengguna AWS Secrets Manager.

Untuk pipeline di mana Anda meneruskan parameter yang merupakan rahasia (seperti OAuth kredensil) dalam CloudFormation template, Anda harus menyertakan referensi dinamis dalam template Anda yang mengakses rahasia yang telah Anda simpan di Secrets Manager. Untuk pola ID referensi dan contoh, lihat Secrets Manager Secrets di Panduan AWS CloudFormation Pengguna. Untuk contoh yang menggunakan referensi dinamis dalam cuplikan templat untuk GitHub webhook dalam pipeline, lihat Konfigurasi Sumber Daya Webhook.

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan mengelola rahasia.

- Secrets Manager dapat memutar kredensyal database secara otomatis, seperti untuk rotasi rahasia Amazon RDS. Untuk informasi selengkapnya, lihat Memutar AWS Rahasia Secrets Manager Anda di Panduan Pengguna AWS Secrets Manager.
- Untuk melihat petunjuk untuk menambahkan referensi dinamis Secrets Manager ke CloudFormation template Anda, lihathttps://aws.amazon.com/blogs/security/how-to-create-and-retrieve-secrets-managed-in-aws-secrets-manager-using-aws-cloudformation-template/.

Identitas dan manajemen akses untuk AWS CodePipeline

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya. CodePipeline IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Audiens
- Mengautentikasi dengan identitas
- Mengelola akses menggunakan kebijakan
- Bagaimana AWS CodePipeline bekerja dengan IAM
- AWS CodePipeline contoh kebijakan berbasis identitas
- AWS CodePipeline contoh kebijakan berbasis sumber daya
- Memecahkan masalah AWS CodePipeline identitas dan akses
- CodePipeline referensi izin
- Kelola peran CodePipeline layanan

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda berdasarkan peran Anda:

- Pengguna layanan - minta izin dari administrator Anda jika Anda tidak dapat mengakses fitur (lihat Memecahkan masalah AWS CodePipeline identitas dan akses)
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat Bagaimana AWS CodePipeline bekerja dengan IAM)
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat AWS CodePipeline contoh kebijakan berbasis identitas)

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensional dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensional. Google/Facebook Untuk informasi selengkapnya tentang cara masuk, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS sediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat AWS Signature Version 4 untuk permintaan API dalam Panduan Pengguna IAM.

Pengguna root akun AWS

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

Pengguna IAM adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk

informasi selengkapnya, lihat Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensi sementara di Panduan Pengguna IAM.

Grup IAM menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan beralih dari pengguna ke peran IAM (konsol) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini ketika kepala sekolah membuat permintaan. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh

identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada banyak identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contohnya termasuk kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Anda harus menentukan principal dalam kebijakan berbasis sumber daya.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCPs) — Tentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam Panduan Pengguna AWS Organizations .
- Kebijakan kontrol sumber daya (RCPs) — Tetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat Kebijakan kontrol sumber daya (RCPs) di Panduan AWS Organizations Pengguna.
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Bagaimana AWS CodePipeline bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses CodePipeline, Anda harus memahami fitur IAM apa yang tersedia untuk digunakan. CodePipeline Untuk mendapatkan tampilan tingkat tinggi tentang bagaimana CodePipeline dan lainnya Layanan AWS yang bekerja dengan IAM, lihat Layanan AWS yang bekerja dengan IAM di Panduan Pengguna IAM.

Topik

- CodePipeline kebijakan berbasis identitas
- CodePipeline kebijakan berbasis sumber daya
- Otorisasi berdasarkan tag CodePipeline
- CodePipeline Peran IAM

CodePipeline kebijakan berbasis identitas

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak, serta kondisi di mana tindakan tersebut diperbolehkan atau ditolak. CodePipeline mendukung tindakan tertentu, sumber daya, dan kunci syarat. Untuk mempelajari semua elemen yang Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan IAM JSON dalam Panduan Pengguna IAM.

Tindakan

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Tindakan kebijakan CodePipeline menggunakan awalan berikut sebelum tindakan:`codepipeline:`.

Misalnya, untuk memberikan izin kepada seseorang untuk melihat pipeline yang ada di akun, Anda menyertakan `codepipeline:GetPipeline` tindakan tersebut dalam kebijakan mereka. Pernyataan kebijakan harus mencakup salah satu `Action` atau `NotAction` elemen. CodePipeline mendefinisikan serangkaian tindakannya sendiri yang menggambarkan tugas yang dapat Anda lakukan dengan layanan ini.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, pisahkan dengan koma seperti berikut:

```
"Action": [
      "codepipeline:action1",
      "codepipeline:action2"
```

Anda dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Get`, sertakan tindakan berikut:

```
"Action": "codepipeline:Get*"
```

Untuk daftar tindakan, lihat CodePipeline Tindakan yang Ditentukan oleh AWS CodePipeline dalam Panduan Pengguna IAM.

Sumber daya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

CodePipeline sumber daya dan operasi

Pada tahun CodePipeline, sumber daya utama adalah pipa. Dalam kebijakan, Anda menggunakan Nama Sumber Daya Amazon (ARN) untuk mengidentifikasi sumber daya yang berlaku untuk kebijakan tersebut. CodePipeline mendukung sumber daya lain yang dapat digunakan dengan sumber daya utama, seperti tahapan, tindakan, dan tindakan khusus. Ini dirujuk sebagai sub-sumber daya. Sumber daya dan subsumber daya ini memiliki Nama Sumber Daya Amazon (ARNs) unik yang terkait dengannya. Untuk informasi selengkapnya ARNs, lihat Nama Sumber Daya Amazon (ARN)

dan Layanan AWS ruang nama di. Referensi Umum Amazon Web Services Untuk mendapatkan ARN pipeline yang terkait dengan pipeline Anda, Anda dapat menemukan ARN pipeline di bawah Pengaturan di konsol. Untuk informasi selengkapnya, lihat Lihat ARN pipeline dan peran layanan ARN (konsol).

Jenis Sumber Daya	Format ARN
Pipa	arn:aws:codepipeline:: *region account pipeline-name*
Stage	arn:aws:codepipeline:::/*regionaccountpipeline-name stage-name*
Tindakan	arn:aws:codepipeline:::///*regionaccountpipeline-name stage-name action-name*
Tindakan kustom	arn:aws:codepipeline: ::actiont ype:///*regionaccountownercategoryproviderversion*
Semua CodePipeline sumber daya	arn:aws:codepipeline: *
Semua CodePipeline sumber daya yang dimiliki oleh akun yang ditentukan di Wilayah yang ditentukan	arn:aws:codepipeline::: * *region account*

> **Note**
>
> Sebagian besar layanan dalam AWS memperlakukan titik dua (:) atau garis miring (/) sebagai karakter yang sama di ARNs. Namun, CodePipeline menggunakan kecocokan tepat dalam pola dan aturan acara. Pastikan untuk menggunakan karakter ARN yang benar saat membuat pola acara sehingga cocok dengan sintaks ARN di pipeline yang ingin Anda cocokkan.

Di CodePipeline, ada panggilan API yang mendukung izin tingkat sumber daya. Izin tingkat sumber daya menunjukkan apakah panggilan API dapat menentukan ARN sumber daya, atau

apakah panggilan API hanya dapat menentukan semua sumber daya menggunakan wildcard. Lihat CodePipeline referensi izin penjelasan terperinci tentang izin tingkat sumber daya dan daftar panggilan CodePipeline API yang mendukung izin tingkat sumber daya.

Misalnya, Anda dapat menunjukkan pipeline (*myPipeline*) tertentu dalam pernyataan Anda menggunakan ARN sebagai berikut:

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:myPipeline"
```

Anda juga dapat menentukan semua pipeline milik akun tertentu dengan menggunakan karakter wildcard (*) sebagai berikut:

```
"Resource": "arn:aws:codepipeline:us-east-2:111222333444:*"
```

Untuk menentukan semua sumber daya, atau jika tindakan API tertentu tidak mendukung ARNs, gunakan karakter wildcard (*) dalam `Resource` elemen sebagai berikut:

```
"Resource": "*"
```

> **Note**
>
> Saat Anda membuat kebijakan IAM, ikuti saran keamanan standar untuk memberikan hak istimewa paling sedikit—yaitu, hanya memberikan izin yang diperlukan untuk melakukan tugas. Jika panggilan API mendukung ARNs, maka panggilan tersebut mendukung izin tingkat sumber daya, dan Anda tidak perlu menggunakan karakter wildcard (*).

Beberapa panggilan CodePipeline API menerima beberapa sumber daya (misalnya,`GetPipeline`). Untuk menentukan beberapa sumber daya dalam satu pernyataan, pisahkan ARNs dengan koma, sebagai berikut:

```
"Resource": ["arn1", "arn2"]
```

CodePipeline menyediakan satu set operasi untuk bekerja dengan CodePipeline sumber daya. Untuk daftar operasi yang tersedia, lihat CodePipeline referensi izin.

Kunci syarat

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Elemen `Condition` menentukan ketika pernyataan dieksekusi berdasarkan kriteria yang ditetapkan. Anda dapat membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

CodePipeline mendefinisikan kumpulan kunci kondisinya sendiri dan juga mendukung penggunaan beberapa kunci kondisi global. Untuk melihat semua kunci kondisi AWS global, lihat Kunci Konteks Kondisi AWS Global di Panduan Pengguna IAM.

Semua tindakan Amazon EC2 men-support kunci syarat `aws:RequestedRegion` dan `ec2:Region`. Untuk informasi selengkapnya, lihat Contoh: Membatasi Akses ke Wilayah Tertentu.

Untuk melihat daftar kunci CodePipeline kondisi, lihat Kunci Kondisi untuk AWS CodePipeline dalam Panduan Pengguna IAM. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang Ditentukan oleh AWS CodePipeline.

Contoh

Untuk melihat contoh kebijakan CodePipeline berbasis identitas, lihat. AWS CodePipeline contoh kebijakan berbasis identitas

CodePipeline kebijakan berbasis sumber daya

CodePipeline tidak mendukung kebijakan berbasis sumber daya. Namun, contoh kebijakan berbasis sumber daya untuk layanan S3 yang terkait dengan disediakan. CodePipeline

Contoh

Untuk melihat contoh kebijakan CodePipeline berbasis sumber daya, lihat, AWS CodePipeline contoh kebijakan berbasis sumber daya

Otorisasi berdasarkan tag CodePipeline

Anda dapat melampirkan tag ke CodePipeline sumber daya atau meneruskan tag dalam permintaan CodePipeline. Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di elemen kondisi dari kebijakan menggunakan kunci kondisi `codepipeline:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Untuk informasi selengkapnya tentang penandaan sumber daya CodePipeline , lihat Penandaan pada sumber daya.

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat Menggunakan tag untuk mengontrol akses ke CodePipeline sumber daya.

CodePipeline Peran IAM

Peran IAM adalah entitas di AWS akun Anda yang memiliki izin tertentu.

Menggunakan kredensi sementara dengan CodePipeline

Anda dapat menggunakan kredensial sementara untuk masuk dengan gabungan, menjalankan IAM role, atau menjalankan peran lintas akun. Anda memperoleh kredensil keamanan sementara dengan memanggil operasi AWS STS API seperti AssumeRoleatau. GetFederationToken

CodePipeline mendukung penggunaan kredensyal sementara.

Peran layanan

CodePipeline memungkinkan layanan untuk mengambil peran layanan atas nama Anda. Peran ini mengizinkan layanan untuk mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran layanan muncul di akun IAM Anda dan dimiliki oleh akun tersebut. Ini berarti administrator IAM dapat mengubah izin untuk peran ini. Namun, melakukan hal itu dapat merusak fungsionalitas layanan.

CodePipeline mendukung peran layanan.

AWS CodePipeline contoh kebijakan berbasis identitas

Secara default, pengguna dan peran IAM tidak memiliki izin untuk membuat atau memodifikasi sumber daya CodePipeline . Mereka juga tidak dapat melakukan tugas menggunakan Konsol Manajemen AWS, AWS CLI, atau AWS API. Administrator IAM harus membuat kebijakan IAM yang

memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber daya yang diperlukan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna IAM atau grup yang memerlukan izin tersebut.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat Kebijakan pada Tab JSON dalam Panduan Pengguna IAM.

Untuk mempelajari cara membuat pipeline yang menggunakan sumber daya dari akun lain, dan untuk contoh kebijakan terkait, lihatBuat pipeline CodePipeline yang menggunakan sumber daya dari AWS akun lain.

Topik

- Praktik terbaik kebijakan
- Menampilkan sumber daya di konsol
- Mengizinkan pengguna melihat izin mereka sendiri
- Contoh kebijakan berbasis identitas (IAM)
- Menggunakan tag untuk mengontrol akses ke CodePipeline sumber daya
- Izin yang diperlukan untuk menggunakan konsol CodePipeline
- Izin diperlukan untuk melihat log komputasi di konsol CodePipeline
- AWS kebijakan terkelola untuk AWS CodePipeline
- Contoh kebijakan yang dikelola pelanggan

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus CodePipeline sumber daya di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS atau Kebijakan yang dikelola AWS untuk fungsi tugas dalam Panduan Pengguna IAM.

- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat Kebijakan dan izin dalam IAM dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan dengan IAM Access Analyzer dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Amankan akses API dengan MFA dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Menampilkan sumber daya di konsol

CodePipeline Konsol memerlukan `ListRepositories` izin untuk menampilkan daftar repositori untuk AWS akun Anda di AWS Wilayah tempat Anda masuk. Konsol juga termasuk fungsi Pergi ke sumber daya untuk secara cepat melakukan pencarian sensitif huruf besar/kecil untuk sumber daya. Pencarian ini dilakukan di AWS akun Anda di AWS Wilayah tempat Anda masuk. Sumber daya berikut ditampilkan di seluruh layanan berikut:

- AWS CodeBuild: Bangun proyek
- AWS CodeCommit: Repositori
- AWS CodeDeploy: Aplikasi
- AWS CodePipeline: Alur

Untuk melakukan pencarian ini di sumber daya di semua layanan, Anda harus memiliki izin berikut:

- CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`
- CodePipeline: `ListPipelines`

Hasil tidak dikembalikan untuk sumber daya layanan jika Anda tidak memiliki izin untuk layanan tersebut. Bahkan jika Anda memiliki izin untuk melihat sumber daya, beberapa sumber daya tidak dikembalikan jika ada eksplisit `Deny` untuk melihat sumber daya tersebut.

Mengizinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI 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": "*"
        }
    ]
}
```

Contoh kebijakan berbasis identitas (IAM)

Anda dapat melampirkan kebijakan ke identitas IAM. Misalnya, Anda dapat melakukan hal berikut:

- Lampirkan kebijakan izin ke pengguna atau grup di akun Anda — Untuk memberikan izin pengguna untuk melihat saluran pipa di CodePipeline konsol, Anda dapat melampirkan kebijakan izin ke pengguna atau grup tempat pengguna tersebut berada.
- Melampirkan kebijakan izin pada peran (memberikan izin lintas akun) – Anda dapat melampirkan kebijakan izin berbasis identitas ke peran IAM untuk memberikan izin lintas akun. Misalnya, administrator di Akun A dapat membuat peran untuk memberikan izin lintas akun ke AWS akun lain (misalnya, Akun B) atau Layanan AWS sebagai berikut:
  1. Administrator akun A membuat peran IAM dan melampirkan kebijakan izin ke peran ini yang memberikan izin pada sumber daya di akun A.
  2. Administrator akun A melampirkan kebijakan kepercayaan ke peran yang mengidentifikasi Akun B sebagai prinsipal yang dapat mengambil peran tersebut.
  3. Administrator Akun B kemudian dapat mendelegasikan izin untuk mengambil peran kepada setiap pengguna di Akun B. Melakukan hal ini memungkinkan pengguna di Akun B untuk membuat atau mengakses sumber daya di Akun A. Prinsip dalam kebijakan kepercayaan juga dapat menjadi Layanan AWS prinsipal jika Anda ingin memberikan Layanan AWS izin untuk mengambil peran.

Untuk informasi selengkapnya tentang penggunaan IAM untuk mendelegasikan izin, lihat Manajemen Akses dalam Panduan Pengguna IAM.

Berikut ini menunjukkan contoh kebijakan izin yang memberikan izin untuk menonaktifkan dan mengaktifkan transisi di antara semua tahapan dalam pipeline yang disebutkan dalam: `MyFirstPipeline` `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/*"
      ]
    }
  ]
}
```

Contoh berikut menunjukkan kebijakan di akun 111222333444 yang memungkinkan pengguna untuk melihat, tetapi tidak mengubah, pipeline yang dinamai di konsol. `MyFirstPipeline` CodePipeline Kebijakan ini didasarkan pada kebijakan yang `AWSCodePipeline_ReadOnlyAccess` dikelola, tetapi karena khusus untuk `MyFirstPipeline` pipeline, kebijakan ini tidak dapat menggunakan kebijakan terkelola secara langsung. Jika Anda tidak ingin membatasi kebijakan ke pipeline tertentu, pertimbangkan untuk menggunakan salah satu kebijakan terkelola yang dibuat dan dikelola oleh CodePipeline. Untuk informasi lebih lanjut, lihat Bekerja dengan Kebijakan Terkelola. Anda harus melampirkan kebijakan ini ke peran IAM yang Anda buat untuk akses, misalnya, peran bernama`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*"
        }
      }
    }
  ]
}
```

Setelah Anda membuat kebijakan ini, buat peran IAM di akun 111222333444 dan lampirkan kebijakan ke peran tersebut. Dalam hubungan kepercayaan peran, Anda harus menambahkan AWS akun yang akan mengambil peran ini. Contoh berikut menunjukkan kebijakan yang memungkinkan pengguna dari *111111111111* AWS akun untuk mengambil peran yang ditentukan dalam akun 111222333444:

JSON

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

```
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Contoh berikut menunjukkan kebijakan yang dibuat di *111111111111* AWS akun yang memungkinkan pengguna untuk mengambil peran yang disebutkan *CrossAccountPipelineViewers* di akun 111222333444:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource":
 "arn:aws:iam::111222333444:role/CrossAccountPipelineViewers"
        }
    ]
}
```

Anda dapat membuat kebijakan IAM untuk membatasi panggilan dan sumber daya yang dapat diakses oleh pengguna di akun Anda, lalu melampirkan kebijakan tersebut ke pengguna administratif Anda. Untuk informasi selengkapnya tentang cara membuat peran IAM dan untuk menjelajahi contoh pernyataan kebijakan IAM CodePipeline, lihat. Contoh kebijakan yang dikelola pelanggan

Menggunakan tag untuk mengontrol akses ke CodePipeline sumber daya

Kondisi dalam pernyataan kebijakan IAM adalah bagian dari sintaks yang Anda gunakan untuk menentukan izin ke sumber daya yang diperlukan oleh tindakan. CodePipeline Menggunakan tanda dalam kondisi adalah salah satu cara untuk mengontrol akses ke sumber daya dan permintaan.

Untuk informasi tentang menandai CodePipeline sumber daya, lihatPenandaan pada sumber daya. Topik ini membahas kontrol akses berbasis tanda.

Saat merancang kebijakan IAM, Anda mungkin menetapkan izin terperinci dengan memberikan akses ke sumber daya tertentu. Saat jumlah sumber daya yang Anda kelola bertambah, tugas ini menjadi lebih sulit. Menandai sumber daya dan menggunakan tanda dalam kondisi pernyataan kebijakan dapat mempermudah tugas ini. Anda memberikan akses secara massal ke sumber daya dengan tag tertentu. Kemudian Anda menerapkan tag ini berulang kali ke sumber daya yang relevan, selama pembuatan atau yang lebih baru.

Tag dapat dilampirkan ke sumber daya atau diteruskan atas permintaan ke layanan yang mendukung penandaan. Di CodePipeline, sumber daya dapat memiliki tag, dan beberapa tindakan dapat menyertakan tag. Saat membuat kebijakan IAM, Anda dapat menggunakan kunci kondisi tag untuk mengontrol:

- Manakah pengguna yang dapat melakukan tindakan pada sumber daya alur, berdasarkan tanda yang telah dimiliki.
- Tanda apa yang dapat diteruskan dalam permintaan tindakan.
- Apakah kunci tanda tertentu dapat digunakan dalam permintaan.

Operator ketentuan string memungkinkan Anda membangun elemen `Condition` yang membatasi akses berdasarkan perbandingan kunci ke nilai string. Anda dapat menambahkan `IfExists` ke akhir nama operator kondisi apa pun kecuali kondisi Null. Anda melakukan ini untuk mengatakan "Jika kunci kebijakan ada dalam konteks permintaan, proses kuncinya seperti yang ditentukan dalam kebijakan. Jika kuncinya tidak ada, evaluasi elemen ketentuan sebagai benar." Misalnya, Anda dapat menggunakan `StringEqualsIfExists` untuk membatasi dengan kunci kondisi yang mungkin tidak ada pada jenis sumber daya lainnya.

Untuk sintaks dan semantik lengkap kunci kondisi tag, lihat Mengontrol Akses Menggunakan Tag. Untuk informasi tambahan tentang kunci kondisi, lihat sumber daya berikut. Contoh CodePipeline kebijakan di bagian ini selaras dengan informasi berikut tentang kunci kondisi dan memperluasnya dengan contoh nuansa CodePipeline seperti penyarangan sumber daya.

- Operator kondisi string
- Layanan AWS yang bekerja dengan IAM
- Sintaks SCP
- Elemen kebijakan IAM JSON: Kondisi

- aws: RequestTag /tag-kunci
- Kunci kondisi untuk CodePipeline

Contoh berikut menunjukkan cara menentukan kondisi tag dalam kebijakan untuk CodePipeline pengguna.

Example 1: Batasi tindakan berdasarkan tanda dalam permintaan

Kebijakan pengguna `AWSCodePipeline_FullAccess` terkelola memberi pengguna izin tak terbatas untuk melakukan CodePipeline tindakan apa pun pada sumber daya apa pun.

Kebijakan berikut membatasi kekuatan ini dan menolak izin pengguna yang tidak sah untuk membuat saluran pipa di mana tag tertentu tercantum dalam permintaan. Untuk melakukan itu, ia menolak tindakan `CreatePipeline` jika permintaan menentukan tag bernama `Project` dengan salah satu nilai `ProjectA` atau `ProjectB`. (Kunci syarat `aws:RequestTag` digunakan untuk mengontrol tanda yang dapat diteruskan dalam permintaan IAM.)

Dalam contoh berikut, maksud kebijakan adalah untuk menolak izin pengguna yang tidak sah untuk membuat pipeline dengan nilai tag yang ditentukan. Namun, membuat pipa membutuhkan akses sumber daya selain pipa itu sendiri (misalnya, tindakan dan tahapan pipa). Karena yang `'Resource'` ditentukan dalam kebijakan adalah`'*'`, kebijakan dievaluasi terhadap setiap sumber daya yang memiliki ARN dan dibuat ketika pipeline sedang dibuat. Sumber daya tambahan ini tidak memiliki kunci kondisi tag, sehingga `StringEquals` pemeriksaan gagal, dan pengguna tidak diberikan kemampuan untuk membuat pipeline apa pun. Untuk mengatasi ini, gunakan operator ketentuan `StringEqualsIfExists`. Dengan cara ini, pengujian hanya terjadi jika terdapat kunci kondisi.

Anda dapat membaca berikut ini sebagai: “Jika sumber daya yang diperiksa memiliki kunci `"RequestTag/Project"` kondisi tag, maka izinkan tindakan hanya jika nilai kunci dimulai dengan`projectA`. Jika sumber daya yang diperiksa tidak memiliki kunci kondisi itu, jangan khawatir tentang hal tersebut.”

Selain itu, kebijakan mencegah pengguna yang tidak sah ini merusak sumber daya dengan menggunakan kunci `aws:TagKeys` kondisi untuk tidak mengizinkan tindakan modifikasi tag menyertakan nilai tag yang sama. Administrator pelanggan harus melampirkan kebijakan IAM ini kepada pengguna administratif yang tidak sah, selain kebijakan pengguna yang dikelola.

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: Batasi tindakan penandaan berdasarkan tag sumber daya

Kebijakan pengguna `AWSCodePipeline_FullAccess` terkelola memberi pengguna izin tak terbatas untuk melakukan CodePipeline tindakan apa pun pada sumber daya apa pun.

Kebijakan berikut membatasi kekuatan ini dan menolak izin pengguna yang tidak sah untuk melakukan tindakan pada pipeline proyek tertentu. Untuk melakukan itu, kebijakan menolak beberapa tindakan jika sumber daya memiliki tanda bernama `Project` dengan salah satu nilai

`ProjectA` atau `ProjectB`. (Kunci syarat `aws:ResourceTag` digunakan untuk mengontrol akses ke sumber daya berdasarkan tanda pada sumber daya tersebut.) Administrator pelanggan harus melampirkan kebijakan IAM ini kepada pengguna IAM yang tidak sah, selain kebijakan pengguna terkelola.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": ["ProjectA", "ProjectB"]
        }
      }
    }
  ]
}
```

Example 3: Izinkan tindakan berdasarkan tag dalam permintaan

Kebijakan berikut memberikan izin kepada pengguna untuk membuat pipeline pengembangan di. CodePipeline

Untuk melakukan itu, memungkinkan tindakan `CreatePipeline` dan `TagResource` jika permintaan menentukan tag bernama `Project` dengan nilai `ProjectA`. Dengan kata lain, satu-satunya kunci tag yang dapat ditentukan adalah`Project`, dan nilainya harus`ProjectA`.

Kunci `aws:RequestTag` kondisi digunakan untuk mengontrol tag mana yang dapat diteruskan dalam permintaan IAM. Syarat `aws:TagKeys` memastikan kunci tanda peka huruf besar dan kecil. Kebijakan ini berguna bagi pengguna atau peran yang tidak memiliki kebijakan pengguna `AWSCodePipeline_FullAccess` terkelola yang dilampirkan. Kebijakan terkelola memberi pengguna izin tak terbatas untuk melakukan CodePipeline tindakan apa pun pada sumber daya apa pun.

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: Batasi tindakan untagging berdasarkan tag sumber daya

Kebijakan pengguna `AWSCodePipeline_FullAccess` terkelola memberi pengguna izin tak terbatas untuk melakukan CodePipeline tindakan apa pun pada sumber daya apa pun.

Kebijakan berikut membatasi kekuatan ini dan menolak izin pengguna yang tidak sah untuk melakukan tindakan pada pipeline proyek tertentu. Untuk melakukan itu, kebijakan menolak beberapa tindakan jika sumber daya memiliki tanda bernama `Project` dengan salah satu nilai `ProjectA` atau `ProjectB`.

Selain itu, kebijakan ini mencegah pengguna yang tidak sah ini merusak sumber daya dengan menggunakan kunci `aws:TagKeys` kondisi untuk tidak mengizinkan tindakan modifikasi tag menghapus tag sepenuhnya. `Project` Administrator pelanggan harus melampirkan kebijakan IAM ini ke pengguna atau peran yang tidak sah, selain kebijakan pengguna yang dikelola.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codepipeline:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

Izin yang diperlukan untuk menggunakan konsol CodePipeline

Untuk digunakan CodePipeline di CodePipeline konsol, Anda harus memiliki set izin minimum dari layanan berikut:

- AWS Identity and Access Management
- Amazon Simple Storage Service

Izin ini memungkinkan Anda menjelaskan AWS sumber daya lain untuk AWS akun Anda.

Bergantung pada layanan lain yang Anda masukkan ke dalam pipeline, Anda mungkin memerlukan izin dari satu atau beberapa hal berikut:

- AWS CodeCommit
- CodeBuild
- CloudFormation
- AWS CodeDeploy
- AWS Elastic Beanstalk

- AWS Lambda
- AWS OpsWorks

Jika Anda membuat kebijakan IAM yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya bagi pengguna dengan kebijakan IAM tersebut. Untuk memastikan bahwa pengguna tersebut masih dapat menggunakan CodePipeline konsol, lampirkan juga kebijakan `AWSCodePipeline_ReadOnlyAccess` terkelola ke pengguna, seperti yang dijelaskan dalamAWS kebijakan terkelola untuk AWS CodePipeline.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan ke AWS CLI atau CodePipeline API.

Izin diperlukan untuk melihat log komputasi di konsol CodePipeline

Untuk melihat log dalam tindakan Perintah di CodePipeline konsol, peran konsol harus memiliki izin. Untuk melihat log di konsol, tambahkan `logs:GetLogEvents` izin ke peran konsol.

Dalam pernyataan kebijakan peran konsol, cakupkan izin ke tingkat pipeline seperti yang ditunjukkan pada contoh berikut.

```
{
    "Effect": "Allow",
    "Action": [
        "Action": "logs:GetLogEvents"
    ],
    "Resource": "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
}
```

AWS kebijakan terkelola untuk AWS CodePipeline

Kebijakan AWS terkelola adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. AWS Kebijakan terkelola dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan AWS terkelola mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan semua pelanggan. AWS Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan AWS terkelola. Jika AWS memperbarui izin yang ditentukan dalam kebijakan AWS terkelola, pembaruan akan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan AWS terkelola saat baru Layanan AWS diluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat Kebijakan terkelola AWS dalam Panduan Pengguna IAM.

> **Important**
>
> AWS mengelola kebijakan `AWSCodePipelineFullAccess` dan `AWSCodePipelineReadOnlyAccess` telah diganti. Gunakan `AWSCodePipeline_FullAccess` dan `AWSCodePipeline_ReadOnlyAccess` kebijakan.

AWS kebijakan terkelola: **`AWSCodePipeline_FullAccess`**

Ini adalah kebijakan yang memberikan akses penuh ke CodePipeline. Untuk melihat dokumen kebijakan JSON di konsol IAM, lihat. AWSCodePipeline_FullAccess

Detail izin

Kebijakan ini mencakup izin berikut.

- `codepipeline`— Memberikan izin untuk. CodePipeline
- `chatbot`— Memberikan izin untuk mengizinkan kepala sekolah mengelola sumber daya di Amazon Q Developer dalam aplikasi obrolan.
- `cloudformation`— Memberikan izin untuk memungkinkan prinsipal mengelola tumpukan sumber daya di. CloudFormation
- `cloudtrail`— Memberikan izin untuk memungkinkan prinsipal mengelola sumber daya logging di. CloudTrail
- `codebuild`— Memberikan izin untuk memungkinkan kepala sekolah mengakses sumber daya build di. CodeBuild

- `codecommit`— Memberikan izin untuk memungkinkan prinsipal mengakses sumber daya sumber di. CodeCommit
- `codedeploy`— Memberikan izin untuk memungkinkan prinsipal mengakses sumber daya penerapan di. CodeDeploy
- `codestar-notifications`— Memberikan izin untuk mengizinkan kepala sekolah mengakses sumber daya di Pemberitahuan. AWS CodeStar
- `ec2`— Memberikan izin untuk mengizinkan penerapan untuk mengelola penyeimbangan beban elastis di CodeCatalyst Amazon EC2.
- `ecr`— Memberikan izin untuk memungkinkan akses ke sumber daya di Amazon ECR.
- `elasticbeanstalk`— Memberikan izin untuk memungkinkan prinsipal mengakses sumber daya di Elastic Beanstalk.
- `iam`— Memberikan izin untuk mengizinkan kepala sekolah mengelola peran dan kebijakan di IAM.
- `lambda`— Memberikan izin untuk mengizinkan kepala sekolah mengelola sumber daya di Lambda.
- `events`— Memberikan izin untuk memungkinkan kepala sekolah mengelola sumber daya di Acara. CloudWatch
- `opsworks`— Memberikan izin untuk memungkinkan kepala sekolah mengelola sumber daya di. AWS OpsWorks
- `s3`— Memberikan izin untuk mengizinkan prinsipal mengelola sumber daya di Amazon S3.
- `sns`— Memberikan izin untuk mengizinkan kepala sekolah mengelola sumber daya notifikasi di Amazon SNS.
- `states`— Memberikan izin untuk memungkinkan kepala sekolah melihat mesin status di. AWS Step Functions Mesin negara terdiri dari kumpulan negara yang mengelola tugas dan transisi antar negara bagian.

Untuk kebijakan, lihat AWSCodePipeline_FullAccess.

AWS kebijakan terkelola: **`AWSCodePipeline_ReadOnlyAccess`**

Ini adalah kebijakan yang memberikan akses hanya-baca ke. CodePipeline Untuk melihat dokumen kebijakan JSON di konsol IAM, lihat. AWSCodePipeline_ReadOnlyAccess

Detail izin

Kebijakan ini mencakup izin berikut.

- `codepipeline`— Memberikan izin untuk tindakan di. CodePipeline
- `codestar-notifications`— Memberikan izin untuk mengizinkan kepala sekolah mengakses sumber daya di Pemberitahuan. AWS CodeStar
- `s3`— Memberikan izin untuk mengizinkan prinsipal mengelola sumber daya di Amazon S3.
- `sns`— Memberikan izin untuk mengizinkan kepala sekolah mengelola sumber daya notifikasi di Amazon SNS.

Untuk kebijakan, lihat AWSCodePipeline_ReadOnlyAccess.

AWS kebijakan terkelola: **`AWSCodePipelineApproverAccess`**

Ini adalah kebijakan yang memberikan izin untuk menyetujui atau menolak tindakan persetujuan manual. Untuk melihat dokumen kebijakan JSON di konsol IAM, lihat. AWSCodePipelineApproverAccess

Detail izin

Kebijakan ini mencakup izin berikut.

- `codepipeline`— Memberikan izin untuk tindakan di. CodePipeline

Untuk kebijakan, lihat AWSCodePipelineApproverAccess.

AWS kebijakan terkelola: **`AWSCodePipelineCustomActionAccess`**

Ini adalah kebijakan yang memberikan izin untuk membuat tindakan kustom di CodePipeline atau mengintegrasikan sumber daya Jenkins untuk tindakan build atau pengujian. Untuk melihat dokumen kebijakan JSON di konsol IAM, lihat. AWSCodePipelineCustomActionAccess

Detail izin

Kebijakan ini mencakup izin berikut.

- `codepipeline`— Memberikan izin untuk tindakan di. CodePipeline

Untuk kebijakan, lihat AWSCodePipelineCustomActionAccess.

CodePipeline kebijakan dan pemberitahuan terkelola

CodePipeline mendukung pemberitahuan, yang dapat memberi tahu pengguna tentang perubahan penting pada saluran pipa. Kebijakan terkelola untuk CodePipeline menyertakan pernyataan kebijakan untuk fungsionalitas notifikasi. Untuk informasi selengkapnya, lihat Apa itu notifikasi?.

Izin yang terkait dengan notifikasi dalam kebijakan terkelola akses penuh

Kebijakan terkelola ini memberikan izin untuk CodePipeline bersama dengan layanan CodeCommit, CodeBuild CodeDeploy, dan AWS CodeStar Pemberitahuan terkait. Kebijakan ini juga memberikan izin yang Anda perlukan untuk bekerja dengan layanan lain yang terintegrasi dengan pipeline Anda, seperti Amazon S3, Elastic Beanstalk, Amazon EC2, dan. CloudTrail CloudFormation Pengguna dengan kebijakan terkelola ini diterapkan juga dapat membuat dan mengelola topik Amazon SNS untuk notifikasi, berlangganan dan berhenti berlangganan pengguna ke topik, mencantumkan topik untuk dipilih sebagai target aturan notifikasi, dan daftar Pengembang Amazon Q di aplikasi obrolan klien yang dikonfigurasi untuk Slack.

Kebijakan terkelola `AWSCodePipeline_FullAccess` mencakup pernyataan berikut untuk mengizinkan akses penuh ke notifikasi.

```
    {
        "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": "*"
    }
```

Izin yang terkait dengan notifikasi dalam kebijakan terkelola hanya-baca

Kebijakan terkelola `AWSCodePipeline_ReadOnlyAccess` mencakup pernyataan berikut untuk mengizinkan akses penuh ke notifikasi. Pengguna dengan kebijakan ini diterapkan dapat melihat pemberitahuan untuk sumber daya, tetapi tidak dapat membuat, mengelola, atau berlangganan.

```
  {
       "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": "*"
   }
```

Untuk informasi selengkapnya tentang IAM dan notifikasi, lihat Identity and Access Management for AWS CodeStar Notifications.

AWS CodePipeline pembaruan kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola CodePipeline sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Riwayat CodePipeline dokumen.

Perubahan	Deskripsi	Date
AWSCodePipeline_FullAccess— Pembaruan kebijakan yang ada	CodePipeline menambahkan izin ke kebijakan ini untuk `ListStacks` mendukung CloudFormation.	Maret 15, 2024

Perubahan	Deskripsi	Date
AWSCodePipeline_FullAccess— Pembaruan kebijakan yang ada	Kebijakan ini diperbarui untuk menambahkan izin untuk Pengembang Amazon Q di aplikasi obrolan. Untuk informasi selengkapnya, lihat CodePipeline kebijakan dan pemberitahuan terkelola.	Juni 21, 2023
AWSCodePipeline_FullAccessdan kebijakan AWSCodePipeline_ReadOnlyAccessterkelola - Pembaruan kebijakan yang ada	CodePipeline menambahkan izin ke kebijakan ini untuk mendukung jenis notifikasi tambahan menggunakan Pengembang Amazon Q di aplikasi obrolan`chatbot:ListMicrosoftTeamsChannelConfigurations` .	16 Mei 2023
AWSCodePipelineFullAccess— Usang	Kebijakan ini telah diganti dengan `AWSCodePipeline_FullAccess` . Setelah 17 November 2022, kebijakan ini tidak dapat dilampirkan ke pengguna, grup, atau peran baru mana pun. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk AWS CodePipeline.	17 November 2022

Perubahan	Deskripsi	Date
AWSCodePipelineReadOnlyAccess— Usang	Kebijakan ini telah diganti dengan `AWSCodePipeline_ReadOnlyAccess` . Setelah 17 November 2022, kebijakan ini tidak dapat dilampirkan ke pengguna, grup, atau peran baru mana pun. Untuk informasi selengkapnya, lihat AWS kebijakan terkelola untuk AWS CodePipeline.	17 November 2022
CodePipeline mulai melacak perubahan	CodePipeline mulai melacak perubahan untuk kebijakan AWS terkelolanya.	12 Maret 2021

Contoh kebijakan yang dikelola pelanggan

Di bagian ini, Anda dapat menemukan contoh kebijakan pengguna yang memberikan izin untuk berbagai CodePipeline tindakan. Kebijakan ini berfungsi saat Anda menggunakan CodePipeline API, AWS SDKs, atau AWS CLI. Saat menggunakan konsol, Anda harus memberikan izin tambahan khusus untuk konsol. Untuk informasi selengkapnya, lihat Izin yang diperlukan untuk menggunakan konsol CodePipeline.

> **Note**
>
> Semua contoh menggunakan Wilayah Barat AS (Oregon) (`us-west-2`) dan berisi akun fiktif. IDs

Contoh

- Contoh 1: Berikan izin untuk mendapatkan status pipa

- Contoh 2: Berikan izin untuk mengaktifkan dan menonaktifkan transisi antar tahapan
- Contoh 3: Berikan izin untuk mendapatkan daftar semua jenis tindakan yang tersedia
- Contoh 4: Berikan izin untuk menyetujui atau menolak tindakan persetujuan manual
- Contoh 5: Berikan izin untuk melakukan polling untuk pekerjaan untuk tindakan kustom
- Contoh 6: Lampirkan atau edit kebijakan untuk integrasi Jenkins dengan AWS CodePipeline
- Contoh 7: Konfigurasikan akses lintas akun ke pipeline

Contoh 1: Berikan izin untuk mendapatkan status pipa

Contoh berikut memberikan izin untuk mendapatkan status pipeline bernama: `MyFirstPipeline`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:GetPipelineState"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline"
        }
    ]
}
```

Contoh 2: Berikan izin untuk mengaktifkan dan menonaktifkan transisi antar tahapan

Contoh berikut memberikan izin untuk menonaktifkan dan mengaktifkan transisi antara semua tahapan dalam pipeline bernama: `MyFirstPipeline`

JSON

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

```
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:DisableStageTransition",
                "codepipeline:EnableStageTransition"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline/*"
        }
    ]
}
```

Untuk memungkinkan pengguna menonaktifkan dan mengaktifkan transisi untuk satu tahap dalam pipeline, Anda harus menentukan tahapannya. Misalnya, untuk memungkinkan pengguna mengaktifkan dan menonaktifkan transisi untuk tahap bernama `Staging` dalam pipeline bernama`MyFirstPipeline`:

```
"Resource": "arn:aws:codepipeline:us-west-2:111222333444:MyFirstPipeline/Staging"
```

Contoh 3: Berikan izin untuk mendapatkan daftar semua jenis tindakan yang tersedia

Contoh berikut memberikan izin untuk mendapatkan daftar semua jenis tindakan yang tersedia yang tersedia untuk pipeline di Wilayah: `us-west-2`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:ListActionTypes"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:actiontype:*"
        }
    ]
}
```

Contoh 4: Berikan izin untuk menyetujui atau menolak tindakan persetujuan manual

Contoh berikut memberikan izin untuk menyetujui atau menolak tindakan persetujuan manual dalam tahapan yang disebutkan dalam pipeline bernama`Staging`: `MyFirstPipeline`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:PutApprovalResult"
            ],
            "Resource": "arn:aws:codepipeline:us-
west-2:111222333444:MyFirstPipeline/Staging/*"
        }
    ]
}
```

Contoh 5: Berikan izin untuk melakukan polling untuk pekerjaan untuk tindakan kustom

Contoh berikut memberikan izin untuk melakukan polling pekerjaan untuk tindakan kustom bernama`TestProvider`, yang merupakan tipe `Test` tindakan dalam versi pertamanya, di semua pipeline:

> **Note**
>
> Pekerja pekerjaan untuk tindakan kustom mungkin dikonfigurasi di bawah AWS akun yang berbeda atau memerlukan peran IAM tertentu agar berfungsi.

JSON

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

```
            "Effect": "Allow",
            "Action": [
                "codepipeline:PollForJobs"
            ],
            "Resource": [
                "arn:aws:codepipeline:us-
west-2:111222333444:actionType:Custom/Test/TestProvider/1"
            ]
        }
    ]
}
```

Contoh 6: Lampirkan atau edit kebijakan untuk integrasi Jenkins dengan AWS CodePipeline

Jika Anda mengonfigurasi pipeline untuk menggunakan Jenkins untuk membangun atau menguji, buat identitas terpisah untuk integrasi tersebut dan lampirkan kebijakan IAM yang memiliki izin minimum yang diperlukan untuk integrasi antara Jenkins dan. CodePipeline Kebijakan ini sama dengan kebijakan yang `AWSCodePipelineCustomActionAccess` dikelola. Contoh berikut menunjukkan kebijakan untuk integrasi Jenkins:

JSON

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

    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codepipeline:AcknowledgeJob",
                "codepipeline:GetJobDetails",
                "codepipeline:PollForJobs",
                "codepipeline:PutJobFailureResult",
                "codepipeline:PutJobSuccessResult"
            ],
            "Resource": "*"
        }
    ]
}
```

Contoh 7: Konfigurasikan akses lintas akun ke pipeline

Anda dapat mengonfigurasi akses ke saluran pipa untuk pengguna dan grup di AWS akun lain. Cara yang disarankan adalah membuat peran di akun tempat pipeline dibuat. Peran tersebut harus memungkinkan pengguna dari AWS akun lain untuk mengambil peran itu dan mengakses pipeline. Untuk informasi selengkapnya, lihat Panduan: Akses Lintas Akun Menggunakan Peran.

Contoh berikut menunjukkan kebijakan di akun 80398EXAMPLE yang memungkinkan pengguna untuk melihat, tetapi tidak mengubah, pipeline yang dinamai `MyFirstPipeline` di konsol. CodePipeline Kebijakan ini didasarkan pada kebijakan yang `AWSCodePipeline_ReadOnlyAccess` dikelola, tetapi karena khusus untuk `MyFirstPipeline` pipeline, kebijakan ini tidak dapat menggunakan kebijakan terkelola secara langsung. Jika Anda tidak ingin membatasi kebijakan ke pipeline tertentu, pertimbangkan untuk menggunakan salah satu kebijakan terkelola yang dibuat dan dikelola oleh CodePipeline. Untuk informasi lebih lanjut, lihat Bekerja dengan Kebijakan Terkelola. Anda harus melampirkan kebijakan ini ke peran IAM yang Anda buat untuk akses, misalnya, peran bernama`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"
        }
    ]
}
```

Setelah Anda membuat kebijakan ini, buat peran IAM di akun 80398EXAMPLE dan lampirkan kebijakan ke peran tersebut. Dalam hubungan kepercayaan peran, Anda harus menambahkan AWS akun yang mengasumsikan peran ini.

Contoh berikut menunjukkan kebijakan yang dibuat di *111111111111* AWS akun yang memungkinkan pengguna untuk mengambil peran yang disebutkan `CrossAccountPipelineViewers` di akun 80398EXAMPLE:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource":
 "arn:aws:iam::111122223333:role/CrossAccountPipelineViewers"
        }
    ]
}
```

AWS CodePipeline contoh kebijakan berbasis sumber daya

Layanan lain, seperti Amazon S3, juga mendukung kebijakan izin berbasis sumber daya. Misalnya, Anda dapat melampirkan kebijakan ke bucket S3 untuk mengelola izin akses ke bucket tersebut. Meskipun CodePipeline tidak mendukung kebijakan berbasis sumber daya, ia menyimpan artefak untuk digunakan dalam saluran pipa di bucket S3 berversi.

Example Untuk membuat kebijakan untuk bucket S3 untuk digunakan sebagai penyimpanan artefak CodePipeline

Anda dapat menggunakan bucket S3 berversi apa pun sebagai penyimpanan artefak. CodePipeline Jika Anda menggunakan wizard Create Pipeline untuk membuat pipeline pertama Anda, bucket S3 ini dibuat untuk memastikan bahwa semua objek yang diunggah ke penyimpanan artefak dienkripsi dan koneksi ke bucket aman. Jika Anda membuat bucket S3 sendiri, sebagai praktik terbaik, pertimbangkan untuk menambahkan kebijakan berikut atau elemennya ke bucket. Dalam kebijakan ini, ARN untuk bucket S3 adalah. `codepipeline-us-east-2-1234567890` Ganti ARN ini dengan ARN untuk bucket S3 Anda:

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
                }
            }
        }
    ]
```

```
}
```

Memecahkan masalah AWS CodePipeline identitas dan akses

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan CodePipeline dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di CodePipeline
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya seorang administrator dan ingin mengizinkan orang lain mengakses CodePipeline
- Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses CodePipeline sumber daya saya

Saya tidak berwenang untuk melakukan tindakan di CodePipeline

Jika Konsol Manajemen AWS memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda.

Contoh kesalahan berikut terjadi ketika pengguna `mateojackson` IAM mencoba menggunakan konsol untuk melihat detail tentang pipeline, tetapi tidak memiliki `codepipeline:GetPipeline` izin.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 codepipeline:GetPipeline on resource: my-pipeline
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk mengizinkan dia mengakses sumber daya `my-pipeline` menggunakan tindakan `codepipeline:GetPipeline`.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak berwenang untuk melakukan `iam:PassRole` tindakan, Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda. Minta orang tersebut untuk memperbarui kebijakan Anda agar Anda dapat memberikan peran ke CodePipeline.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan itu, alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol tersebut untuk melakukan tindakan di CodePipeline. Namun, tindakan ini mengharuskan layanan memiliki izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut ke layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, Mary meminta administrator untuk memperbarui kebijakannya agar dia dapat melakukan tindakan `iam:PassRole`.

Saya seorang administrator dan ingin mengizinkan orang lain mengakses CodePipeline

Untuk memungkinkan orang lain mengakses CodePipeline, Anda harus memberikan izin kepada orang atau aplikasi yang membutuhkan akses. Jika Anda menggunakan AWS IAM Identity Center untuk mengelola orang dan aplikasi, Anda menetapkan set izin kepada pengguna atau grup untuk menentukan tingkat akses mereka. Set izin secara otomatis membuat dan menetapkan kebijakan IAM ke peran IAM yang terkait dengan orang atau aplikasi. Untuk informasi selengkapnya, lihat Set izin di Panduan AWS IAM Identity Center Pengguna.

Jika Anda tidak menggunakan IAM Identity Center, Anda harus membuat entitas IAM (pengguna atau peran) untuk orang atau aplikasi yang membutuhkan akses. Anda kemudian harus melampirkan kebijakan yang memberi mereka izin yang tepat di CodePipeline. Setelah izin diberikan, berikan kredensialnya kepada pengguna atau pengembang aplikasi. Mereka akan menggunakan kredensyal tersebut untuk mengakses. AWSUntuk mempelajari selengkapnya tentang membuat pengguna, grup, kebijakan, dan izin IAM, lihat Identitas dan Kebijakan IAM dan izin di IAM di Panduan Pengguna IAM.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses CodePipeline sumber daya saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis

sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mempelajari apakah CodePipeline mendukung fitur ini, lihatBagaimana AWS CodePipeline bekerja dengan IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

CodePipeline referensi izin

Gunakan tabel berikut sebagai referensi saat menyiapkan kontrol akses dan menulis kebijakan izin yang dapat dilampirkan ke identitas IAM (kebijakan berbasis identitas). Tabel mencantumkan setiap operasi CodePipeline API dan tindakan terkait yang dapat Anda berikan izin untuk melakukan tindakan. Untuk operasi yang mendukung izin tingkat sumber daya, tabel mencantumkan AWS sumber daya yang dapat Anda berikan izin. Anda menentukan tindakan di bidang `Action` kebijakan.

Izin tingkat sumber daya adalah izin yang memungkinkan Anda menentukan sumber daya pengguna mana yang diizinkan untuk melakukan tindakan. AWS CodePipeline memberikan dukungan sebagian untuk izin tingkat sumber daya. Ini berarti bahwa untuk beberapa panggilan AWS CodePipeline API, Anda dapat mengontrol kapan pengguna diizinkan untuk menggunakan tindakan tersebut berdasarkan kondisi yang harus dipenuhi, atau sumber daya mana yang diizinkan untuk digunakan pengguna. Misalnya, Anda dapat memberikan izin kepada pengguna untuk mencantumkan informasi eksekusi pipeline, tetapi hanya untuk pipeline atau pipeline tertentu.

> **Note**
>
> Kolom Resources mencantumkan sumber daya yang diperlukan untuk panggilan API yang mendukung izin tingkat sumber daya. Untuk panggilan API yang tidak mendukung izin tingkat sumber daya, Anda dapat memberikan izin kepada pengguna untuk menggunakannya, tetapi Anda harus menentukan wildcard (*) untuk elemen sumber daya pernyataan kebijakan Anda.

CodePipeline Operasi API dan Izin yang Diperlukan untuk Tindakan

AcknowledgeJob

Aksi: `codepipeline:AcknowledgeJob`

Diperlukan untuk melihat informasi tentang pekerjaan tertentu dan apakah pekerjaan itu telah diterima oleh pekerja pekerjaan. Digunakan untuk tindakan kustom saja.

Resources: Hanya mendukung wildcard (*) di `Resource` elemen kebijakan.

AcknowledgeThirdPartyJob

Aksi: `codepipeline:AcknowledgeThirdPartyJob`

Diperlukan untuk mengonfirmasi bahwa pekerja pekerjaan telah menerima pekerjaan yang ditentukan. Digunakan hanya untuk tindakan mitra.

Resources: Hanya mendukung wildcard (*) di `Resource` elemen kebijakan.

CreateCustomActionType

Aksi: `codepipeline:CreateCustomActionType`

Diperlukan untuk membuat tindakan kustom baru yang dapat digunakan di semua pipeline yang terkait dengan AWS akun. Digunakan untuk tindakan kustom saja.

Sumber Daya:

Tipe Aksi

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versior`

CreatePipeline

Aksi: `codepipeline:CreatePipeline`

Diperlukan untuk membuat pipa.

Sumber Daya:

Pipa

`arn:aws:codepipeline:region:account:pipeline-name`

DeleteCustomActionType

Aksi: `codepipeline:DeleteCustomActionType`

Diperlukan untuk menandai tindakan kustom sebagai dihapus. `PollForJobs`untuk tindakan kustom gagal setelah tindakan ditandai untuk dihapus. Digunakan untuk tindakan kustom saja.

Sumber Daya:

Tipe Aksi

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versio`

DeletePipeline

Aksi: `codepipeline:DeletePipeline`

Diperlukan untuk menghapus pipa.

Sumber Daya:

Pipa

`arn:aws:codepipeline:region:account:pipeline-name`

DeleteWebhook

Action`codepipeline:DeleteWebhook`:

Diperlukan untuk menghapus webhook.

Sumber Daya:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

DeregisterWebhookWithThirdParty

Action`codepipeline:DeregisterWebhookWithThirdParty`:

Sebelum webhook dihapus, diperlukan untuk menghapus koneksi antara webhook yang dibuat oleh CodePipeline dan alat eksternal dengan peristiwa yang akan dideteksi. Saat ini hanya didukung untuk webhook yang menargetkan jenis tindakan. GitHub

Sumber Daya:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

DisableStageTransition

Aksi: `codepipeline:DisableStageTransition`

Diperlukan untuk mencegah artefak dalam pipa dari transisi ke tahap berikutnya dalam pipa.

Sumber Daya:

Pipa

`arn:aws:codepipeline:region:account:pipeline-name`

EnableStageTransition

Aksi: `codepipeline:EnableStageTransition`

Diperlukan untuk mengaktifkan artefak dalam pipa untuk transisi ke tahap dalam pipa.

Sumber Daya:

Pipa

`arn:aws:codepipeline:region:account:pipeline-name`

GetJobDetails

Aksi: `codepipeline:GetJobDetails`

Diperlukan untuk mengambil informasi tentang suatu pekerjaan. Hanya digunakan untuk tindakan kustom.

Sumber Daya: Tidak ada sumber daya yang dibutuhkan.

GetPipeline

Aksi: `codepipeline:GetPipeline`

Diperlukan untuk mengambil struktur, tahapan, tindakan, dan metadata pipa, termasuk ARN pipa.

Sumber Daya:

Pipa

`arn:aws:codepipeline:region:account:pipeline-name`

GetPipelineExecution

Aksi: `codepipeline:GetPipelineExecution`

Diperlukan untuk mengambil informasi tentang eksekusi pipeline, termasuk detail tentang artefak, ID eksekusi pipeline, dan nama, versi, dan status pipeline.

Sumber Daya:

Pipa

`arn:aws:codepipeline:region:account:pipeline-name`

GetPipelineState

Aksi: `codepipeline:GetPipelineState`

Diperlukan untuk mengambil informasi tentang keadaan pipa, termasuk tahapan dan tindakan.

Sumber Daya:

Pipa

`arn:aws:codepipeline:region:account:pipeline-name`

GetThirdPartyJobDetails

Aksi: `codepipeline:GetThirdPartyJobDetails`

Diperlukan untuk meminta rincian pekerjaan untuk tindakan pihak ketiga. Digunakan hanya untuk tindakan mitra.

Resources: Hanya mendukung wildcard (*) di `Resource` elemen kebijakan.

ListActionTypes

Aksi: `codepipeline:ListActionTypes`

Diperlukan untuk membuat ringkasan semua jenis CodePipeline tindakan yang terkait dengan akun Anda.

Sumber Daya:

Tipe Aksi

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

ListPipelineExecutions

Aksi: `codepipeline:ListPipelineExecutions`

Diperlukan untuk membuat ringkasan eksekusi terbaru untuk pipeline.

Sumber Daya:

Pipa

`arn:aws:codepipeline:region:account:pipeline-name`

ListPipelines

Aksi: `codepipeline:ListPipelines`

Diperlukan untuk membuat ringkasan semua pipeline yang terkait dengan akun Anda.

Sumber Daya:

ARN saluran pipa dengan wildcard (izin tingkat sumber daya pada tingkat nama pipa tidak didukung)

`arn:aws:codepipeline:region:account:*`

ListTagsForResource

Aksi: `codepipeline:ListTagsForResource`

Diperlukan untuk mencantumkan tanda untuk sumber daya tertentu.

Sumber Daya:

Tipe Aksi

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/version`

Alur

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

ListWebhooks

Actioncodepipeline:ListWebhooks:

Diperlukan untuk mencantumkan semua webhook di akun untuk Wilayah itu.

Sumber Daya:

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

PollForJobs

Tindakan: `codepipeline:PollForJobs`

Diperlukan untuk mengambil informasi tentang pekerjaan apa pun CodePipeline untuk ditindaklanjuti.

Sumber Daya:

Tipe Aksi

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versio`

PollForThirdPartyJobs

Aksi: `codepipeline:PollForThirdPartyJobs`

Diperlukan untuk menentukan apakah ada pekerjaan pihak ketiga untuk ditindaklanjuti oleh pekerja kerja. Digunakan hanya untuk tindakan mitra.

Resources: Hanya mendukung wildcard (*) di `Resource` elemen kebijakan.

PutActionRevision

Aksi: `codepipeline:PutActionRevision`

Diperlukan untuk melaporkan informasi CodePipeline tentang revisi baru ke sumber.

Sumber Daya:

Tindakan

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

PutApprovalResult

Aksi: `codepipeline:PutApprovalResult`

Diperlukan untuk melaporkan tanggapan atas permintaan persetujuan manual ke CodePipeline. Tanggapan yang valid adalah `Approved` dan`Rejected`.

Sumber Daya:

Tindakan

`arn:aws:codepipeline:region:account:pipeline-name/stage-name/action-name`

> **Note**
>
> Panggilan API ini mendukung izin tingkat sumber daya. Namun, Anda mungkin mengalami kesalahan jika menggunakan konsol IAM atau Policy Generator untuk membuat kebijakan `"codepipeline:PutApprovalResult"` yang menentukan ARN sumber daya. Jika Anda mengalami kesalahan, Anda dapat menggunakan tab JSON di konsol IAM atau CLI untuk membuat kebijakan.

PutJobFailureResult

Aksi: `codepipeline:PutJobFailureResult`

Diperlukan untuk melaporkan kegagalan pekerjaan yang dikembalikan ke pipa oleh pekerja kerja. Digunakan untuk tindakan kustom saja.

Resources: Hanya mendukung wildcard (*) di `Resource` elemen kebijakan.

PutJobSuccessResult

Aksi: `codepipeline:PutJobSuccessResult`

Diperlukan untuk melaporkan keberhasilan suatu pekerjaan yang dikembalikan ke pipa oleh pekerja kerja. Digunakan untuk tindakan kustom saja.

Resources: Hanya mendukung wildcard (*) di `Resource` elemen kebijakan.

PutThirdPartyJobFailureResult

Aksi: `codepipeline:PutThirdPartyJobFailureResult`

Diperlukan untuk melaporkan kegagalan pekerjaan pihak ketiga yang dikembalikan ke pipa oleh pekerja kerja. Digunakan hanya untuk tindakan mitra.

Resources: Hanya mendukung wildcard (*) di `Resource` elemen kebijakan.

PutThirdPartyJobSuccessResult

Aksi: `codepipeline:PutThirdPartyJobSuccessResult`

Diperlukan untuk melaporkan keberhasilan pekerjaan pihak ketiga yang dikembalikan ke pipa oleh pekerja kerja. Digunakan hanya untuk tindakan mitra.

Resources: Hanya mendukung wildcard (*) di `Resource` elemen kebijakan.

PutWebhook

Action`codepipeline:PutWebhook`:

Diperlukan untuk membuat webhook.

Sumber Daya:

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RegisterWebhookWithThirdParty

Action`codepipeline:RegisterWebhookWithThirdParty`:

Sumber Daya:

Setelah webhook dibuat, diperlukan untuk mengonfigurasi pihak ketiga yang didukung untuk memanggil URL webhook yang dihasilkan.

Webhook

`arn:aws:codepipeline:`*`region`*`:`*`account`*`:webhook:`*`webhook-name`*

RetryStageExecution

Aksi: `codepipeline:RetryStageExecution`

Diperlukan untuk melanjutkan eksekusi pipeline dengan mencoba kembali tindakan gagal terakhir dalam satu tahap.

Sumber Daya:

Pipa

`arn:aws:codepipeline:region:account:pipeline-name`

StartPipelineExecution

Aksi: `codepipeline:StartPipelineExecution`

Diperlukan untuk memulai pipeline yang ditentukan (khususnya, untuk mulai memproses komit terbaru ke lokasi sumber yang ditentukan sebagai bagian dari pipeline).

Sumber Daya:

Pipa

`arn:aws:codepipeline:region:account:pipeline-name`

TagResource

Aksi: `codepipeline:TagResource`

Diperlukan untuk menandai sumber daya yang ditentukan.

Sumber Daya:

Tipe Aksi

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versio`

Alur

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UntagResource

Aksi: `codepipeline:UntagResource`

Diperlukan untuk menandai sumber daya yang ditentukan.

Sumber Daya:

Tipe Aksi

`arn:aws:codepipeline:region:account:actiontype:owner/category/provider/versio`

Alur

`arn:aws:codepipeline:region:account:pipeline-name`

Webhook

`arn:aws:codepipeline:region:account:webhook:webhook-name`

UpdatePipeline

Aksi: `codepipeline:UpdatePipeline`

Diperlukan untuk memperbarui pipeline tertentu dengan suntingan atau perubahan strukturnya.

Sumber Daya:

Pipa

`arn:aws:codepipeline:region:account:pipeline-name`

Kelola peran CodePipeline layanan

Peran CodePipeline layanan dikonfigurasi dengan satu atau beberapa kebijakan yang mengontrol akses ke AWS sumber daya yang digunakan oleh pipeline. Anda mungkin ingin melampirkan kebijakan lainnya ke peran ini, mengedit kebijakan yang dilampirkan pada peran, atau mengonfigurasi kebijakan untuk peran layanan lainnya AWS. Anda mungkin juga ingin melampirkan kebijakan ke peran saat mengonfigurasi akses lintas akun ke pipeline Anda.

Important

Memodifikasi pernyataan kebijakan atau melampirkan kebijakan lain ke peran tersebut dapat mencegah jaringan pipa Anda berfungsi. Pastikan Anda memahami implikasinya sebelum memodifikasi peran layanan dengan CodePipeline cara apa pun. Pastikan Anda menguji pipeline Anda setelah Anda membuat perubahan apa pun pada peran layanan.

Note

Di konsol, peran layanan yang dibuat sebelum September 2018 dibuat dengan namaoneClick_AWS-CodePipeline-Service_*ID-Number*.

Peran layanan yang dibuat setelah September 2018 menggunakan format nama peran layananAWSCodePipelineServiceRole-*Region*-*Pipeline_Name*. Misalnya, untuk pipeline bernama MyFirstPipeline ineu-west-2, konsol menamai peran dan kebijakan tersebutAWSCodePipelineServiceRole-eu-west-2-MyFirstPipeline.

CodePipeline kebijakan peran layanan

Pernyataan kebijakan peran CodePipeline layanan berisi izin minimum untuk mengelola saluran pipa. Anda dapat mengedit pernyataan peran layanan untuk menghapus atau menambahkan akses ke sumber daya yang tidak Anda gunakan. Lihat referensi tindakan yang sesuai untuk CodePipeline penggunaan izin minimum yang diperlukan untuk setiap tindakan.

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**
>
> Dalam kebijakan, izin berikut diperlukan saat objek S3 di bucket sumber Anda memiliki tag di dalamnya:
>
> ```
> s3:PutObjectTagging
> s3:GetObjectTagging
> s3:GetObjectVersionTagging
> ```

Hapus izin dari peran CodePipeline layanan

Anda dapat mengedit pernyataan peran layanan untuk menghapus akses ke sumber daya yang tidak Anda gunakan. Misalnya, jika tidak ada pipeline yang menyertakan Elastic Beanstalk, Anda dapat mengedit pernyataan kebijakan untuk menghapus bagian yang memberikan akses ke sumber daya Elastic Beanstalk.

Demikian pula, jika tidak ada pipeline yang disertakan CodeDeploy, Anda dapat mengedit pernyataan kebijakan untuk menghapus bagian yang memberikan akses ke CodeDeploy sumber daya:

```
    {
    "Action": [
```

```
        "codedeploy:CreateDeployment",
        "codedeploy:GetApplicationRevision",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:RegisterApplicationRevision"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
```

Menambahkan izin ke peran CodePipeline layanan

Anda harus memperbarui pernyataan kebijakan peran layanan dengan izin untuk pernyataan kebijakan peran layanan default yang Layanan AWS belum disertakan dalam pernyataan kebijakan peran layanan default sebelum dapat menggunakannya di pipeline.

Hal ini sangat penting jika peran layanan yang Anda gunakan untuk pipeline Anda dibuat sebelum dukungan ditambahkan ke CodePipeline untuk. Layanan AWS

Tabel berikut menunjukkan kapan dukungan ditambahkan untuk lainnya Layanan AWS.

Layanan AWS	CodePipeline tanggal dukungan
CodePipeline memanggil dukungan tindakan ditambahkan. Lihat Izin kebijakan peran layanan untuk tindakan CodePipeline pemanggilan.	Maret 14, 2025
EC2dukungan tindakan ditambahkan. Lihat Izin kebijakan peran layanan untuk tindakan penerapan EC2.	Februari 21, 2025
EKSdukungan tindakan ditambahkan. Lihat Izin kebijakan peran layanan.	Februari 20, 2025
Dukungan `ECRBuildAndPublish` tindakan Amazon Elastic Container Registry ditambahkan. Lihat Izin peran layanan: tindakan `ECRBuildAndPublish` .	November 22, 2024

Layanan AWS	CodePipeline tanggal dukungan
Dukungan `InspectorScan` tindakan Amazon Inspector ditambahkan. Lihat Izin peran layanan: tindakan `InspectorScan`.	November 22, 2024
Dukungan tindakan perintah ditambahkan. Lihat Izin peran layanan: Tindakan perintah.	Oktober 03, 2024
CloudFormation dukungan tindakan ditambahkan. Lihat Izin peran layanan: tindakan `CloudFormationStackSet` dan Izin peran layanan: tindakan `CloudFormationStackInstances`.	30 Desember 2020
CodeCommit dukungan aksi format artefak keluaran klon penuh ditambahkan. Lihat Izin peran layanan: tindakan CodeCommit.	11 November 2020
CodeBuild batch membangun dukungan tindakan ditambahkan. Lihat Izin peran layanan: tindakan CodeCommit.	30 Juli 2020
AWS AppConfig dukungan tindakan ditambahkan. Lihat Izin peran layanan: tindakan `AppConfig`.	22 Juni 2020
AWS Step Functions dukungan tindakan ditambahkan. Lihat Izin peran layanan: tindakan `StepFunctions`.	27 Mei 2020
AWS CodeStar Dukungan tindakan koneksi ditambahkan. Lihat Izin peran layanan: tindakan CodeConnections.	18 Desember 2019
Dukungan tindakan penerapan S3 ditambahkan. Lihat Izin peran layanan: Tindakan penerapan S3.	16 Januari 2019

Layanan AWS	CodePipeline tanggal dukungan
Dukungan `CodeDeployToECS` tindakan tindakan ditambahkan. Lihat Izin peran layanan: tindakan `CodeDeployToECS` .	27 November 2018
Dukungan tindakan Amazon ECR ditambahkan. Lihat Izin peran layanan: Tindakan Amazon ECR.	27 November 2018
Dukungan tindakan Service Catalog ditambahkan. Lihat Izin peran layanan: Tindakan Service Catalog.	16 Oktober 2018
AWS Device Farm dukungan tindakan ditambahkan. Lihat Izin peran layanan: tindakan AWS Device Farm.	19 Juli 2018
Dukungan tindakan Amazon ECS ditambahkan. Lihat Izin peran layanan: Tindakan standar Amazon ECS.	12 Desember 2017/Pembaruan untuk memilih untuk menandai otorisasi pada 21 Juli 2017
CodeCommit dukungan tindakan ditambahkan. Lihat Izin peran layanan: tindakan CodeCommit.	18 April 2016
AWS OpsWorks dukungan tindakan ditambahkan. Lihat Izin peran layanan: tindakan AWS OpsWorks.	Juni 2, 2016
CloudFormation dukungan tindakan ditambahkan. Lihat Izin peran layanan: tindakan CloudFormation.	Selasa, 3 November 2016
AWS CodeBuild dukungan tindakan ditambahkan. Lihat Izin peran layanan: tindakan CodeBuild.	1 Desember 2016

Layanan AWS	CodePipeline tanggal dukungan
Dukungan aksi Elastic Beanstalk ditambahkan. Lihat Izin peran layanan: tindakan `ElasticBeanstalk` penerapan.	Peluncuran layanan awal
CodeDeploy dukungan tindakan ditambahkan. Lihat Izin peran layanan: tindakan AWS CodeDeploy.	Peluncuran layanan awal
Dukungan tindakan sumber S3 ditambahkan. Lihat Izin peran layanan: Tindakan sumber S3.	Peluncuran layanan awal

Ikuti langkah-langkah berikut untuk menambahkan izin untuk layanan yang didukung:

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di konsol IAM, di panel navigasi, pilih Peran, lalu pilih `AWS-CodePipeline-Service` peran Anda dari daftar peran.
3. Pada tab Izin, di Kebijakan sebaris, di baris kebijakan peran layanan Anda, pilih Edit Kebijakan.
4. Tambahkan izin yang diperlukan di kotak Dokumen kebijakan.

> **Note**
>
> Saat Anda membuat kebijakan IAM, ikuti saran keamanan standar untuk memberikan hak istimewa paling sedikit—yaitu, hanya memberikan izin yang diperlukan untuk melakukan tugas. Beberapa panggilan API mendukung izin berbasis sumber daya dan memungkinkan akses dibatasi. Misalnya, dalam hal ini, untuk membatasi izin saat memanggil `DescribeTasks` dan`ListTasks`, Anda dapat mengganti karakter wildcard (*) dengan ARN sumber daya atau dengan ARN sumber daya yang berisi karakter wildcard (*). Untuk informasi selengkapnya tentang membuat kebijakan yang memberikan akses hak istimewa paling sedikit, lihat. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege

5. Pilih Kebijakan tinjau untuk memastikan kebijakan tidak mengandung kesalahan. Jika kebijakan bebas dari kesalahan, pilih Terapkan kebijakan.

Penebangan dan pemantauan di CodePipeline

Anda dapat menggunakan fitur masuk AWS untuk menentukan tindakan yang telah diambil pengguna di akun Anda dan sumber daya yang digunakan. File log menunjukkan:

- Waktu dan tanggal tindakan.
- Alamat IP sumber untuk suatu tindakan.
- Tindakan mana yang gagal karena izin yang tidak memadai.

Fitur pencatatan tersedia dalam hal berikut Layanan AWS:

- AWS CloudTrail dapat digunakan untuk mencatat panggilan AWS API dan peristiwa terkait yang dibuat oleh atau atas nama Akun AWS. Untuk informasi selengkapnya, lihat Pencatatan panggilan CodePipeline API dengan AWS CloudTrail.
- Amazon CloudWatch Events dapat digunakan untuk memantau AWS Cloud sumber daya Anda dan aplikasi yang Anda jalankan AWS. Anda dapat membuat peringatan di Amazon CloudWatch Events berdasarkan metrik yang Anda tentukan. Untuk informasi selengkapnya, lihat Memantau CodePipeline peristiwa.

Validasi kepatuhan untuk AWS CodePipeline

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat Dokumentasi AWS Keamanan.

Ketahanan di AWS CodePipeline

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung

dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

Keamanan infrastruktur di AWS CodePipeline

Sebagai layanan terkelola, AWS CodePipeline dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk mendesain AWS lingkungan Anda menggunakan praktik terbaik untuk keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja yang AWS Diarsiteksikan dengan Baik Pilar Keamanan.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses CodePipeline melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Pengangkutan (TLS). Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Sandi cocok dengan sistem kerahasiaan maju sempurna (perfect forward secrecy, PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Praktik terbaik keamanan

CodePipeline menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau tidak memadai untuk lingkungan Anda, perlakukan itu sebagai pertimbangan yang bermanfaat, bukan sebagai resep.

Anda menggunakan enkripsi dan otentikasi untuk repositori sumber yang terhubung ke pipeline Anda. Ini adalah praktik CodePipeline terbaik untuk keamanan:

- Jika Anda membuat konfigurasi pipeline atau tindakan yang perlu menyertakan rahasia, seperti token atau kata sandi, jangan masukkan rahasia secara langsung dalam konfigurasi tindakan, atau

nilai default variabel yang ditentukan pada level pipeline atau CloudFormation konfigurasi, karena informasi akan ditampilkan di log. Gunakan Secrets Manager untuk mengatur dan menyimpan rahasia, lalu gunakan rahasia yang direferensikan dalam konfigurasi pipeline dan tindakan, seperti yang dijelaskan dalamGunakan AWS Secrets Manager untuk melacak kata sandi database atau kunci API pihak ketiga.

- Jika Anda membuat pipeline yang menggunakan bucket sumber S3, konfigurasikan enkripsi sisi server untuk artefak yang disimpan di Amazon S3 dengan mengelola, seperti yang dijelaskan dalam. CodePipeline AWS KMS keysKonfigurasikan enkripsi sisi server untuk artefak yang disimpan di Amazon S3 CodePipeline
- Jika Anda menggunakan penyedia tindakan Jenkins, saat Anda menggunakan penyedia build Jenkins untuk tindakan build atau pengujian pipeline Anda, instal Jenkins pada instans EC2 dan konfigurasikan profil instans EC2 terpisah. Pastikan bahwa profil instance memberi Jenkins hanya AWS izin yang diperlukan untuk melakukan tugas untuk proyek Anda, seperti mengambil file dari Amazon S3. Untuk mempelajari cara membuat peran untuk profil instans Jenkins Anda, lihat langkah-langkahnya. Buat peran IAM untuk digunakan untuk integrasi Jenkins

CodePipeline referensi struktur pipa

Anda dapat menggunakannya CodePipeline untuk menyusun CI/CD alur langkah otomatis yang menyelesaikan tugas yang membangun, menguji, dan menerapkan kode sumber aplikasi Anda. Bagian referensi ini memberikan detail tentang struktur dan parameter JSON di pipeline Anda. Untuk daftar konsep tingkat tinggi yang menggambarkan bagaimana jaringan pipa digunakan, lihat. CodePipeline konsep

- Saat membuat pipeline, Anda memilih tindakan dan penyedia sumber yang tersedia, seperti bucket S3, repositori, CodeCommit repositori Bitbucket, atau GitHub repositori yang berisi kode sumber Anda dan memulai pipeline saat Anda melakukan perubahan kode sumber. Bagian referensi ini memberikan informasi referensi tentang sumber yang tersedia untuk pipeline Anda. Untuk informasi selengkapnya tentang cara bekerja dengan tindakan sumber, lihatMulai pipa di CodePipeline.
- Anda dapat memilih tindakan dan penyedia pengujian, pembuatan, dan penerapan yang ingin Anda sertakan secara otomatis saat pipeline berjalan. Bagian referensi ini memberikan informasi referensi tentang tindakan yang tersedia dan bagaimana kesesuaiannya dengan JSON pipeline Anda.
- Pipeline Anda yang sudah selesai akan terdiri dari tahap sumber bersama dengan tahapan tambahan di mana Anda mengonfigurasi tindakan untuk menerapkan dan menguji aplikasi Anda. Untuk contoh konseptual DevOps pipeline yang menerapkan aplikasi Anda, lihat. DevOps contoh pipa

Secara default, setiap pipeline yang berhasil Anda buat AWS CodePipeline memiliki struktur yang valid. Namun, jika Anda membuat atau mengedit file JSON secara manual untuk membuat pipeline atau memperbarui pipeline dari AWS CLI, Anda mungkin secara tidak sengaja membuat struktur yang tidak valid. Referensi berikut dapat membantu Anda lebih memahami persyaratan untuk struktur pipa Anda dan cara memecahkan masalah. Lihat kendala diKuota di AWS CodePipeline, yang berlaku untuk semua jaringan pipa.

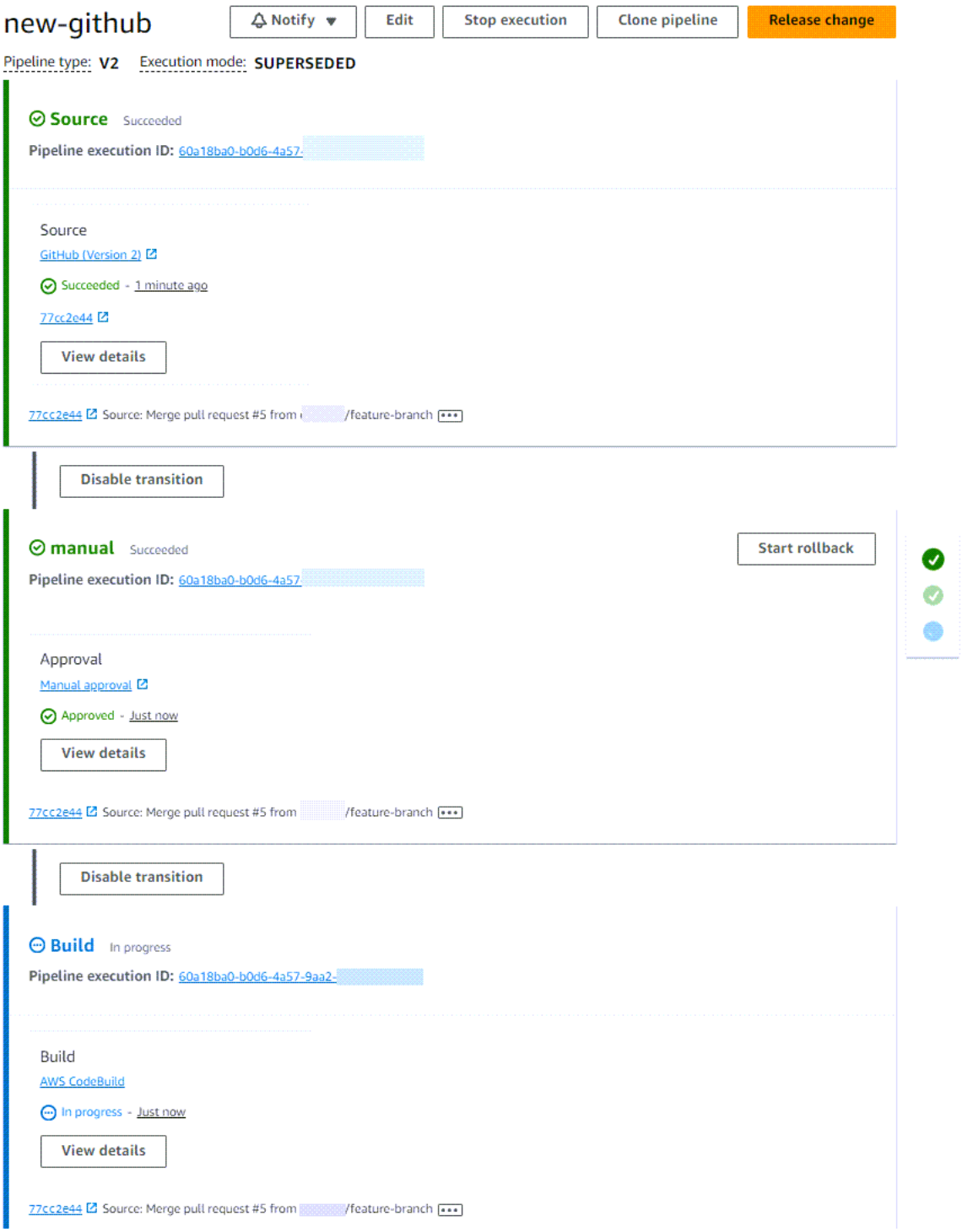
Bagian berikut memberikan parameter tingkat tinggi dan posisinya dalam struktur pipa. Persyaratan struktur pipa dirinci di setiap bagian untuk jenis komponen pipa berikut:

- Referensi lapangan untuk Deklarasi pipa
- Referensi lapangan untuk Deklarasi tahap

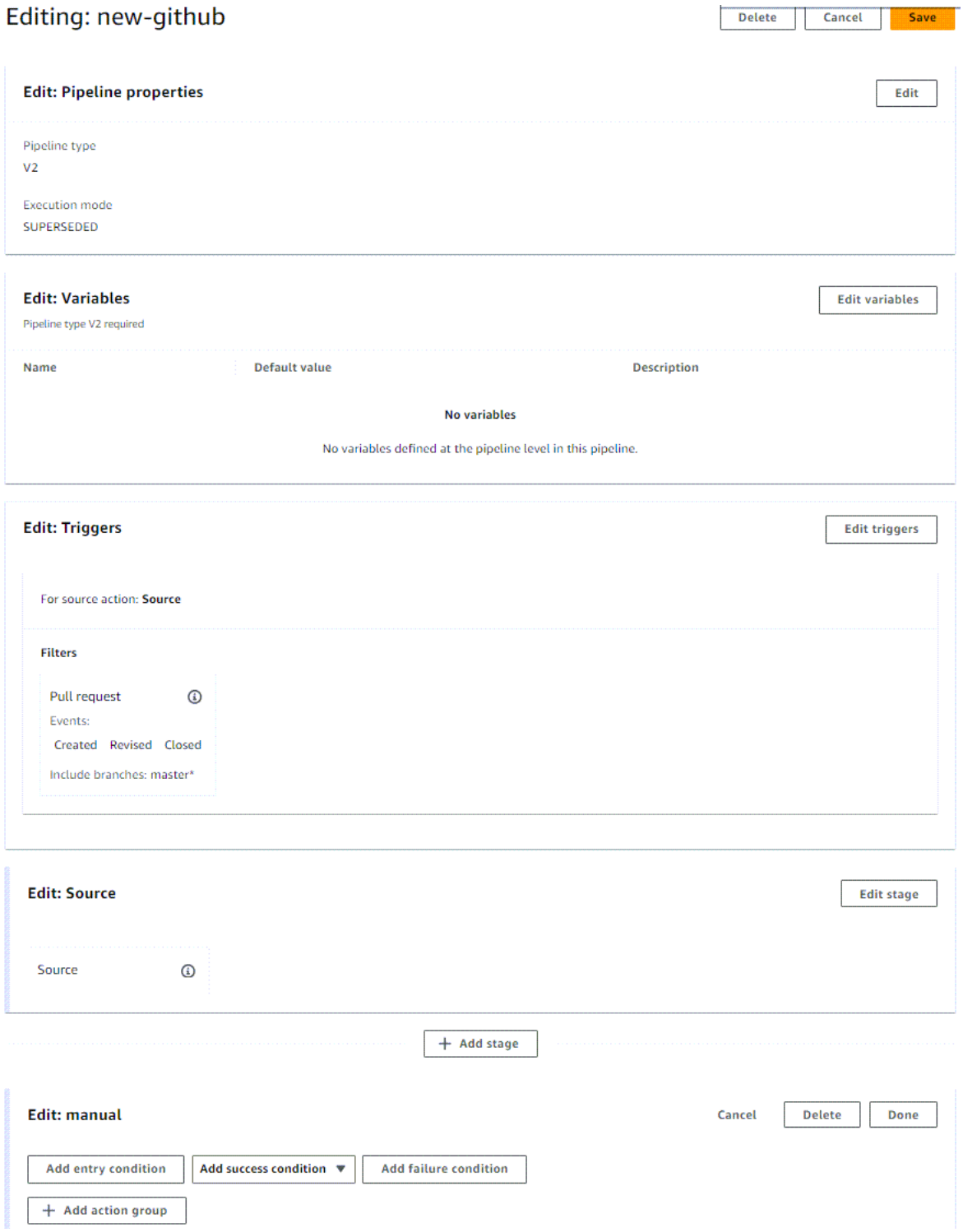
- Referensi lapangan untuk Deklarasi tindakan
- Daftar Penyedia tindakan yang valid di CodePipeline berdasarkan jenis tindakan
- Referensi untuk Pengaturan yang valid untuk `PollForSourceChanges` parameter
- Referensi untuk Artefak input dan output yang valid untuk setiap jenis tindakan
- Daftar tautan untuk Parameter konfigurasi yang valid untuk setiap jenis penyedia

Untuk informasi selengkapnya, lihat PipelineDeclarationobjek di Panduan CodePipeline API.

Contoh tampilan konsol pipeline berikut menunjukkan pipeline bernama new-github, tahapan bernama, dan `Source` `manualBuild`, dan tindakan dari GitHub (melalui GitHub Aplikasi), persetujuan manual, dan penyedia tindakan. CodeBuild



Mode pengeditan pipeline, jika dilihat di diagram konsol, memungkinkan Anda mengedit penggantian sumber, pemicu, dan tindakan seperti yang ditunjukkan pada contoh berikut.



Topik

- Deklarasi pipa
- Deklarasi tahap
- Deklarasi tindakan

- Penyedia tindakan yang valid di CodePipeline
- Pengaturan yang valid untuk PollForSourceChanges parameter
- Artefak input dan output yang valid untuk setiap jenis tindakan
- Parameter konfigurasi yang valid untuk setiap jenis penyedia

Deklarasi pipa

Tingkat pipa dan metadata pipa memiliki struktur dasar yang mencakup parameter dan sintaks berikut. Parameter pipa mewakili struktur tindakan dan tahapan yang harus dilakukan dalam pipa.

Untuk informasi selengkapnya, lihat PipelineDeclarationobjek di Panduan CodePipeline API.

Contoh berikut menunjukkan tingkat pipa dan metadata struktur pipa di JSON dan YAMAL untuk pipa tipe 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

Nama dari alur. Saat Anda mengedit atau memperbarui pipeline, nama pipeline tidak dapat diubah.

> **Note**
>
> Jika Anda ingin mengganti nama pipeline yang ada, Anda dapat menggunakan perintah `get-pipeline` CLI untuk membuat file JSON yang berisi struktur pipeline Anda. Anda kemudian dapat menggunakan `create-pipeline` perintah CLI untuk membuat pipeline dengan struktur itu dan memberinya nama baru.

`roleArn`

ARN IAM untuk peran layanan, seperti arn:aws:iam: CodePipeline :80398Example:Role/ _Service_Role. CodePipeline

Untuk menggunakan konsol untuk melihat peran layanan pipeline ARN, bukan struktur JSON, pilih pipeline Anda di konsol, lalu pilih Pengaturan. Di bawah tab Umum, bidang ARN peran Layanan ditampilkan.

`artifactStore`ATAU `artifactStores`

`artifactStore`Bidang berisi jenis bucket artefak dan lokasi untuk pipeline dengan semua tindakan di AWS Wilayah yang sama. Jika Anda menambahkan tindakan di Wilayah yang berbeda dari pipeline, `artifactStores` pemetaan akan digunakan untuk mencantumkan bucket artefak untuk setiap AWS Wilayah tempat tindakan dijalankan. Saat membuat atau mengedit pipeline, Anda harus memiliki bucket artefak di Wilayah pipeline dan kemudian Anda harus memiliki satu bucket artefak per Wilayah tempat Anda berencana untuk menjalankan suatu tindakan.

> **Note**
>
> Dalam struktur pipa, Anda harus memasukkan salah satu `artifactStore` atau `artifactStores` dalam pipa Anda, tetapi Anda tidak dapat menggunakan keduanya. Jika Anda membuat tindakan lintas wilayah di alur Anda, Anda harus menggunakan `artifactStores`.

Contoh berikut menunjukkan struktur dasar untuk pipeline dengan tindakan lintas wilayah yang menggunakan `artifactStores` parameter:

```
    "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`

Jenis lokasi untuk bucket artefak, ditentukan sebagai Amazon S3.

`location`

Nama bucket Amazon S3 dibuat secara otomatis untuk Anda saat pertama kali membuat pipeline menggunakan konsol, seperti codepipeline-us-east -2-1234567890, atau bucket Amazon S3 apa pun yang Anda berikan untuk tujuan ini

`stages`

Parameter ini berisi nama setiap tahap dalam pipa. Untuk informasi selengkapnya tentang parameter dan sintaks pada tingkat tahap struktur pipeline, lihat StageDeclarationobjek di Panduan CodePipeline API.

Struktur pipa untuk tahapan memiliki persyaratan sebagai berikut:

- Pipa harus mengandung setidaknya dua tahap.
- Tahap pertama dari pipa harus berisi setidaknya satu tindakan sumber. Ini dapat berisi tindakan sumber saja.
- Hanya tahap pertama dari pipa yang dapat berisi tindakan sumber.
- Setidaknya satu tahap di setiap pipeline harus berisi tindakan yang bukan merupakan tindakan sumber.

- Semua nama panggung dalam pipeline harus unik.
- Nama panggung tidak dapat diedit di CodePipeline konsol. Jika Anda mengedit nama panggung dengan menggunakan AWS CLI, dan tahap berisi tindakan dengan satu atau beberapa parameter rahasia (seperti OAuth token), nilai parameter rahasia tersebut tidak dipertahankan. Anda harus memasukkan nilai parameter secara manual (yang ditutupi oleh empat tanda bintang di JSON yang dikembalikan oleh AWS CLI) dan memasukkannya ke dalam struktur JSON.

> **Important**
>
> Pipa yang tidak aktif selama lebih dari 30 hari akan menonaktifkan pemungutan suara untuk pipa. Untuk informasi lebih lanjut, lihat pollingDisabledAtdi referensi struktur pipa. Untuk langkah-langkah memigrasikan pipeline Anda dari polling ke deteksi perubahan berbasis peristiwa, lihat Mengubah Metode Deteksi.

`version`

Nomor versi pipeline dibuat dan diperbarui secara otomatis setiap kali Anda memperbarui pipeline.

`executionMode`

Anda dapat mengatur mode eksekusi pipeline sehingga Anda dapat menentukan perilaku pipeline untuk menjalankan berturut-turut, seperti antrian, menggantikan, atau menjalankan dalam mode paralel. Untuk informasi selengkapnya, lihat Mengatur atau mengubah mode eksekusi pipa.

> **Important**
>
> Untuk saluran pipa dalam mode PARALLEL, rollback panggung tidak tersedia. Demikian pula, kondisi kegagalan dengan jenis hasil rollback tidak dapat ditambahkan ke pipa mode PARALLEL.

`pipelineType`

Jenis pipa menentukan struktur dan fitur yang tersedia dalam pipa, seperti untuk pipa tipe V2. Untuk informasi selengkapnya, lihat Jenis pipa.

variables

Variabel pada tingkat pipa didefinisikan ketika pipa dibuat dan diselesaikan pada waktu proses pipa. Untuk informasi selengkapnya, lihat Referensi variabel. Untuk tutorial dengan variabel tingkat pipa yang dilewatkan pada saat eksekusi pipeline, lihat. Tutorial: Gunakan variabel tingkat pipa

triggers

Pemicu memungkinkan Anda mengonfigurasi pipeline untuk memulai pada jenis peristiwa tertentu atau jenis peristiwa yang difilter, seperti saat perubahan pada cabang tertentu atau permintaan tarik terdeteksi. Pemicu dapat dikonfigurasi untuk tindakan sumber dengan koneksi yang menggunakan `CodeStarSourceConnection` tindakan di CodePipeline, seperti, Bitbucket GitHub, dan. GitLab Untuk informasi selengkapnya tentang tindakan sumber yang menggunakan koneksi, lihatTambahkan penyedia sumber pihak ketiga ke saluran pipa menggunakan CodeConnections.

Untuk informasi lebih lanjut dan contoh lebih rinci, lihatOtomatiskan saluran pipa awal menggunakan pemicu dan penyaringan.

Untuk pemfilteran, pola ekspresi reguler dalam format glob didukung seperti yang dijelaskan dalam. Bekerja dengan pola glob dalam sintaks

> **Note**
>
> Tindakan sumber CodeCommit dan S3 memerlukan sumber daya deteksi perubahan yang dikonfigurasi (EventBridge aturan) atau menggunakan opsi untuk melakukan polling repositori untuk perubahan sumber. Untuk pipeline dengan Bitbucket, GitHub, atau tindakan sumber Server GitHub Perusahaan, Anda tidak perlu menyiapkan webhook atau default untuk polling. Tindakan koneksi mengelola deteksi perubahan untuk Anda.

> **Important**
>
> Pipa yang tidak aktif selama lebih dari 30 hari akan menonaktifkan pemungutan suara untuk pipa. Untuk informasi lebih lanjut, lihat pollingDisabledAtdi referensi struktur pipa. Untuk langkah-langkah memigrasikan pipeline Anda dari polling ke deteksi perubahan berbasis peristiwa, lihat Mengubah Metode Deteksi.

gitConfigurationbidang

Konfigurasi Git untuk pemicu, termasuk jenis peristiwa dan parameter apa pun untuk memfilter berdasarkan cabang, jalur file, tag, atau peristiwa permintaan tarik.

Bidang dalam struktur JSON didefinisikan sebagai berikut:

- `sourceActionName`: Nama tindakan sumber pipeline dengan konfigurasi Git.
- `push`: Dorong acara dengan pemfilteran. Peristiwa ini menggunakan operasi OR antara filter push yang berbeda dan operasi AND di dalam filter.
  - `branches`: Cabang untuk disaring. Cabang menggunakan operasi AND antara termasuk dan tidak termasuk.
    - `includes`: Pola untuk memfilter untuk cabang yang akan disertakan. Termasuk menggunakan operasi OR.
    - `excludes`: Pola untuk memfilter untuk cabang yang akan dikecualikan. Tidak termasuk menggunakan operasi OR.
  - `filePaths`: Nama jalur file untuk difilter.
    - `includes`: Pola untuk memfilter untuk jalur file yang akan disertakan. Termasuk menggunakan operasi OR.
    - `excludes`: Pola untuk memfilter untuk jalur file yang akan dikecualikan. Tidak termasuk menggunakan operasi OR.
  - `tags`: Nama tag untuk difilter.
    - `includes`: Pola untuk memfilter tag yang akan disertakan. Termasuk menggunakan operasi OR.
    - `excludes`: Pola untuk memfilter tag yang akan dikecualikan. Tidak termasuk menggunakan operasi OR.
- `pullRequest`: Tarik peristiwa permintaan dengan pemfilteran pada peristiwa permintaan tarik dan filter permintaan tarik.
  - `events`: Filter pada acara permintaan tarik terbuka, diperbarui, atau ditutup seperti yang ditentukan.
  - `branches`: Cabang untuk disaring. Cabang menggunakan operasi AND antara termasuk dan tidak termasuk.
    - `includes`: Pola untuk memfilter untuk cabang yang akan disertakan. Termasuk menggunakan operasi OR.

- `excludes`: Pola untuk memfilter untuk cabang yang akan dikecualikan. Tidak termasuk menggunakan operasi OR.

- `filePaths`: Nama jalur file untuk difilter.
  - `includes`: Pola untuk memfilter untuk jalur file yang akan disertakan. Termasuk menggunakan operasi OR.
  - `excludes`: Pola untuk memfilter untuk jalur file yang akan dikecualikan. Tidak termasuk menggunakan operasi OR.

Berikut ini adalah contoh konfigurasi pemicu untuk jenis acara permintaan push dan pull.

```
"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"
                            ]
                        }
                    }
                ]
            }
        }
    ],
```

pushBidang jenis acara untuk menyertakan dan mengecualikan

Menyertakan dan mengecualikan perilaku untuk level bidang konfigurasi Git untuk jenis acara push ditampilkan dalam daftar berikut:

```
push (OR operation is used between push and pullRequest or multiples)
```

```
    filePaths (AND operation is used between filePaths, branches, and tags)
        includes (AND operation is used between includes and excludes)
            **/FILE.md, **/FILE2 (OR operation is used between file path names)
        excludes (AND operation is used between includes and excludes)
            **/FILE.md, **/FILE2 (OR operation is used between file path names)
    branches (AND operation is used between filePaths, branches, and tags)
        includes (AND operation is used between includes and excludes)
            BRANCH/**", "BRANCH2/** (OR operation is used between branch names)
        excludes (AND operation is used between includes and excludes)
            BRANCH/**", "BRANCH2/** (OR operation is used between branch names)
    tags (AND operation is used between filePaths, branches, and tags)
         includes (AND operation is used between includes and excludes)
            TAG/**", "TAG2/** (OR operation is used between tag names)
         excludes (AND operation is used between includes and excludes)
            TAG/**", "TAG2/** (OR operation is used between tag names)
```

`pull request`Bidang jenis acara untuk menyertakan dan mengecualikan

Sertakan dan kecualikan perilaku untuk level bidang konfigurasi Git untuk jenis peristiwa permintaan tarik ditampilkan dalam daftar berikut:

```
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`

Bidang metadata pipa berbeda dari struktur pipa dan tidak dapat diedit. Saat Anda memperbarui pipeline, tanggal di bidang `updated` metadata berubah secara otomatis.

`pipelineArn`

Nama Sumber Daya Amazon (ARN) dari pipeline.

Untuk menggunakan konsol untuk melihat ARN pipeline alih-alih struktur JSON, pilih pipeline Anda di konsol, lalu pilih Pengaturan. Di bawah tab Umum, bidang ARN Pipeline ditampilkan.

`created`

Tanggal dan waktu ketika pipa dibuat.

`updated`

Tanggal dan waktu ketika pipa terakhir diperbarui.

`pollingDisabledAt`

Tanggal dan waktu kapan, untuk pipeline yang dikonfigurasi untuk polling untuk deteksi perubahan, saat polling dinonaktifkan.

Pipa yang tidak aktif selama lebih dari 30 hari akan menonaktifkan pemungutan suara untuk pipa.

- Jaringan pipa yang tidak aktif akan menonaktifkan pemungutan suara setelah 30 hari tidak ada eksekusi.
- Saluran pipa yang menggunakan EventBridge, CodeStar Koneksi, atau webhook tidak akan terpengaruh.
- Pipa aktif tidak akan terpengaruh.

Untuk informasi selengkapnya, lihat `pollingDisabledAt` parameter di bawah PipelineMetadataobjek di Panduan CodePipeline API. Untuk langkah-langkah memigrasikan pipeline Anda dari polling ke deteksi perubahan berbasis peristiwa, lihat Mengubah Metode Deteksi.

Deklarasi tahap

Tingkat tahap pipa memiliki struktur dasar yang mencakup parameter dan sintaks berikut. Untuk informasi selengkapnya, lihat StageDeclarationobjek di Panduan CodePipeline API.

Contoh berikut menunjukkan tingkat tahap struktur pipa di kedua JSON dan YAMAL. Contoh menunjukkan dua tahap bernama `Source` dan`Build`. Contoh berisi dua kondisi, satu untuk on`Success` dan satu untuk`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

Nama tahapan.

actions

Tingkat tindakan pipa memiliki struktur dasar yang mencakup parameter dan sintaks berikut. Untuk melihat parameter dan contoh, lihatDeklarasi tindakan.

conditions

Ketentuan berisi satu atau lebih aturan yang tersedia dalam daftar aturan di CodePipeline. Jika semua aturan dalam suatu kondisi berhasil, maka kondisinya terpenuhi. Anda dapat mengonfigurasi kondisi sehingga ketika kriteria tidak terpenuhi, hasil yang ditentukan akan terlibat.

Anda dapat mengonfigurasi jenis kondisi berikut:

- `beforeEntry`
- `onFailure`
- `onSuccess`

Untuk informasi selengkapnya dan contoh tambahan, lihat Konfigurasikan kondisi untuk sebuah panggung.

`rules`

Setiap kondisi memiliki set aturan yang merupakan seperangkat aturan yang diurutkan yang dievaluasi bersama. Karena itu, jika satu aturan gagal dalam kondisi tersebut, maka kondisinya gagal. Anda dapat mengganti kondisi aturan saat runtime pipeline.

Aturan yang tersedia disediakan dalam referensi Aturan. Untuk informasi selengkapnya, lihat referensi struktur Aturan diReferensi struktur aturan.

Deklarasi tindakan

Tingkat tindakan pipa memiliki struktur dasar yang mencakup parameter dan sintaks berikut. Untuk informasi selengkapnya, lihat ActionDeclarationobjek di Panduan CodePipeline API.

Contoh berikut menunjukkan tingkat tindakan struktur pipa di kedua JSON dan YAMB.

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"
                }
            ]

. . .
```

Untuk daftar `configuration` detail contoh yang sesuai dengan jenis penyedia, lihatParameter konfigurasi yang valid untuk setiap jenis penyedia.

Struktur tindakan memiliki persyaratan sebagai berikut:

- Semua nama aksi dalam satu panggung harus unik.
- Tindakan sumber diperlukan untuk setiap pipa.
- Tindakan sumber yang tidak menggunakan koneksi dapat dikonfigurasi untuk deteksi perubahan atau untuk mematikan deteksi perubahan. Lihat Mengubah Metode Deteksi.
- Ini berlaku untuk semua tindakan, apakah mereka berada dalam tahap yang sama atau dalam tahap berikutnya, tetapi artefak input tidak harus menjadi tindakan berikutnya dalam urutan yang ketat dari tindakan yang menyediakan artefak keluaran. Tindakan secara paralel dapat mendeklarasikan bundel artefak keluaran yang berbeda, yang, pada gilirannya, dikonsumsi oleh tindakan berikut yang berbeda.
- Saat menggunakan bucket Amazon S3 sebagai lokasi penerapan, Anda juga menentukan kunci objek. Kunci objek dapat berupa nama file (objek) atau kombinasi awalan (path folder) dan nama file. Anda dapat menggunakan variabel untuk menentukan nama lokasi yang ingin digunakan pipeline. Tindakan penerapan Amazon S3 mendukung penggunaan variabel berikut di kunci objek Amazon S3.

  Menggunakan variabel di Amazon S3

Variabel	Contoh input konsol	Output
datetime	js-aplikasi/ {datetime} .zip	Stempel waktu UTC dalam format ini: < YYYY >-< MM >- DD >_< HH>-< MM >- < SS >

Variabel	Contoh input konsol	Output
		Contoh: js-application/2019-01-10_07-39-57.zip
uuid	js-aplikasi/ {uuid} .zip	UUID adalah pengidentifikasi unik global yang dijamin berbeda dari pengenal lainnya. UUID dalam format ini (semua digit dalam format heksadesimal): < 8-digit >-< 4-digit >- 4-digit >-< 4-digit >-< 12-digit > Contoh: js-application/54a60075-b96a-4bf3-9013-db3a9EXAMPLE.zip

`name`

Nama aksinya.

`region`

Untuk tindakan di mana penyedia adalah Layanan AWS, Wilayah AWS sumber daya.

Tindakan lintas wilayah menggunakan `Region` bidang untuk menentukan di Wilayah AWS mana tindakan akan dibuat. Sumber AWS daya yang dibuat untuk tindakan ini harus dibuat di Wilayah yang sama yang disediakan di `region` lapangan. Anda tidak dapat membuat tindakan lintas wilayah untuk jenis tindakan berikut:

- Tindakan sumber
- Tindakan oleh penyedia pihak ketiga
- Tindakan oleh penyedia kustom

roleArn

ARN dari peran layanan IAM yang melakukan tindakan yang dinyatakan. Ini diasumsikan melalui RoLearn yang ditentukan pada tingkat pipa.

namespace

Tindakan dapat dikonfigurasi dengan variabel. Anda menggunakan `namespace` bidang untuk mengatur namespace dan informasi variabel untuk variabel eksekusi. Untuk informasi referensi tentang variabel eksekusi dan variabel keluaran tindakan, lihatReferensi variabel.

> **Note**
>
> Untuk Amazon ECR, Amazon S3, CodeCommit atau sumber, Anda juga dapat membuat penggantian sumber menggunakan entri transformasi input untuk menggunakan `revisionValue` peristiwa EventBridge in untuk pipeline Anda, yang berasal dari variabel peristiwa sumber untuk kunci objek, komit, atau ID gambar Anda. `revisionValue` Untuk informasi selengkapnya, lihat langkah opsional untuk entri transformasi masukan yang disertakan dalam prosedur di bawahTindakan sumber dan sumber daya Amazon ECR EventBridge,Menyambungkan ke tindakan sumber Amazon S3 dengan sumber yang diaktifkan untuk acara, atauCodeCommit tindakan sumber dan EventBridge.

actionTypeId

ID tipe tindakan diidentifikasi sebagai kombinasi dari empat bidang berikut.

category

Jenis tindakan, atau langkah, dalam pipa, seperti tindakan sumber. Setiap jenis tindakan memiliki sekumpulan penyedia tindakan yang valid. Untuk daftar penyedia yang valid berdasarkan jenis tindakan, lihatReferensi struktur aksi.

Ini adalah `actionTypeId` kategori yang valid (tipe tindakan) untuk CodePipeline:

- `Source`
- `Build`
- `Approval`

- `Deploy`
- `Test`
- `Invoke`
- `Compute`

`owner`

Untuk semua jenis tindakan yang saat ini didukung, satu-satunya string pemilik yang valid adalah`AWS`,`ThirdParty`, atau`Custom`. Untuk string pemilik yang valid untuk tindakan tertentu, lihat[Referensi struktur aksi](#).

Untuk informasi lebih lanjut, lihat [Referensi API CodePipeline](#) .

`version`

Versi aksinya.

`provider`

Penyedia tindakan, seperti CodeBuild.

- Jenis penyedia yang valid untuk kategori tindakan bergantung pada kategori. Misalnya, untuk kategori tindakan sumber, jenis penyedia yang valid adalah`S3`,`CodeStarSourceConnection`,`CodeCommit`, atau`Amazon ECR`. Contoh ini menunjukkan struktur untuk tindakan sumber dengan `S3` penyedia:

```
"actionTypeId": {
  "category": "Source",
  "owner": "AWS",
  "version": "1",
  "provider": "S3"},
```

`InputArtifacts`

Bidang ini berisi struktur artefak masukan, jika didukung untuk kategori tindakan. Artefak masukan dari suatu tindakan harus sama persis dengan artefak keluaran yang dideklarasikan dalam tindakan sebelumnya. Misalnya, jika tindakan sebelumnya mencakup deklarasi berikut:

```
"outputArtifacts": [
    {
    "MyApp"
    }
],
```

dan tidak ada artefak keluaran lainnya, maka artefak input dari tindakan berikut harus:

```
"inputArtifacts": [
    {
    "MyApp"
    }
],
```

Misalnya, aksi sumber tidak dapat memiliki artefak input karena ini adalah tindakan pertama dalam pipeline. Namun, aksi sumber akan selalu memiliki artefak keluaran yang diproses oleh tindakan berikut. Artefak keluaran untuk aksi sumber adalah file aplikasi dari repositori sumber, di-zip dan disediakan melalui bucket artefak, yang diproses oleh tindakan berikut, seperti CodeBuild tindakan yang bertindak pada file aplikasi dengan perintah build.

Sebagai contoh tindakan yang tidak dapat memiliki artefak keluaran, tindakan penerapan tidak memiliki artefak keluaran karena tindakan ini umumnya merupakan tindakan terakhir dalam pipeline.

`name`

Nama artefak untuk artefak masukan aksi.

`outputArtifacts`

Nama artefak keluaran harus unik dalam pipa. Misalnya, pipeline dapat menyertakan satu tindakan yang memiliki artefak keluaran bernama `"MyApp"` dan tindakan lain yang memiliki artefak keluaran bernama. `"MyBuiltApp"` Namun, pipeline tidak dapat menyertakan dua tindakan yang keduanya memiliki artefak keluaran bernama`"MyApp"`.

Bidang ini berisi struktur artefak keluaran, jika didukung untuk kategori tindakan. Artefak keluaran dari suatu tindakan harus sama persis dengan artefak keluaran yang dinyatakan dalam tindakan sebelumnya. Misalnya, jika tindakan sebelumnya mencakup deklarasi berikut:

```
"outputArtifacts": [
```

```
    {
    "MyApp"
    }
],
```

dan tidak ada artefak keluaran lainnya, maka artefak input dari tindakan berikut harus:

```
"inputArtifacts": [
    {
    "MyApp"
    }
],
```

Misalnya, aksi sumber tidak dapat memiliki artefak input karena ini adalah tindakan pertama dalam pipeline. Namun, aksi sumber akan selalu memiliki artefak keluaran yang diproses oleh tindakan berikut. Artefak keluaran untuk aksi sumber adalah file aplikasi dari repositori sumber, di-zip dan disediakan melalui bucket artefak, yang diproses oleh tindakan berikut, seperti CodeBuild tindakan yang bertindak pada file aplikasi dengan perintah build.

Sebagai contoh tindakan yang tidak dapat memiliki artefak keluaran, tindakan penerapan tidak memiliki artefak keluaran karena tindakan ini umumnya merupakan tindakan terakhir dalam pipeline.

`name`

Nama artefak untuk artefak keluaran aksi.

`configuration`(oleh penyedia tindakan)

Konfigurasi tindakan berisi detail dan parameter yang sesuai dengan jenis penyedia. Pada bagian di bawah ini, parameter konfigurasi tindakan contoh khusus untuk tindakan sumber S3.

Konfigurasi tindakan dan batas input/output artefak dapat bervariasi menurut penyedia tindakan. Untuk daftar contoh konfigurasi tindakan menurut penyedia tindakan, lihat Referensi struktur aksi dan tabel diParameter konfigurasi yang valid untuk setiap jenis penyedia. Tabel menyediakan tautan ke referensi tindakan untuk setiap jenis penyedia, yang mencantumkan parameter konfigurasi untuk setiap tindakan secara detail. Untuk tabel dengan batas artefak input dan output untuk setiap penyedia tindakan, lihatArtefak input dan output yang valid untuk setiap jenis tindakan.

Pertimbangan berikut berlaku untuk bekerja dengan tindakan:

- Tindakan sumber tidak memiliki artefak masukan, dan tindakan penerapan tidak memiliki artefak keluaran.
- Untuk penyedia tindakan sumber yang tidak menggunakan koneksi, seperti S3, Anda harus menggunakan `PollForSourceChanges` parameter untuk menentukan apakah Anda ingin pipeline dimulai secara otomatis ketika perubahan terdeteksi. Lihat Pengaturan yang valid untuk `PollForSourceChanges` parameter.
- Untuk mengonfigurasi deteksi perubahan otomatis untuk memulai pipeline, atau menonaktifkan deteksi perubahan, lihatTindakan sumber dan metode deteksi perubahan.
- Untuk mengonfigurasi pemicu dengan pemfilteran, gunakan tindakan sumber untuk koneksi, lalu lihat. Otomatiskan saluran pipa awal menggunakan pemicu dan penyaringan
- Untuk variabel output untuk setiap tindakan, lihatReferensi variabel.

> **Note**
>
> Untuk Amazon ECR, Amazon S3, CodeCommit atau sumber, Anda juga dapat membuat penggantian sumber menggunakan entri transformasi input untuk menggunakan `revisionValue` peristiwa EventBridge in untuk pipeline Anda, yang berasal dari variabel peristiwa sumber untuk kunci objek, komit, atau ID gambar Anda. `revisionValue` Untuk informasi selengkapnya, lihat langkah opsional untuk entri transformasi masukan yang disertakan dalam prosedur di bawahTindakan sumber dan sumber daya Amazon ECR EventBridge,Menyambungkan ke tindakan sumber Amazon S3 dengan sumber yang diaktifkan untuk acara, atauCodeCommit tindakan sumber dan EventBridge.

> **Important**
>
> Pipa yang tidak aktif selama lebih dari 30 hari akan menonaktifkan pemungutan suara untuk pipa. Untuk informasi lebih lanjut, lihat pollingDisabledAtdi referensi struktur pipa. Untuk langkah-langkah memigrasikan pipeline Anda dari polling ke deteksi perubahan berbasis peristiwa, lihat Mengubah Metode Deteksi.

> **Note**
>
> Tindakan sumber CodeCommit dan S3 memerlukan sumber daya deteksi perubahan yang dikonfigurasi (EventBridge aturan) atau menggunakan opsi untuk melakukan polling

repositori untuk perubahan sumber. Untuk pipeline dengan Bitbucket, GitHub, atau tindakan sumber Server GitHub Perusahaan, Anda tidak perlu menyiapkan webhook atau default untuk polling. Tindakan koneksi mengelola deteksi perubahan untuk Anda.

`runOrder`

Sebuah bilangan bulat positif yang menunjukkan urutan run dari tindakan dalam tahap. Tindakan paralel di panggung ditampilkan sebagai memiliki bilangan bulat yang sama. Misalnya, dua tindakan dengan urutan run dua akan berjalan secara paralel setelah aksi pertama di tahap berjalan.

`runOrder`Nilai default untuk tindakan adalah 1. Nilai harus berupa bilangan bulat positif (bilangan asli). Anda tidak dapat menggunakan pecahan, desimal, angka negatif, atau nol. Untuk menentukan urutan serial tindakan, gunakan angka terkecil untuk tindakan pertama dan angka yang lebih besar untuk masing-masing tindakan lainnya secara berurutan. Untuk menentukan tindakan paralel, gunakan bilangan bulat yang sama untuk setiap tindakan yang ingin Anda jalankan secara paralel. Di konsol, Anda dapat menentukan urutan serial untuk tindakan dengan memilih Tambahkan grup tindakan pada tingkat di tahap di mana Anda ingin menjalankannya, atau Anda dapat menentukan urutan paralel dengan memilih Tambah tindakan. Kelompok aksi mengacu pada urutan run dari satu atau lebih tindakan pada tingkat yang sama.

Misalnya, jika Anda ingin tiga tindakan berjalan secara berurutan dalam satu tahap, Anda akan memberikan tindakan pertama `runOrder` nilai 1, tindakan kedua `runOrder` nilai 2, dan yang ketiga `runOrder` nilai 3. Namun, jika Anda ingin tindakan kedua dan ketiga berjalan secara paralel, Anda akan memberikan tindakan pertama `runOrder` nilai 1 dan tindakan kedua dan ketiga `runOrder` nilai 2.

> **Note**
>
> Penomoran tindakan serial tidak harus dalam urutan yang ketat. Misalnya, jika Anda memiliki tiga tindakan secara berurutan dan memutuskan untuk menghapus tindakan kedua, Anda tidak perlu memberi nomor ulang `runOrder` nilai tindakan ketiga. Karena `runOrder` nilai tindakan itu (3) lebih tinggi dari `runOrder` nilai tindakan pertama (1), ia berjalan secara serial setelah tindakan pertama di panggung.

Penyedia tindakan yang valid di CodePipeline

Format struktur pipa digunakan untuk membangun tindakan dan tahapan dalam pipa. Tipe tindakan terdiri dari kategori tindakan dan jenis penyedia.

Setiap kategori tindakan memiliki daftar penyedia tindakan yang valid. Untuk mereferensikan penyedia tindakan yang valid untuk setiap kategori tindakan, lihatReferensi struktur aksi.

Setiap kategori tindakan memiliki seperangkat penyedia yang ditunjuk. Setiap penyedia tindakan, seperti Amazon S3, memiliki nama penyedia, sepertiS3, yang harus digunakan di `Provider` bidang dalam kategori tindakan dalam struktur pipeline Anda.

Ada tiga nilai yang valid untuk `Owner` bidang di bagian kategori tindakan dalam struktur pipeline Anda:`AWS`,`ThirdParty`, dan`Custom`.

Untuk menemukan nama penyedia dan informasi pemilik untuk penyedia tindakan Anda, lihat Referensi struktur aksi atauArtefak input dan output yang valid untuk setiap jenis tindakan.

Tabel ini mencantumkan penyedia yang valid berdasarkan jenis tindakan.

> **Note**
>
> Untuk tindakan Bitbucket GitHub,, atau GitHub Enterprise Server, lihat topik referensi CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri tindakan.

Penyedia tindakan yang valid berdasarkan jenis tindakan

Kategori aksi	Penyedia tindakan yang valid	Jenis pipa didukung	Referensi tindakan
Sumber	Amazon S3	V1, V2	Referensi tindakan sumber Amazon S3
	Amazon ECR	V1, V2	Referensi tindakan sumber Amazon ECR
	CodeCommit	V1, V2	CodeCommit referensi tindakan sumber

Kategori aksi	Penyedia tindakan yang valid	Jenis pipa didukung	Referensi tindakan
	CodeStarSourceConnection (untuk Bitbucket, GitHub, tindakan Server GitHub Perusahaan)	V1, V2	CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri
Build	Tindakan ECR `ECRBuildAndPublish` Amazon	Hanya V2	`ECRBuildAndPublish` membangun referensi tindakan
	CodeBuild	V1, V2	AWS CodeBuild membangun dan menguji referensi tindakan
	Tindakan perintah (lihat Compute)	Hanya V2	
	Kustom CloudBees	V1, V2	Artefak input dan output yang valid untuk setiap jenis tindakan
	Jenkins Kustom	V1, V2	Artefak input dan output yang valid untuk setiap jenis tindakan
	Kustom TeamCity	V1, V2	Artefak input dan output yang valid untuk setiap jenis tindakan
Uji	CodeBuild	V1, V2	AWS CodeBuild membangun dan menguji referensi tindakan

Kategori aksi	Penyedia tindakan yang valid	Jenis pipa didukung	Referensi tindakan
	AWS Device Farm	V1, V2	Artefak input dan output yang valid untuk setiap jenis tindakan
	Kustom BlazeMeter	V1, V2	Artefak input dan output yang valid untuk setiap jenis tindakan
	ThirdParty GhostInspector		Artefak input dan output yang valid untuk setiap jenis tindakan
	Jenkins Kustom		Artefak input dan output yang valid untuk setiap jenis tindakan
	ThirdParty StormRunner Beban Fokus Mikro		Artefak input dan output yang valid untuk setiap jenis tindakan
	ThirdParty Nouvola		Artefak input dan output yang valid untuk setiap jenis tindakan
	ThirdParty Runscope		Artefak input dan output yang valid untuk setiap jenis tindakan
Deploy	Amazon S3		Referensi tindakan penerapan Amazon S3
	CloudFormation		CloudFormation menyebarkan referensi tindakan

Kategori aksi	Penyedia tindakan yang valid	Jenis pipa didukung	Referensi tindakan
	CodeDeploy		Artefak input dan output yang valid untuk setiap jenis tindakan
	EC2 Menyebarkan tindakan	Hanya V2	Referensi tindakan Amazon EC2
	Amazon ECS		Artefak input dan output yang valid untuk setiap jenis tindakan
	Amazon ECS (Biru/Hijau) (ini adalah aksinya) `CodeDeployToECS`		Artefak input dan output yang valid untuk setiap jenis tindakan
	Aksi Amazon EKS	Hanya V2	Referensi tindakan penerapan Amazon Elastic Kubernetes Service EKS
	Elastic Beanstalk		Artefak input dan output yang valid untuk setiap jenis tindakan
	AWS AppConfig		AWS AppConfig menyebarkan referensi tindakan
	OpsWorks		Artefak input dan output yang valid untuk setiap jenis tindakan
	Service Catalog		Artefak input dan output yang valid untuk setiap jenis tindakan

Kategori aksi	Penyedia tindakan yang valid	Jenis pipa didukung	Referensi tindakan
	Amazon Alexa		Artefak input dan output yang valid untuk setiap jenis tindakan
	Kustom XebiaLabs		Artefak input dan output yang valid untuk setiap jenis tindakan
Persetujuan	Manual		Artefak input dan output yang valid untuk setiap jenis tindakan
Panggil	CodePipelineMemanggil tindakan		AWS CodePipeline memanggil referensi tindakan
	AWS Lambda		AWS Lambda memanggil referensi tindakan
	AWS Step Functions		AWS Step Functions memanggil referensi tindakan
	InspectorScan		Amazon Inspector `InspectorScan` memanggil referensi tindakan
Hitung	Perintah tindakan		Referensi tindakan perintah

Beberapa jenis tindakan hanya CodePipeline tersedia di AWS Wilayah tertentu. Ada kemungkinan bahwa tipe tindakan tersedia di AWS Wilayah, tetapi AWS penyedia untuk jenis tindakan tersebut tidak tersedia.

Untuk informasi selengkapnya tentang setiap penyedia tindakan, lihatIntegrasi dengan tipe CodePipeline tindakan.

Pengaturan yang valid untuk **PollForSourceChanges** parameter

`PollForSourceChanges`Parameter default ditentukan oleh metode yang digunakan untuk membuat pipeline, seperti yang dijelaskan dalam tabel berikut. Dalam banyak kasus, `PollForSourceChanges` parameter default ke true dan harus dinonaktifkan.

Ketika `PollForSourceChanges` parameter default ke true, Anda harus melakukan hal berikut:

- Tambahkan `PollForSourceChanges` parameter ke file atau CloudFormation template JSON.
- Buat sumber daya deteksi perubahan (Aturan CloudWatch acara, sebagaimana berlaku).
- Atur `PollForSourceChanges` parameter ke false.

> **Note**
>
> Jika Anda membuat aturan CloudWatch Acara atau webhook, Anda harus menyetel parameter ke false untuk menghindari memicu pipeline lebih dari sekali.

`PollForSourceChanges`Parameter ini tidak digunakan untuk tindakan sumber Amazon ECR.

- **`PollForSourceChanges`**parameter default

Sumber	Metode pembuatan	Contoh “konfigurasi” output struktur JSON
CodeCommit	Pipeline dibuat dengan konsol (dan sumber daya deteksi perubahan dibuat oleh konsol). Parameter ditampilkan dalam output struktur pipa dan defaultnya. `false`	`BranchName": "main", "PollForSourceChanges": "false", "RepositoryName": "my-repo"`
	Pipeline dibuat dengan CLI atau CloudFormation, dan `PollForSourceChanges`	`BranchName": "main", "RepositoryName": "my-repo"`

Sumber	Metode pembuatan	Contoh “konfigurasi” output struktur JSON
	parameter tidak ditampilkan dalam output JSON, tetapi disetel ke .² `true`	
Amazon S3	Pipeline dibuat dengan konsol (dan sumber daya deteksi perubahan dibuat oleh konsol). Parameter ditampilkan dalam output struktur pipa dan defaultnya. `false`	`"S3Bucket": "my-bucket", "S3ObjectKey": "object.zip", "PollForSourceChanges": "false"`
	Pipeline dibuat dengan CLI atau CloudFormation, dan `PollForSourceChanges` parameter tidak ditampilkan dalam output JSON, tetapi disetel ke .² `true`	`"S3Bucket": "my-bucket", "S3ObjectKey": "object.zip"`
GitHub	Pipeline dibuat dengan konsol (dan sumber daya deteksi perubahan dibuat oleh konsol). Parameter ditampilkan dalam output struktur pipa dan defaultnya. `false`	`"Owner": "MyGitHubAccountName ", "Repo": " MyGitHubRepositoryName " "PollForSourceChanges": "false", "Branch": " main" "OAuthToken": " ****"`
	Pipeline dibuat dengan CLI atau CloudFormation, dan `PollForSourceChanges` parameter tidak ditampilkan dalam output JSON, tetapi disetel ke .² `true`	`"Owner": "MyGitHubAccountName ", "Repo": "MyGitHubRepositoryName ", "Branch": " main", "OAuthToken": " ****"`

Sumber	Metode pembuatan	Contoh “konfigurasi” output struktur JSON
	² Jika `PollForSourceChanges` telah ditambahkan pada setiap titik ke struktur JSON atau CloudFormation template, itu ditampilkan seperti yang ditunjukkan: `"PollForSourceChanges": "true",` ³ Untuk informasi tentang sumber daya deteksi perubahan yang berlaku untuk setiap penyedia sumber, lihat Mengubah Metode Deteksi.	

Artefak input dan output yang valid untuk setiap jenis tindakan

Bergantung pada jenis tindakan dan penyedia, Anda dapat memiliki jumlah artefak input dan output berikut.

Kendala tipe aksi untuk artefak

Pemilik	Jenis tindakan	Penyedia	Jumlah artefak masukan yang valid	Jumlah artefak keluaran yang valid
`AWS`	Sumber	`S3`	0	1
`AWS`	Sumber	`CodeCommit`	0	1
`AWS`	Sumber	`ECR`	0	1
`ThirdParty`	Sumber	`CodeStarSourceConnection`	0	1
`AWS`	Build	`CodeBuild`	1 sampai 5	0 hingga 5
`AWS`	Uji	`CodeBuild`	1 sampai 5	0 hingga 5

Pemilik	Jenis tindakan	Penyedia	Jumlah artefak masukan yang valid	Jumlah artefak keluaran yang valid
AWS	Uji	DeviceFarm	1	0
AWS	Persetujuan	ThirdParty	0	0
AWS	Deploy	S3	1	0
AWS	Deploy	CloudFormation	0 hingga 10	0 hingga 1
AWS	Deploy	CodeDeploy	1	0
AWS	Deploy	ElasticBeanstalk	1	0
AWS	Deploy	OpsWorks	1	0
AWS	Deploy	ECS	1	0
AWS	Deploy	ServiceCatalog	1	0
AWS	Panggil	Lambda	0 hingga 5	0 hingga 5
ThirdParty	Deploy	AlexaSkillsKit	1 hingga 2	0
Custom	Build	Jenkins	0 hingga 5	0 hingga 5
Custom	Uji	Jenkins	0 hingga 5	0 hingga 5
Custom	Setiap kategori yang didukung	Seperti yang ditentukan dalam tindakan kustom	0 hingga 5	0 hingga 5

Parameter konfigurasi yang valid untuk setiap jenis penyedia

Bagian ini mencantumkan `configuration` parameter yang valid untuk setiap penyedia tindakan.

Setiap tindakan harus memiliki konfigurasi tindakan yang valid, yang bergantung pada jenis penyedia untuk tindakan tersebut. Tabel berikut mencantumkan elemen konfigurasi tindakan yang diperlukan untuk setiap jenis penyedia yang valid:

Properti konfigurasi tindakan untuk tipe penyedia

Nama penyedia	Nama penyedia dalam tipe tindakan	Properti konfigurasi	Diperlukan/ Opsional
Amazon S3 (Menyebarkan penyedia tindakan)	Untuk informasi selengkapnya, termasuk contoh yang terkait dengan parameter tindakan Menerapkan Amazon S3, lihat. Referensi tindakan penerapan Amazon S3		
Amazon S3 (Penyedia tindakan sumber)	Untuk informasi selengkapnya, termasuk contoh yang terkait dengan parameter tindakan sumber Amazon S3, lihat. Referensi tindakan sumber Amazon S3		
Amazon ECR	Untuk informasi selengkapnya, termasuk contoh yang terkait dengan parameter Amazon ECR, lihatReferensi tindakan sumber Amazon ECR.		
CodeCommit	Untuk informasi selengkapnya, termasuk contoh yang terkait dengan CodeCommit parameter, lihatCodeCommit referensi tindakan sumber.		
CodeStarSourceConnection tindakan untuk Bitbucket, GitHub (melalui GitHub	Untuk informasi selengkapnya, termasuk contoh konfigurasi tindakan, lihatParameter konfigurasi.		

Nama penyedia	Nama penyedia dalam tipe tindakan	Properti konfigurasi	Diperlukan/ Opsional
aplikasi), GHES, dan GitLab			
GitHub (melalui OAuth aplikasi)	Untuk informasi selengkapnya, termasuk contoh yang terkait dengan GitHub parameter, lihatGitHub (melalui OAuth aplikasi) referensi tindakan sumber. Ini adalah GitHub tindakan Versi 1.		
AWS CloudFormation	Untuk informasi selengkapnya, termasuk contoh yang terkait dengan AWS CloudFormation parameter, lihatCloudFormation menyebarkan referensi tindakan.		
CodeBuild	Untuk deskripsi dan contoh selengkapnya yang terkait dengan CodeBuild parameter, lihatAWS CodeBuild membangun dan menguji referensi tindakan.		
CodeDeploy	Untuk deskripsi dan contoh selengkapnya yang terkait dengan CodeDeploy parameter, lihatAWS CodeDeploy menyebarkan referensi tindakan.		
AWS Device Farm	Untuk deskripsi dan contoh selengkapnya yang terkait dengan AWS Device Farm parameter, lihatAWS Device Farm referensi tindakan uji.		
AWS Elastic Beanstalk	`ElasticBeanstalk`	`ApplicationName`	Diperlukan
		`EnvironmentName`	Diperlukan
AWS Lambda	Untuk informasi selengkapnya, termasuk contoh yang terkait dengan AWS Lambda parameter, lihatAWS Lambda memanggil referensi tindakan.		
AWS OpsWorks Stacks	`OpsWorks`	`Stack`	Diperlukan
		`Layer`	Opsional
		`App`	Diperlukan

Nama penyedia	Nama penyedia dalam tipe tindakan	Properti konfigurasi	Diperlukan/ Opsional
Amazon ECS	Untuk deskripsi selengkapnya dan contoh yang terkait dengan parameter Amazon ECS, lihatReferensi tindakan penerapan Amazon Elastic Container Service.		
Amazon ECS dan CodeDeploy (Biru/Hijau)	Untuk deskripsi dan contoh selengkapnya yang terkait dengan Amazon ECS dan parameter CodeDeploy biru/hijau, lihat. Amazon Elastic Container Service dan CodeDeploy referensi aksi penerapan biru-hijau		
Service Catalog	`ServiceCatalog`	`TemplateFilePath`	Diperlukan
		`ProductVersionName`	Diperlukan
		`ProductType`	Diperlukan
		`ProductVersionDescription`	Opsional
		`ProductId`	Diperlukan
Kit Keterampilan Alexa	`AlexaSkillsKit`	`ClientId`	Diperlukan
		`ClientSecret`	Diperlukan
		`RefreshToken`	Diperlukan
		`SkillId`	Diperlukan
Jenkins	Nama tindakan yang Anda berikan di CodePipeline Plugin untuk Jenkins (misalnya ,*MyJenkins ProviderName*)	`ProjectName`	Diperlukan
Persetujuan Manual	`Manual`	`CustomData`	Opsional
		`ExternalEntityLink`	Opsional

Nama penyedia	Nama penyedia dalam tipe tindakan	Properti konfigurasi	Diperlukan/ Opsional
		`NotificationArn`	Opsional

Contoh berikut menunjukkan konfigurasi yang valid untuk tindakan penerapan yang menggunakan Alexa Skills Kit:

```
"configuration": {
  "ClientId": "amzn1.application-oa2-client.aadEXAMPLE",
  "ClientSecret": "****",
  "RefreshToken": "****",
  "SkillId": "amzn1.ask.skill.22649d8f-0451-4b4b-9ed9-bfb6cEXAMPLE"
}
```

Contoh berikut menunjukkan konfigurasi yang valid untuk persetujuan manual:

```
"configuration": {
  "CustomData": "Comments on the manual approval",
  "ExternalEntityLink": "http://my-url.com",
  "NotificationArn": "arn:aws:sns:us-west-2:12345EXAMPLE:Notification"
}
```

Referensi struktur aksi

Bagian ini adalah referensi untuk konfigurasi tindakan saja. Untuk gambaran konseptual dari struktur pipa, lihatCodePipeline referensi struktur pipa.

Setiap penyedia tindakan CodePipeline menggunakan satu set bidang konfigurasi wajib dan opsional dalam struktur pipa. Bagian ini memberikan informasi referensi berikut oleh penyedia tindakan:

- Nilai yang valid untuk `ActionType` bidang yang termasuk dalam blok tindakan struktur pipa, seperti `Owner` dan`Provider`.
- Deskripsi dan informasi referensi lainnya untuk `Configuration` parameter (wajib dan opsional) termasuk dalam bagian tindakan struktur pipa.
- Contoh bidang tindakan JSON dan YAMAL yang valid.

Bagian ini diperbarui secara berkala dengan lebih banyak penyedia tindakan. Informasi referensi saat ini tersedia untuk penyedia tindakan berikut:

Topik

- Referensi tindakan Amazon EC2
- Referensi tindakan sumber Amazon ECR
- ECRBuildAndPublishmembangun referensi tindakan
- Amazon Elastic Container Service dan CodeDeploy referensi aksi penerapan biru-hijau
- Referensi tindakan penerapan Amazon Elastic Container Service
- Referensi tindakan penerapan Amazon Elastic Kubernetes Service EKS
- AWS Lambda menyebarkan referensi tindakan
- Referensi tindakan penerapan Amazon S3
- Referensi tindakan sumber Amazon S3
- AWS AppConfig menyebarkan referensi tindakan
- CloudFormation menyebarkan referensi tindakan
- CloudFormation StackSets menyebarkan referensi tindakan
- AWS CodeBuild membangun dan menguji referensi tindakan
- AWS CodePipeline memanggil referensi tindakan
- CodeCommit referensi tindakan sumber

- AWS CodeDeploy menyebarkan referensi tindakan
- CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri
- Referensi tindakan perintah
- AWS Device Farm referensi tindakan uji
- Referensi aksi penyebaran Elastic Beanstalk
- Amazon Inspector InspectorScan memanggil referensi tindakan
- AWS Lambda memanggil referensi tindakan
- AWS OpsWorks menyebarkan referensi tindakan
- AWS Service Catalog menyebarkan referensi tindakan
- AWS Step Functions memanggil referensi tindakan

Referensi tindakan Amazon EC2

Anda menggunakan EC2 tindakan Amazon EC2 untuk menyebarkan kode aplikasi ke armada penerapan Anda. Armada penyebaran Anda dapat terdiri dari instans Amazon EC2 Linux atau node yang dikelola Linux SSM. Instans Anda harus memiliki agen SSM yang diinstal.

> **Note**
>
> Tindakan ini hanya mendukung jenis instance Linux. Ukuran armada maksimum yang didukung adalah 500 instance.

Tindakan akan memilih sejumlah instance berdasarkan maksimum yang ditentukan. Instance yang gagal dari instance sebelumnya akan dipilih terlebih dahulu. Tindakan akan melewati penerapan pada instance tertentu jika instance telah menerima penerapan artefak input yang sama, seperti kasus di mana tindakan gagal sebelumnya.

> **Note**
>
> Tindakan ini hanya didukung untuk pipeline tipe V2.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Izin kebijakan peran layanan untuk tindakan penerapan EC2
- Menyebarkan referensi file spesifikasi
- Deklarasi tindakan
- Deklarasi tindakan dengan contoh spesifikasi Deploy
- Lihat juga

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `EC2`
- Versi: `1`

Parameter konfigurasi

InstanceTagKey

Wajib: Ya

Kunci tag dari instans yang Anda buat di Amazon EC2, seperti. `Name`

InstanceTagValue

Wajib: Tidak

Nilai tag instans yang Anda buat di Amazon EC2, seperti. `my-instances`

Ketika nilai ini tidak ditentukan, semua instance dengan InstanceTagKeyakan dicocokkan.

InstanceType

Wajib: Ya

Jenis instance atau node SSM yang dibuat di Amazon EC2. Nilai yang valid adalah `EC2` dan `SSM_MANAGED_NODE`.

Anda harus sudah membuat, menandai, dan menginstal agen SSM pada semua instance.

> **Note**
>
> Saat membuat instance, Anda membuat atau menggunakan peran instans EC2 yang ada. Untuk menghindari `Access Denied` kesalahan, Anda harus menambahkan izin bucket S3 ke peran instance untuk memberikan izin instance ke bucket artefak CodePipeline . Buat peran default atau perbarui peran yang ada dengan `s3:GetObject` izin yang tercakup ke bucket artefak untuk Wilayah pipeline Anda.

TargetDirectory

Diperlukan: Ya (Jika skrip ditentukan)

Direktori yang akan digunakan pada instans Amazon EC2 Anda untuk menjalankan skrip.

DeploySpec

Diperlukan: Ya (Jika spesifikasi penerapan ditentukan)

File yang akan digunakan untuk mengkonfigurasi instalasi penerapan dan peristiwa siklus hidup. Untuk menyebarkan deskripsi dan informasi bidang spesifikasi, lihat. Menyebarkan referensi file spesifikasi Untuk melihat konfigurasi tindakan dengan file spesifikasi deploy yang ditentukan, lihat contoh di. Deklarasi tindakan dengan contoh spesifikasi Deploy

MaxBatch

Wajib: Tidak

Jumlah maksimum instance yang diizinkan untuk digunakan secara paralel.

MaxError

Wajib: Tidak

Jumlah maksimum kesalahan instans yang diizinkan selama penerapan.

TargetGroupNameList

Wajib: Tidak

Daftar nama grup target untuk penyebaran. Anda harus sudah membuat grup target.

Grup sasaran menyediakan serangkaian instance untuk memproses permintaan tertentu. Jika grup target ditentukan, instance akan dihapus dari grup target sebelum penerapan dan ditambahkan kembali ke grup target setelah penerapan.

PreScript

Wajib: Tidak

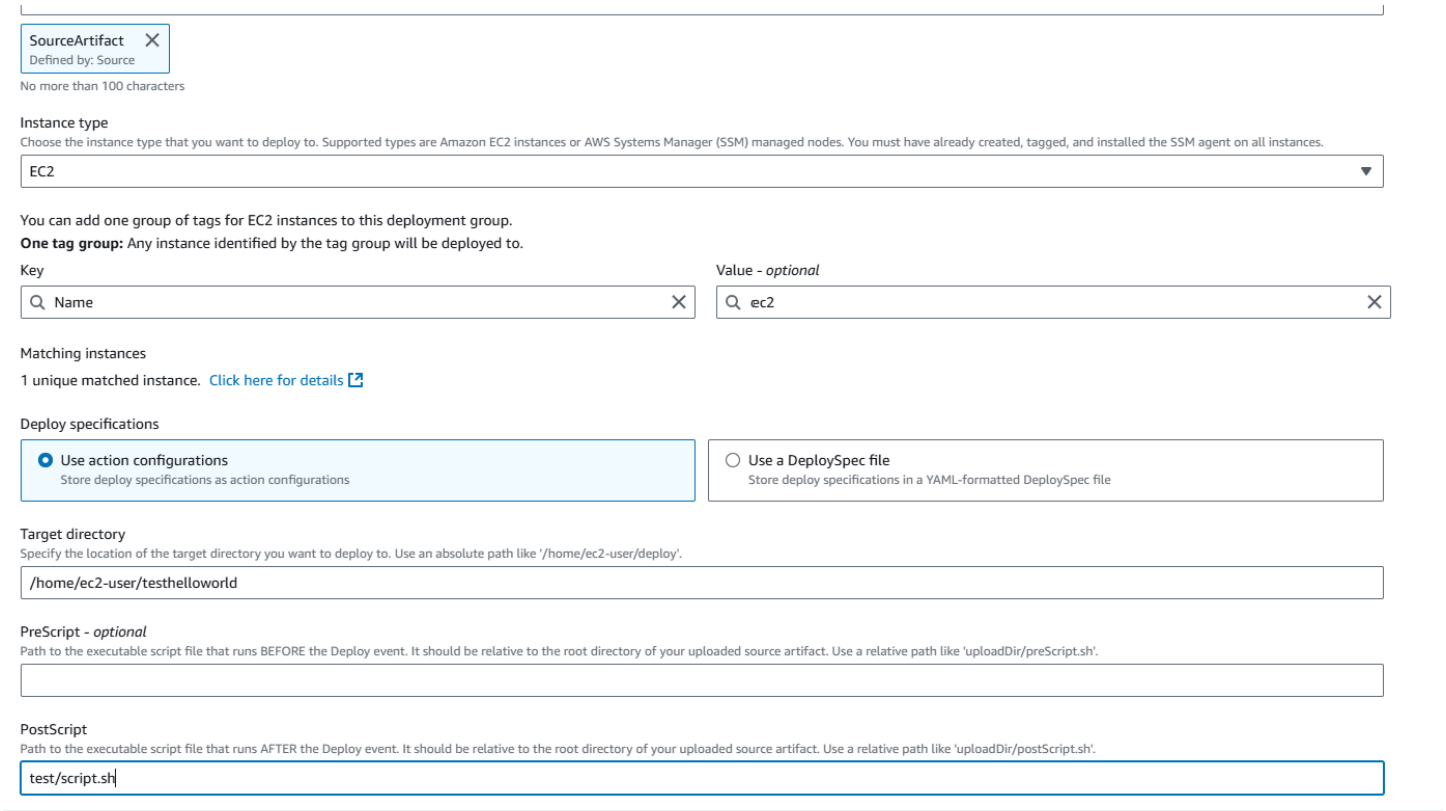
Script yang akan dijalankan sebelum fase aksi Deploy.

PostScript

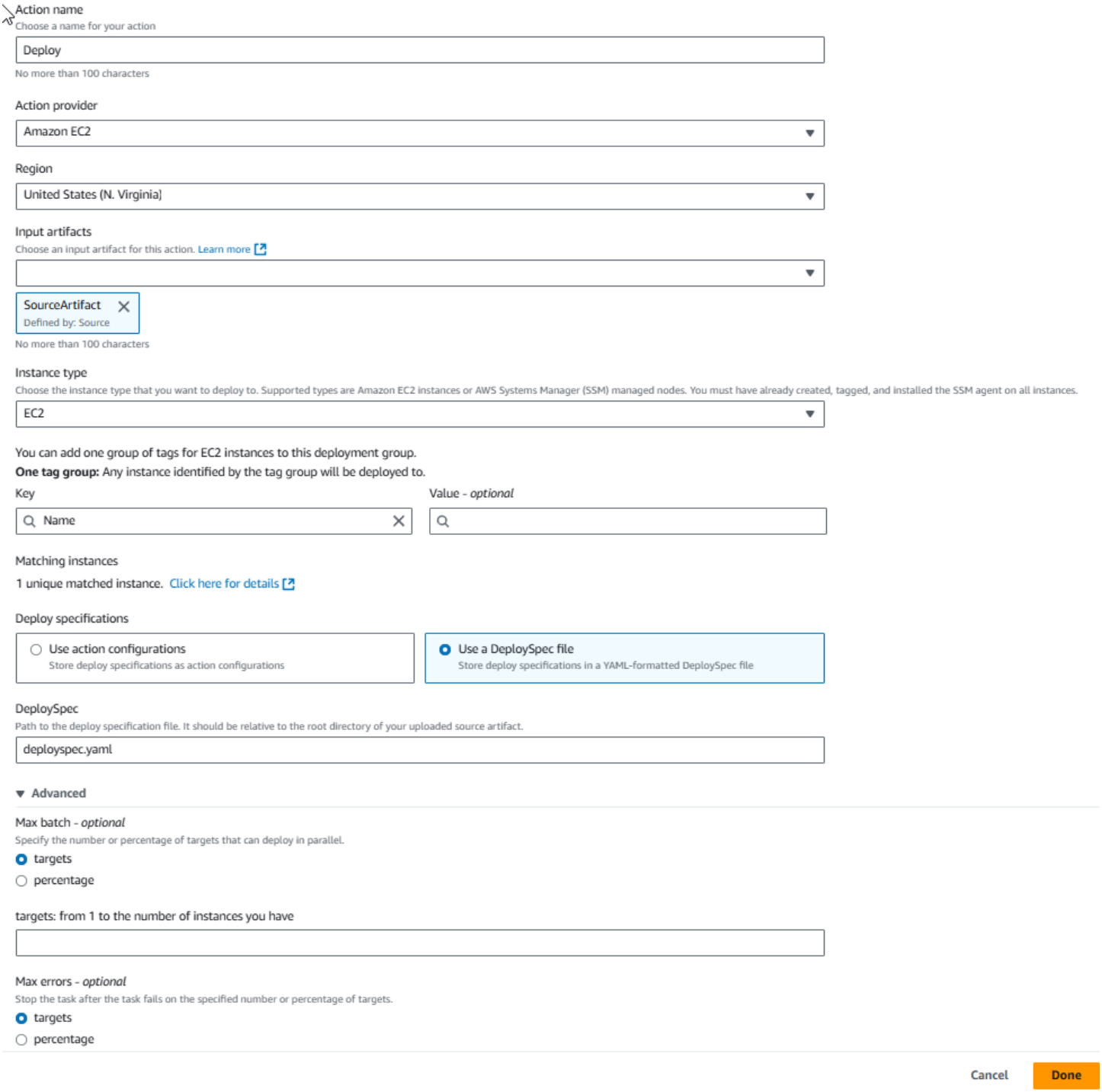
Wajib: Ya

Script yang akan dijalankan setelah fase aksi Deploy.

Gambar berikut menunjukkan contoh halaman Edit untuk tindakan di mana konfigurasi Use action dipilih.



Gambar berikut menunjukkan contoh halaman Edit untuk tindakan di mana Gunakan DeploySpec file dipilih.



Artefak masukan

- Jumlah artefak: 1
- Deskripsi: File yang disediakan, jika ada, untuk mendukung tindakan skrip selama penyebaran.

Artefak keluaran

- Jumlah artefak: `0`
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Izin kebijakan peran layanan untuk tindakan penerapan EC2

Saat CodePipeline menjalankan tindakan, peran CodePipeline layanan memerlukan izin berikut, yang dicakup dengan tepat untuk akses dengan hak istimewa paling sedikit.

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"
            ]
        }
    ]
```

```
}
```

Grup log untuk pipeline Anda di CloudWatch log

Saat CodePipeline menjalankan tindakan, CodePipeline buat grup log menggunakan nama pipeline sebagai berikut. Ini memungkinkan Anda untuk mengurangi izin untuk mencatat sumber daya menggunakan nama pipeline.

```
/aws/codepipeline/MyPipelineName
```

Izin pencatatan berikut disertakan dalam pembaruan di atas untuk peran layanan.

- log: CreateLogGroup
- log: CreateLogStream
- log: PutLogEvents

Untuk melihat log di konsol menggunakan halaman dialog detail tindakan, izin untuk melihat log harus ditambahkan ke peran konsol. Untuk informasi selengkapnya, lihat contoh kebijakan izin konsol dilIzin diperlukan untuk melihat log komputasi di konsol CodePipeline.

Izin kebijakan peran layanan untuk log CloudWatch

Saat CodePipeline menjalankan tindakan, CodePipeline buat grup log menggunakan nama pipeline sebagai berikut. Ini memungkinkan Anda untuk mengurangi izin untuk mencatat sumber daya menggunakan nama pipeline.

```
/aws/codepipeline/MyPipelineName
```

Untuk melihat log di konsol menggunakan halaman dialog detail tindakan, izin untuk melihat log harus ditambahkan ke peran konsol. Untuk informasi selengkapnya, lihat contoh kebijakan izin konsol dilIzin diperlukan untuk melihat log komputasi di konsol CodePipeline.

Menyebarkan referensi file spesifikasi

Saat CodePipeline menjalankan tindakan, Anda dapat menentukan file spesifikasi untuk mengonfigurasi penerapan ke instance Anda. File spesifikasi penerapan menentukan apa yang harus diinstal dan kait peristiwa siklus hidup mana yang akan dijalankan sebagai respons terhadap

peristiwa siklus hidup penerapan. File spesifikasi penerapan selalu diformat YAML. File spesifikasi deploy digunakan untuk:

- Petakan file sumber dalam revisi aplikasi Anda ke tujuan mereka pada instance.
- Tentukan izin khusus untuk file yang digunakan.
- Tentukan skrip yang akan dijalankan pada setiap instance pada berbagai tahap proses penyebaran.

File spesifikasi deploy mendukung parameter konfigurasi penerapan tertentu yang didukung oleh CodeDeploy file. AppSpec Anda dapat menggunakan AppSpec file yang ada secara langsung, dan parameter apa pun yang tidak didukung akan diabaikan. Untuk informasi selengkapnya tentang AppSpec file dalam CodeDeploy, lihat referensi file Spesifikasi Aplikasi di Panduan CodeDeployPengguna.

Parameter penyebaran file ditentukan sebagai berikut.

- `files`- File spesifikasi deploy menunjuk `source:` dan `destination:` untuk file penyebaran.
- `scripts`- Acara skrip untuk penyebaran. Dua acara didukung: `BeforeDeploy` dan`AfterDeploy`.
- `hooks`- Siklus hidup terkait untuk acara tersebut. Kait berikut didukung:`ApplicationStop`,,`BeforeInstall`, `AfterInstallApplicationStart`, dan`ValidateService`.

> **Note**
>
> Parameter kait tersedia untuk AppSpec kompatibilitas dengan CodeDeploy dan hanya tersedia dalam versi 0.0 (AppSpec format). Untuk format ini, CodePipeline akan melakukan upaya terbaik pemetaan acara.

Spasi YAMAL yang benar harus digunakan dalam file spesifikasi; jika tidak, kesalahan akan muncul jika lokasi dan jumlah spasi dalam file spesifikasi penerapan tidak benar. Untuk informasi selengkapnya tentang spasi, lihat spesifikasi YAMM.

Contoh file spesifikasi penerapan ada di bawah ini.

```
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
```

Untuk melihat konfigurasi tindakan dengan file spesifikasi deploy yang ditentukan, lihat contoh di. Deklarasi tindakan dengan contoh spesifikasi Deploy

Deklarasi tindakan

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"
        }
    ]
},
```

Deklarasi tindakan dengan contoh spesifikasi 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"
        }
    ]
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Tutorial: Terapkan ke instans Amazon EC2 dengan CodePipeline— Tutorial ini memandu Anda melalui pembuatan instance EC2 di mana Anda akan menyebarkan file skrip, bersama dengan pembuatan pipeline menggunakan tindakan EC2.
- Tindakan Penerapan EC2 gagal dengan pesan kesalahan `No such file`— Topik ini menjelaskan pemecahan masalah untuk kesalahan file tidak ditemukan dengan tindakan EC2.

Referensi tindakan sumber Amazon ECR

Memicu pipeline saat gambar baru didorong ke repositori Amazon ECR. Tindakan ini menyediakan file definisi gambar yang merujuk URI untuk gambar yang didorong ke Amazon ECR. Tindakan sumber ini sering digunakan bersama dengan aksi sumber lain, seperti CodeCommit, untuk memungkinkan lokasi sumber untuk semua artefak sumber lainnya. Untuk informasi selengkapnya, lihat Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR.

Saat Anda menggunakan konsol untuk membuat atau mengedit pipeline, CodePipeline buat EventBridge aturan yang memulai pipeline saat terjadi perubahan di repositori.

> **Note**
>
> Untuk Amazon ECR, Amazon S3, CodeCommit atau sumber, Anda juga dapat membuat penggantian sumber menggunakan entri transformasi input untuk menggunakan `revisionValue` peristiwa EventBridge in untuk pipeline Anda, yang berasal dari variabel peristiwa sumber untuk kunci objek, komit, atau ID gambar Anda. `revisionValue` Untuk informasi selengkapnya, lihat langkah opsional untuk entri transformasi masukan yang disertakan dalam prosedur di bawahTindakan sumber dan sumber daya Amazon ECR EventBridge,Menyambungkan ke tindakan sumber Amazon S3 dengan sumber yang diaktifkan untuk acara, atauCodeCommit tindakan sumber dan EventBridge.

Anda harus sudah membuat repositori Amazon ECR dan mendorong gambar sebelum Anda menghubungkan pipeline melalui tindakan Amazon ECR.

Topik

- Tipe tindakan

- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Izin peran layanan: Tindakan Amazon ECR
- Deklarasi tindakan (contoh Amazon ECR)
- Lihat juga

Tipe tindakan

- Kategori: `Source`
- Pemilik: `AWS`
- Penyedia: `ECR`
- Versi: `1`

Parameter konfigurasi

RepositoryName

Wajib: Ya

Nama repositori Amazon ECR tempat gambar didorong.

ImageTag

Wajib: Tidak

Tag yang digunakan untuk gambar.

> **Note**
>
> Jika nilai untuk tidak `ImageTag` ditentukan, nilai default ke. `latest`

Artefak masukan

- Jumlah artefak: `0`

- Deskripsi: Artefak masukan tidak berlaku untuk jenis tindakan ini.

Artefak keluaran

- Jumlah artefak: 1
- Deskripsi: Tindakan ini menghasilkan artefak yang berisi `imageDetail.json` file yang berisi URI untuk gambar yang memicu eksekusi pipeline. Untuk informasi tentang `imageDetail.json` file, lihatFile ImageDetail.json untuk tindakan penerapan Amazon ECS blue/green.

Variabel keluaran

Ketika dikonfigurasi, tindakan ini menghasilkan variabel yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir dalam pipeline. Tindakan ini menghasilkan variabel yang dapat dilihat sebagai variabel keluaran, bahkan jika tindakan tidak memiliki namespace. Anda mengonfigurasi tindakan dengan namespace untuk membuat variabel-variabel tersebut tersedia untuk konfigurasi tindakan hilir.

Untuk informasi selengkapnya, lihat Referensi variabel.

RegistryId

ID AWS akun yang terkait dengan registri yang berisi repositori.

RepositoryName

Nama repositori Amazon ECR tempat gambar didorong.

ImageTag

Tag yang digunakan untuk gambar.

> **Note**
>
> Variabel `ImageTag` keluaran bukan keluaran saat revisi sumber diganti

ImageDigest

`sha256`Intisari manifes gambar.

ImageURI

URI untuk gambar.

Izin peran layanan: Tindakan Amazon ECR

Untuk dukungan Amazon ECR, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

```
{
    "Effect": "Allow",
    "Action": [
        "ecr:DescribeImages"
    ],
    "Resource": "resource_ARN"
},
```

Untuk informasi lebih lanjut tentang tindakan ini, lihatReferensi tindakan sumber Amazon ECR.

Deklarasi tindakan (contoh 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"
        }
    ]
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR— Tutorial ini menyediakan contoh file spesifikasi aplikasi dan contoh CodeDeploy aplikasi dan grup penyebaran untuk membuat pipeline dengan sumber ECR dan CodeCommit Amazon yang menyebarkan ke instans Amazon ECS.

ECRBuildAndPublishmembangun referensi tindakan

Tindakan build ini memungkinkan Anda mengotomatiskan pembuatan dan mendorong gambar baru saat terjadi perubahan di sumber Anda. Tindakan ini dibangun berdasarkan lokasi file Docker tertentu dan mendorong gambar. Tindakan build ini tidak sama dengan aksi sumber ECR Amazon

CodePipeline, yang memicu pipeline saat terjadi perubahan di repositori sumber Amazon ECR Anda. Untuk informasi tentang tindakan itu, lihatReferensi tindakan sumber Amazon ECR.

Ini bukan tindakan sumber yang akan memicu pipa. Tindakan ini membangun gambar dan mendorongnya ke repositori gambar Amazon ECR Anda.

Anda harus sudah membuat repositori Amazon ECR dan telah menambahkan Dockerfile ke repositori kode sumber Anda, seperti GitHub, sebelum Anda menambahkan tindakan ke pipeline Anda.

> **Important**
>
> Tindakan ini menggunakan CodeBuild komputasi CodePipeline terkelola untuk menjalankan perintah di lingkungan build. Menjalankan tindakan perintah akan dikenakan biaya terpisah di AWS CodeBuild.

> **Note**
>
> Tindakan ini hanya tersedia untuk jaringan pipa tipe V2.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Izin peran layanan: tindakan ECRBuildAndPublish
- Deklarasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Build`
- Pemilik: `AWS`

- Penyedia: `ECRBuildAndPublish`
- Versi: 1

Parameter konfigurasi

ECRRepositoryNama

Wajib: Ya

Nama repositori Amazon ECR tempat gambar didorong.

DockerFilePath

Wajib: Tidak

Lokasi file Docker yang digunakan untuk membangun gambar. Secara opsional, Anda dapat memberikan lokasi file docker alternatif jika tidak pada tingkat root.

> **Note**
>
> Jika nilai untuk tidak `DockerFilePath` ditentukan, nilai default ke tingkat root repositori sumber.

ImageTags

Wajib: Tidak

Tag yang digunakan untuk gambar. Anda dapat memasukkan beberapa tag sebagai daftar string yang dibatasi koma.

> **Note**
>
> Jika nilai untuk tidak `ImageTags` ditentukan, nilai default ke. `latest`

RegistryType

Wajib: Tidak

Menentukan apakah repositori publik atau pribadi. Nilai yang valid adalah `private | public`.

> **Note**
>
> Jika nilai untuk tidak `RegistryType` ditentukan, nilai default ke. `private`

Artefak masukan

- Jumlah artefak: `1`
- Deskripsi: Artefak yang dihasilkan oleh aksi sumber yang berisi Dockerfile yang diperlukan untuk membangun gambar.

Artefak keluaran

- Jumlah artefak: `0`

Variabel keluaran

Ketika dikonfigurasi, tindakan ini menghasilkan variabel yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir dalam pipeline. Tindakan ini menghasilkan variabel yang dapat dilihat sebagai variabel keluaran, bahkan jika tindakan tidak memiliki namespace. Anda mengonfigurasi tindakan dengan namespace untuk membuat variabel-variabel tersebut tersedia untuk konfigurasi tindakan hilir.

Untuk informasi selengkapnya, lihat Referensi variabel.

ECRImageDigestId

`sha256`Intisari manifes gambar.

ECRRepositoryNama

Nama repositori Amazon ECR tempat gambar didorong.

Izin peran layanan: tindakan **`ECRBuildAndPublish`**

Untuk dukungan `ECRBuildAndPublish` tindakan, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

```
{
    "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": "*"
        }
    ]
}
```

Selain itu, jika belum ditambahkan untuk `Commands` tindakan, tambahkan izin berikut ke peran layanan Anda untuk melihat CloudWatch log.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Cakupkan izin ke tingkat sumber daya pipeline dengan menggunakan izin berbasis sumber daya dalam pernyataan kebijakan peran layanan.

Untuk informasi lebih lanjut tentang tindakan ini, lihat`ECRBuildAndPublish`membangun referensi tindakan.

Deklarasi tindakan

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"
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Tutorial: Buat dan dorong gambar Docker ke Amazon ECR dengan CodePipeline (tipe V2)— Tutorial ini menyediakan contoh Dockerfile dan instruksi untuk membuat pipeline yang mendorong gambar Anda ke ECR pada perubahan ke repositori sumber Anda dan kemudian menyebarkan ke Amazon ECS.

Amazon Elastic Container Service dan CodeDeploy referensi aksi penerapan biru-hijau

Anda dapat mengonfigurasi pipeline AWS CodePipeline yang menyebarkan aplikasi kontainer menggunakan blue/green penerapan. Dalam blue/green penerapan, Anda dapat meluncurkan versi baru aplikasi Anda bersama versi lama, dan Anda dapat menguji versi baru sebelum Anda mengalihkan lalu lintas ke sana. Anda juga dapat memantau proses penerapan dan memutar kembali dengan cepat jika ada masalah.

Pipeline yang telah selesai mendeteksi perubahan pada gambar atau file definisi tugas Anda dan digunakan CodeDeploy untuk merutekan dan menyebarkan lalu lintas ke klaster Amazon ECS dan penyeimbang beban. CodeDeploy membuat pendengar baru pada penyeimbang beban Anda yang dapat menargetkan tugas baru Anda melalui port khusus. Anda juga dapat mengonfigurasi pipeline untuk menggunakan lokasi sumber, seperti CodeCommit repositori, tempat definisi tugas Amazon ECS disimpan.

Sebelum membuat pipeline, Anda harus sudah membuat resource Amazon ECS, CodeDeploy resource, dan load balancer serta grup target. Anda harus sudah menandai dan menyimpan gambar di repositori gambar Anda, dan mengunggah definisi tugas dan AppSpec file ke repositori file Anda.

> **Note**
>
> Topik ini menjelaskan tindakan penerapan Amazon ECS ke CodeDeploy biru/hijau untuk. CodePipeline Untuk informasi referensi tentang tindakan penerapan standar Amazon ECS CodePipeline, lihat. Referensi tindakan penerapan Amazon Elastic Container Service

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Izin peran layanan: tindakan CodeDeployToECS
- Deklarasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `CodeDeployToECS`
- Versi: `1`

Parameter konfigurasi

ApplicationName

Wajib: Ya

Nama aplikasi di CodeDeploy. Sebelum Anda membuat pipeline Anda, Anda harus sudah membuat aplikasi di CodeDeploy.

DeploymentGroupName

Wajib: Ya

Grup penerapan yang ditentukan untuk set tugas Amazon ECS yang Anda buat untuk aplikasi Anda CodeDeploy . Sebelum membuat pipeline, Anda harus sudah membuat grup deployment di CodeDeploy.

TaskDefinitionTemplateArtifact

Wajib: Ya

Nama artefak input yang menyediakan file definisi tugas ke tindakan penerapan. Ini umumnya nama artefak keluaran dari aksi sumber. Saat Anda menggunakan konsol, nama default untuk artefak keluaran aksi sumber adalah`SourceArtifact`.

AppSpecTemplateArtifact

Wajib: Ya

Nama artefak input yang menyediakan AppSpec file untuk tindakan penyebaran. Nilai ini diperbarui saat pipeline Anda berjalan. Ini umumnya nama artefak keluaran dari aksi sumber. Saat Anda menggunakan konsol, nama default untuk artefak keluaran aksi sumber adalah`SourceArtifact`. Untuk `TaskDefinition` dalam AppSpec file, Anda dapat menyimpan teks `<TASK_DEFINITION>` placeholder seperti yang ditunjukkan di sini.

AppSpecTemplatePath

Wajib: Tidak

Nama file AppSpec file yang disimpan di lokasi file sumber pipeline, seperti CodeCommit repositori pipeline Anda. Nama file default adalah`appspec.yaml`. Jika AppSpec file Anda memiliki nama yang sama dan disimpan di tingkat root di repositori file Anda, Anda tidak perlu memberikan nama file. Jika path bukan default, masukkan path dan nama file.

TaskDefinitionTemplatePath

Wajib: Tidak

Nama file definisi tugas yang disimpan di lokasi sumber file pipeline, seperti CodeCommit repositori pipeline Anda. Nama file default adalah`taskdef.json`. Jika file definisi tugas Anda memiliki nama yang sama dan disimpan di tingkat root di repositori file Anda, Anda tidak perlu memberikan nama file. Jika path bukan default, masukkan path dan nama file.

Gambar <Number>ArtifactName

Wajib: Tidak

Nama artefak input yang menyediakan gambar untuk tindakan penyebaran. Ini umumnya artefak keluaran repositori gambar, seperti output dari aksi sumber Amazon ECR.

Nilai yang tersedia untuk `<Number>` adalah 1 hingga 4.

Gambar <Number>ContainerName

Wajib: Tidak

Nama gambar yang tersedia dari repositori gambar, seperti repositori sumber Amazon ECR.

Nilai yang tersedia untuk `<Number>` adalah 1 hingga 4.

Artefak masukan

- Jumlah Artefak: `1 to 5`
- Deskripsi: `CodeDeployToECS` Tindakan pertama-tama mencari file definisi tugas dan AppSpec file di repositori file sumber, selanjutnya mencari gambar di repositori gambar, kemudian secara dinamis menghasilkan revisi baru definisi tugas, dan akhirnya menjalankan AppSpec perintah untuk menerapkan set tugas dan wadah ke cluster.

`CodeDeployToECS`Tindakan mencari `imageDetail.json` file yang memetakan URI gambar ke gambar. Saat Anda melakukan perubahan ke repositori gambar Amazon ECR, tindakan sumber ECR pipeline akan membuat `imageDetail.json` file untuk komit tersebut. Anda juga dapat menambahkan `imageDetail.json` file secara manual untuk pipeline di mana tindakan tidak otomatis. Untuk informasi tentang `imageDetail.json` file, lihatFile ImageDetail.json untuk tindakan penerapan Amazon ECS blue/green.

`CodeDeployToECS`Tindakan secara dinamis menghasilkan revisi baru dari definisi tugas. Pada fase ini, tindakan ini menggantikan placeholder dalam file definisi tugas menjadi URI gambar yang diambil dari file ImageDetail.json. Misalnya, jika Anda menetapkan IMAGE1_NAME sebagai ContainerName parameter Image1, Anda harus menentukan placeholder **<** IMAGE1 _NAME**>** sebagai nilai bidang gambar dalam file definisi tugas Anda. Dalam hal ini, tindakan CodeDeployTo ECS menggantikan placeholder **<** IMAGE1 _NAME**>** menjadi URI gambar aktual yang diambil dari ImageDetail.json dalam artefak yang Anda tentukan sebagai Image1. ArtifactName

Untuk pembaruan definisi tugas, CodeDeploy `AppSpec.yaml` file berisi `TaskDefinition` properti.

```
TaskDefinition: <TASK_DEFINITION>
```

Properti ini akan diperbarui oleh `CodeDeployToECS` tindakan setelah definisi tugas baru dibuat.

<TASK_DEFINITION**>**Untuk nilai `TaskDefinition` bidang, teks placeholder harus. `CodeDeployToECS`Tindakan menggantikan placeholder ini dengan ARN aktual dari definisi tugas yang dihasilkan secara dinamis.

Artefak keluaran

- Jumlah Artefak: `0`
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Izin peran layanan: tindakan **`CodeDeployToECS`**

Untuk `CodeDeployToECS` tindakan (tindakan blue/green deployments), the following are the minimum permissions needed to create pipelines with a CodeDeploy to Amazon ECS blue/green penyebaran.

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"
                    ]
                }
            }
        }
    ]
}
```

Anda dapat memilih untuk menggunakan otorisasi penandaan di Amazon ECS. Dengan ikut serta, Anda harus memberikan izin berikut:. `ecs:TagResource` Untuk informasi selengkapnya tentang cara ikut serta dan menentukan apakah izin diperlukan dan otorisasi tag diberlakukan, lihat Garis waktu otorisasi penandaan di Panduan Pengembang Layanan Penampung Elastis Amazon.

Anda juga harus menambahkan `iam:PassRole` izin untuk menggunakan peran IAM untuk tugas. Untuk informasi selengkapnya, lihat peran IAM eksekusi tugas Amazon ECS dan Peran IAM untuk Tugas.

Anda juga dapat menambahkan `ecs-tasks.amazonaws.com` ke daftar layanan di bawah `iam:PassedToService` kondisi, seperti yang ditunjukkan pada contoh di atas.

Deklarasi tindakan

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"
        }
    ]
}
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR— Tutorial ini memandu Anda melalui pembuatan sumber CodeDeploy daya Amazon ECS yang Anda butuhkan untuk blue/green penerapan. Tutorial ini menunjukkan cara mendorong gambar Docker ke Amazon ECR dan membuat definisi tugas Amazon ECS yang mencantumkan nama gambar Docker, nama kontainer, nama layanan Amazon ECS, dan konfigurasi penyeimbang beban. Tutorial kemudian memandu Anda melalui pembuatan AppSpec file dan pipeline untuk penyebaran Anda.

> **Note**
>
> Topik dan tutorial ini menjelaskan tindakan CodeDeploy/ECS biru/hijau untuk. CodePipeline Untuk informasi tentang tindakan standar ECS CodePipeline, lihat Tutorial: Penerapan Berkelanjutan dengan. CodePipeline

- AWS CodeDeploy Panduan Pengguna - Untuk informasi tentang cara menggunakan penyeimbang beban, pendengar produksi, grup target, dan aplikasi Amazon ECS Anda dalam blue/green penerapan, lihat Tutorial: Menerapkan Layanan Amazon ECS. Informasi referensi di Panduan AWS CodeDeploy Pengguna ini memberikan gambaran umum untuk penerapan biru/hijau dengan Amazon ECS dan. AWS CodeDeploy
- Panduan Pengembang Layanan Amazon Elastic Container - Untuk informasi tentang bekerja dengan gambar dan kontainer Docker, layanan dan cluster ECS, dan set tugas ECS, lihat Apa itu Amazon ECS?

Referensi tindakan penerapan Amazon Elastic Container Service

Anda dapat menggunakan tindakan Amazon ECS untuk menerapkan layanan Amazon ECS dan set tugas. Layanan Amazon ECS adalah aplikasi kontainer yang digunakan ke cluster Amazon ECS. Cluster Amazon ECS adalah kumpulan instance yang meng-host aplikasi container Anda di cloud. Penerapan memerlukan definisi tugas yang Anda buat di Amazon ECS dan file definisi gambar yang CodePipeline digunakan untuk menyebarkan gambar.

> **Important**
>
> Tindakan penerapan standar Amazon ECS untuk CodePipeline membuat revisi definisi tugas sendiri berdasarkan revisi yang digunakan oleh layanan Amazon ECS. Jika Anda membuat revisi baru untuk definisi tugas tanpa memperbarui layanan Amazon ECS, tindakan penerapan akan mengabaikan revisi tersebut.

Sebelum membuat pipeline, Anda harus sudah membuat resource Amazon ECS, menandai dan menyimpan gambar di repositori gambar Anda, dan mengunggah file ke repositori BuildSpec file Anda.

> **Note**
>
> Topik referensi ini menjelaskan tindakan penerapan standar Amazon ECS untuk. CodePipeline Untuk informasi referensi tentang Amazon ECS ke tindakan penerapan CodeDeploy biru/hijau, lihat. CodePipeline Amazon Elastic Container Service dan CodeDeploy referensi aksi penerapan biru-hijau

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Izin peran layanan: Tindakan standar Amazon ECS
- Deklarasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `ECS`
- Versi: `1`

Parameter konfigurasi

ClusterName

Wajib: Ya

Cluster Amazon ECS di Amazon ECS.

ServiceName

Wajib: Ya

Layanan Amazon ECS yang Anda buat di Amazon ECS.

FileName

Wajib: Tidak

Nama file definisi gambar Anda, file JSON yang menjelaskan nama wadah layanan Anda dan gambar dan tag. Anda menggunakan file ini untuk penerapan standar ECS. Untuk informasi selengkapnya, lihat Artefak masukan dan file imagedefinitions.json untuk tindakan penerapan standar Amazon ECS.

DeploymentTimeout

Wajib: Tidak

Batas waktu tindakan penerapan Amazon ECS dalam hitungan menit. Batas waktu dapat dikonfigurasi hingga batas waktu default maksimum untuk tindakan ini. Contoh:

```
"DeploymentTimeout": "15"
```

Artefak masukan

- Jumlah artefak: 1
- Deskripsi: Tindakan mencari `imagedefinitions.json` file di repositori file sumber untuk pipeline. Dokumen definisi gambar adalah file JSON yang menjelaskan nama wadah Amazon ECS Anda serta gambar dan tag. CodePipeline menggunakan file untuk mengambil gambar dari repositori gambar Anda seperti Amazon ECR. Anda dapat menambahkan `imagedefinitions.json` file secara manual untuk pipeline di mana tindakan tidak otomatis. Untuk informasi tentang `imagedefinitions.json` file, lihatfile imagedefinitions.json untuk tindakan penerapan standar Amazon ECS.

  Tindakan ini membutuhkan gambar yang sudah ada yang telah didorong ke repositori gambar Anda. Karena pemetaan gambar disediakan oleh `imagedefinitions.json` file, tindakan tidak mengharuskan sumber Amazon ECR disertakan sebagai tindakan sumber dalam pipeline.

Artefak keluaran

- Jumlah artefak: `0`
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Izin peran layanan: Tindakan standar Amazon ECS

Untuk Amazon ECS, berikut ini adalah izin minimum yang diperlukan untuk membuat pipeline dengan tindakan penerapan Amazon ECS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "TaskDefinitionPermissions",
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeTaskDefinition",
                "ecs:RegisterTaskDefinition"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "ECSServicePermissions",
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeServices",
                "ecs:UpdateService"
            ],
            "Resource": [
                "arn:aws:ecs:*:111122223333:service/[[clusters]]/*"
            ]
        },
        {
            "Sid": "ECSTagResource",
            "Effect": "Allow",
            "Action": [
                "ecs:TagResource"
            ],
            "Resource": [
                "arn:aws:ecs:*:111122223333:task-definition/
[[taskDefinitions]]:*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "ecs:CreateAction": [
                        "RegisterTaskDefinition"
                    ]
                }
            }
        },
        {
            "Sid": "IamPassRolePermissions",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::111122223333:role/[[passRoles]]"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ecs.amazonaws.com",
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Anda dapat memilih untuk menggunakan otorisasi penandaan di Amazon ECS. Dengan ikut serta, Anda harus memberikan izin berikut:. `ecs:TagResource` Untuk informasi selengkapnya tentang cara ikut serta dan menentukan apakah izin diperlukan dan otorisasi tag diberlakukan, lihat Garis waktu otorisasi penandaan di Panduan Pengembang Layanan Penampung Elastis Amazon.

Anda harus menambahkan `iam:PassRole` izin untuk menggunakan peran IAM untuk tugas. Untuk informasi selengkapnya, lihat peran IAM eksekusi tugas Amazon ECS dan Peran IAM untuk Tugas. Gunakan teks kebijakan berikut.

Deklarasi tindakan

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"
        }
    ]
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Lihat Tutorial: Buat dan dorong gambar Docker ke Amazon ECR dengan CodePipeline (tipe V2) tutorial yang menunjukkan cara menggunakan tindakan ECRBuildand Publikasikan untuk mendorong gambar dan kemudian menggunakan tindakan standar ECS untuk menyebarkan ke Amazon ECS.
- Tutorial: Continuous Deployment with CodePipeline - Tutorial ini menunjukkan cara membuat Dockerfile yang Anda simpan di repositori file sumber seperti. CodeCommit Selanjutnya, tutorial menunjukkan cara menggabungkan CodeBuild BuildSpec file yang membangun dan mendorong gambar Docker Anda ke Amazon ECR dan membuat file imagedefinitions.json Anda. Terakhir, Anda membuat layanan Amazon ECS dan definisi tugas, lalu Anda membuat pipeline dengan tindakan penerapan Amazon ECS.

  > **Note**
  >
  > Topik dan tutorial ini menjelaskan tindakan penerapan standar Amazon ECS untuk. CodePipeline Untuk informasi tentang Amazon ECS ke tindakan penerapan CodeDeploy biru/hijau di, lihat. CodePipeline Tutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR

- Panduan Pengembang Layanan Amazon Elastic Container - Untuk informasi tentang bekerja dengan gambar dan kontainer Docker, layanan dan cluster Amazon ECS, dan set tugas Amazon ECS, lihat Apa itu Amazon ECS?

Referensi tindakan penerapan Amazon Elastic Kubernetes Service EKS

Anda dapat menggunakan `EKSDeploy` tindakan untuk menerapkan layanan Amazon EKS. Penerapan membutuhkan file manifes Kubernetes yang CodePipeline digunakan untuk menyebarkan gambar.

Sebelum Anda membuat pipeline, Anda harus sudah membuat resource Amazon EKS dan telah menyimpan gambar di repositori gambar Anda. Secara opsional, Anda dapat memberikan informasi VPC untuk cluster Anda.

> **Important**
>
> Tindakan ini menggunakan CodeBuild komputasi CodePipeline terkelola untuk menjalankan perintah di lingkungan build. Menjalankan tindakan perintah akan dikenakan biaya terpisah di AWS CodeBuild.

> **Note**
>
> Tindakan EKS penerapan hanya tersedia untuk saluran pipa tipe V2.

Tindakan EKS mendukung kluster EKS publik dan swasta. Cluster pribadi adalah tipe yang direkomendasikan oleh EKS; Namun, kedua jenis didukung.

Tindakan EKS didukung untuk tindakan lintas akun. Untuk menambahkan tindakan EKS lintas akun, tambahkan `actionRoleArn` dari akun target Anda dalam deklarasi tindakan.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Variabel-variabel lingkungan
- Variabel keluaran
- Izin kebijakan peran layanan
- Deklarasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `EKS`
- Versi: `1`

Parameter konfigurasi

ClusterName

Wajib: Ya

Cluster Amazon EKS di Amazon EKS.

Opsi di bawah Helm

Berikut ini adalah opsi yang tersedia ketika Helm adalah alat penyebaran yang dipilih.

HelmReleaseName

Wajib: Ya (Diperlukan hanya untuk tipe Helm)

Nama rilis untuk penerapan Anda.

HelmChartLocation

Wajib: Ya (Diperlukan hanya untuk tipe Helm)

Lokasi bagan untuk penerapan Anda.

HelmValuesFiles

Wajib: Tidak (Opsional hanya untuk tipe Helm)

Untuk mengganti file nilai helm, masukkan file nilai helm yang dipisahkan koma di lokasi bagan helm.

Opsi di bawah Kubectl

Berikut ini adalah opsi yang tersedia ketika Kubectl adalah alat penerapan yang dipilih.

ManifestFiles

Wajib: Ya (Diperlukan hanya untuk tipe Kubectl)

Nama file manifes Anda, file teks yang menjelaskan nama kontainer layanan Anda dan gambar dan tag. Anda menggunakan file ini untuk membuat parameter URI gambar Anda dan informasi lainnya. Anda dapat menggunakan variabel lingkungan untuk tujuan ini.

Anda menyimpan file ini di repositori sumber untuk pipeline Anda.

Namespace

Wajib: Tidak

Namepsace kubernetes yang akan digunakan dalam atau perintah. `kubectl` `helm`

Subnet

Wajib: Tidak

Subnet untuk VPC untuk cluster Anda. Ini adalah bagian dari VPC yang sama yang dilampirkan ke cluster Anda. Anda juga dapat menyediakan subnet yang belum dilampirkan ke cluster Anda dan menentukannya di sini.

SecurityGroupIds

Wajib: Tidak

Grup keamanan untuk VPC untuk cluster Anda. Ini adalah bagian dari VPC yang sama yang dilampirkan ke cluster Anda. Anda juga dapat menyediakan grup keamanan yang belum dilampirkan ke cluster Anda dan menentukannya di sini.

Artefak masukan

- Jumlah artefak: `1`
- Deskripsi: Tindakan mencari file manifes Kubernetes atau bagan Helm di repositori file sumber untuk pipeline. Jika Anda ingin menggunakan bagan helm dalam format.tgz yang disimpan dalam bucket S3, Anda dapat melakukannya dengan mengonfigurasi S3 sebagai tindakan sumber Anda. Bucket/Key Misalnya, kunci objek yang disediakan adalah`my-chart-0.1.0.tgz`.

Artefak keluaran

- Jumlah artefak: `0`
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Variabel-variabel lingkungan

Digunakan untuk mengganti variabel seperti repositori gambar atau tag gambar dalam file manifes atau file nilai bagan helm.

Key

Kunci dalam pasangan variabel lingkungan kunci-nilai, seperti. `$IMAGE_TAG`

Nilai

Nilai untuk pasangan kunci-nilai, seperti. `v1.0` Nilai dapat diparameterisasi dengan variabel keluaran dari tindakan pipa atau variabel pipa. Misalnya, pipeline dapat memiliki ECRBuild AndPublish tindakan yang membuat gambar ECR dengan`${codepipeline.PipelineExecutionId}`, dan tindakan EKS dapat menggunakan gambar ini menggunakan `${codepipeline.PipelineExecutionId}` sebagai nilai variabel lingkungan.

Variabel keluaran

EKSClusterNama

Cluster Amazon EKS di Amazon EKS.

Izin kebijakan peran layanan

Untuk menjalankan tindakan ini, izin berikut harus tersedia dalam kebijakan peran layanan pipeline Anda.

- Tindakan EC2: Saat CodePipeline menjalankan tindakan, izin instans EC2 diperlukan. Perhatikan bahwa ini tidak sama dengan peran instans EC2 yang diperlukan saat Anda membuat kluster EKS Anda.

  Jika Anda menggunakan peran layanan yang ada, untuk menggunakan tindakan ini, Anda perlu menambahkan izin berikut untuk peran layanan.
  - EC2: CreateNetworkInterface
  - EC2: DescribeDhcpOptions
  - EC2: DescribeNetworkInterfaces
  - EC2: DeleteNetworkInterface
  - EC2: DescribeSubnets
  - EC2: DescribeSecurityGroups
  - EC2: DescribeVpcs
- Tindakan EKS: Saat CodePipeline menjalankan tindakan, izin kluster EKS diperlukan. Perhatikan bahwa ini tidak sama dengan peran cluster IAM EKS yang diperlukan saat Anda membuat cluster EKS Anda.

Jika Anda menggunakan peran layanan yang ada, untuk menggunakan tindakan ini, Anda perlu menambahkan izin berikut untuk peran layanan.

- eks: DescribeCluster

- Tindakan aliran log: Saat CodePipeline menjalankan tindakan, CodePipeline buat grup log menggunakan nama pipeline sebagai berikut. Ini memungkinkan Anda untuk mengurangi izin untuk mencatat sumber daya menggunakan nama pipeline.

```
/aws/codepipeline/MyPipelineName
```

Jika Anda menggunakan peran layanan yang ada, untuk menggunakan tindakan ini, Anda perlu menambahkan izin berikut untuk peran layanan.

- log: CreateLogGroup
- log: CreateLogStream
- log: PutLogEvents

Dalam pernyataan kebijakan peran layanan, cakupkan izin ke tingkat sumber daya seperti yang ditunjukkan pada contoh berikut.

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:*"
            ]
        }
    ]
}
```

Untuk melihat log di konsol menggunakan halaman dialog detail tindakan, izin untuk melihat log harus ditambahkan ke peran konsol. Untuk informasi selengkapnya, lihat contoh kebijakan izin konsol diIzin diperlukan untuk melihat log komputasi di konsol CodePipeline.

Menambahkan peran layanan sebagai entri akses untuk klaster Anda

Setelah izin tersedia dalam kebijakan peran layanan pipeline, Anda mengonfigurasi izin klaster dengan menambahkan peran CodePipeline layanan sebagai entri akses untuk klaster Anda.

Anda juga dapat menggunakan peran tindakan yang memiliki izin yang diperbarui. Untuk informasi lebih lanjut, lihat contoh tutorial diLangkah 4: Buat entri akses untuk peran CodePipeline layanan.

Deklarasi tindakan

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"
        }
    ]
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Lihat tutorial Tutorial: Terapkan ke Amazon EKS dengan CodePipeline yang menunjukkan cara membuat cluster EKS dan file manifes Kubernetes untuk menambahkan tindakan ke pipeline Anda.

AWS Lambda menyebarkan referensi tindakan

Anda menggunakan tindakan AWS Lambda penerapan untuk mengelola penerapan kode aplikasi untuk penerapan tanpa server Anda. Anda dapat menerapkan fungsi dan menggunakan strategi penyebaran untuk penyebaran lalu lintas sebagai berikut:

- Penyebaran kenari dan linier untuk perpindahan lalu lintas
- Semua penyebaran sekaligus

> **Note**
>
> Tindakan ini hanya didukung untuk pipeline tipe V2.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Izin kebijakan peran layanan untuk tindakan penerapan Lambda
- Deklarasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `Lambda`
- Versi: `1`

Parameter konfigurasi

FunctionName

Wajib: Ya

Nama fungsi yang Anda buat di Lambda, seperti. `MyLambdaFunction`

Anda harus sudah membuat versi.

FunctionAlias

Wajib: Tidak

Alias fungsi yang Anda buat di Lambda dan merupakan fungsi yang akan digunakan, seperti. `live` Alias harus ada dan memiliki satu versi di belakangnya ketika eksekusi tindakan dimulai. (Ini akan menjadi versi target rollback.)

Jika tidak disediakan, tindakan menyebarkan artefak sumber ke `$LATEST` dan membuat versi baru. Dalam kasus penggunaan ini, strategi penerapan dan opsi versi target tidak tersedia.

PublishedTargetVersion

Wajib: Tidak

Versi Fungsi Lambda yang diinginkan untuk digunakan. FunctionAlias Ini bisa berupa variabel pipeline atau level aksi, seperti`#{variables.lambdaTargetVersion}`. Versi harus dipublikasikan saat eksekusi tindakan dimulai.

Diperlukan jika tidak ada artefak input yang disediakan.

DeployStrategy

Wajib: Tidak (Default adalah`AllAtOnce`)

Menentukan tingkat yang diadopsi oleh tindakan penerapan Lambda untuk mengalihkan lalu lintas dari versi asli fungsi Lambda ke versi baru. FunctionAlias Strategi penyebaran yang tersedia adalah kenari atau linier. Format yang diterima:

- `AllAtOnce` -

  Menggeser semua lalu lintas ke fungsi Lambda yang diperbarui sekaligus.

Jika tidak ditentukan, defaultnya adalah`AllAtOnce`)

- `Canary10Percent5Minutes`- Menggeser 10 persen lalu lintas pada kenaikan pertama. 90 persen sisanya di-deploy lima menit kemudian.

  Nilai untuk persentase dan menit dapat diubah.

- `Linear10PercentEvery1Minute`- Menggeser 10 persen lalu lintas setiap menit sampai semua lalu lintas bergeser.

  Nilai untuk persentase dan menit dapat diubah.

Pertimbangan berikut berlaku untuk bidang ini:

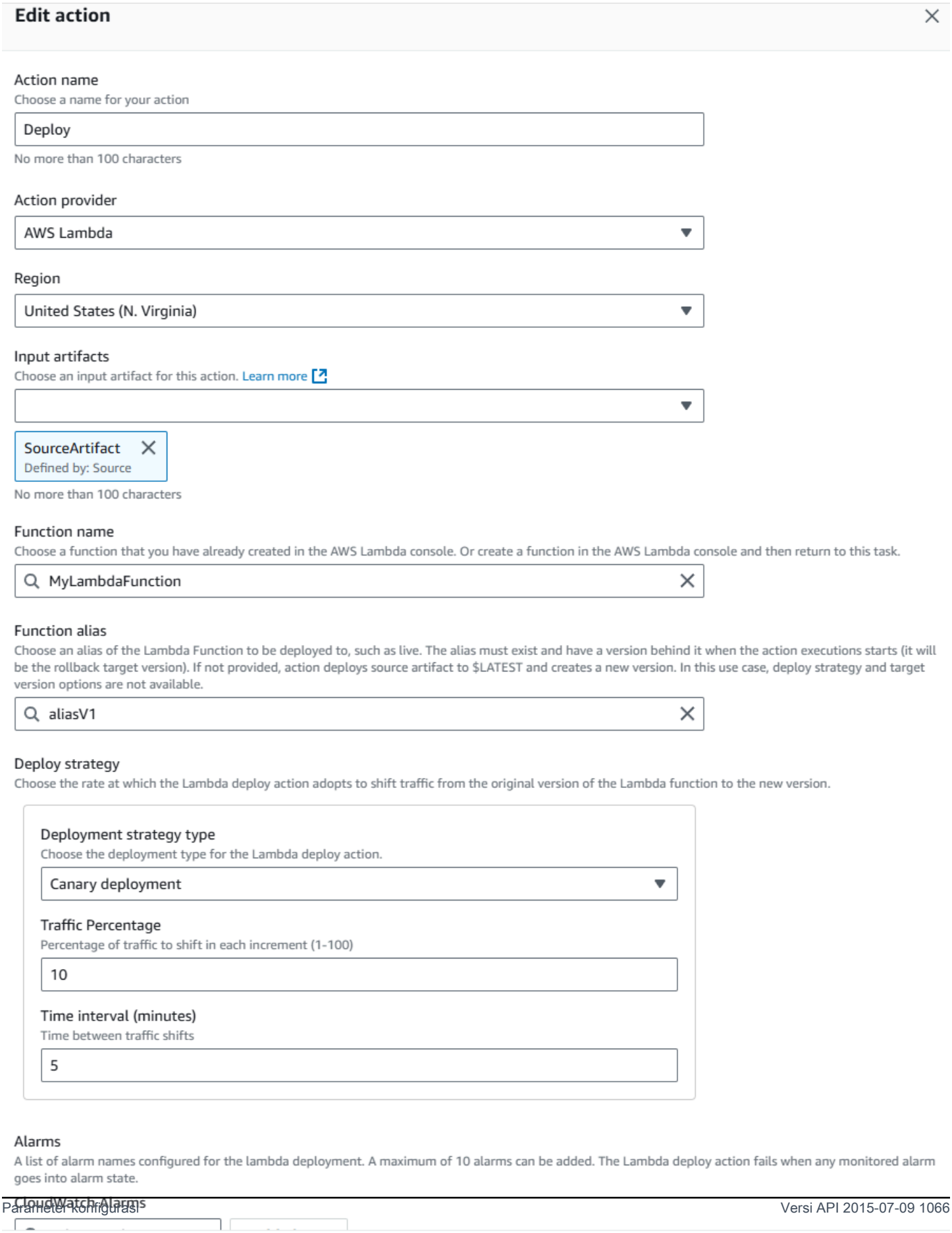
- Total waktu tunggu maksimum adalah 2 hari.
- Hanya tersedia jika FunctionAliasdisediakan.

Alarm

Wajib: Tidak

Daftar nama alarm yang dipisahkan koma yang dikonfigurasi untuk penyebaran Lambda. Maksimal 10 alarm dapat ditambahkan. Tindakan gagal saat alarm yang dipantau masuk ke status ALARM.

Gambar berikut menunjukkan contoh halaman Edit untuk tindakan tersebut.


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

Artefak masukan

- Jumlah artefak: 1
- Deskripsi: File yang disediakan, jika ada, untuk mendukung tindakan skrip selama penyebaran.

Artefak keluaran

- Jumlah artefak: 0
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Variabel keluaran

Ketika dikonfigurasi, tindakan ini menghasilkan variabel yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir dalam pipeline. Tindakan ini menghasilkan variabel yang dapat dilihat sebagai variabel keluaran, bahkan jika tindakan tidak memiliki namespace. Anda mengonfigurasi tindakan dengan namespace untuk membuat variabel-variabel tersebut tersedia untuk konfigurasi tindakan hilir.

Untuk informasi selengkapnya, lihat Referensi variabel.

FunctionVersion

Versi fungsi Lambda baru yang digunakan.

Izin kebijakan peran layanan untuk tindakan penerapan Lambda

Saat CodePipeline menjalankan tindakan, peran CodePipeline layanan memerlukan izin berikut, yang dicakup dengan tepat untuk akses dengan hak istimewa paling sedikit.

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:*"
            ]
        }
    ]
}
```

Deklarasi tindakan

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"
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Tutorial: Penerapan fungsi Lambda dengan CodePipeline— Tutorial ini memandu Anda melalui pembuatan contoh fungsi Lambda di mana Anda akan membuat alias dan versi, menambahkan fungsi Lambda zip ke lokasi sumber Anda, dan menjalankan tindakan Lambda di pipeline Anda.

Referensi tindakan penerapan Amazon S3

Anda menggunakan tindakan penerapan Amazon S3 untuk menyebarkan file ke bucket Amazon S3 untuk hosting atau arsip situs web statis. Anda dapat menentukan apakah akan mengekstrak file penerapan sebelum mengunggah ke bucket.

> **Note**
>
> Topik referensi ini menjelaskan tindakan penerapan Amazon S3 CodePipeline di mana platform penerapan adalah bucket Amazon S3 yang dikonfigurasi untuk hosting. Untuk informasi referensi tentang tindakan sumber Amazon S3 di CodePipeline, lihat. Referensi tindakan sumber Amazon S3

Topik

- Tipe tindakan
- Parameter konfigurasi

- Artefak masukan
- Artefak keluaran
- Izin peran layanan: Tindakan penerapan S3
- Contoh konfigurasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `S3`
- Versi: `1`

Parameter konfigurasi

BucketName

Wajib: Ya

Nama bucket Amazon S3 tempat file akan digunakan.

Ekstrak

Wajib: Ya

Jika benar, menentukan bahwa file yang akan diekstraksi sebelum meng-upload. Jika tidak, file aplikasi tetap di-zip untuk diunggah, seperti dalam kasus situs web statis yang dihosting. Jika salah, maka `ObjectKey` diperlukan.

ObjectKey

Bersyarat. Diperlukan jika `Extract` = salah

Nama kunci objek Amazon S3 yang secara unik mengidentifikasi objek di bucket S3.

KMSEncryptionkeYarn

Wajib: Tidak

ARN dari kunci AWS KMS enkripsi untuk bucket host. `KMSEncryptionKeyARN`Parameter mengenkripsi artefak yang diunggah dengan yang disediakan. AWS KMS key Untuk kunci KMS, Anda dapat menggunakan ID kunci, kunci ARN, atau alias ARN.

> **Note**
>
> Alias hanya dikenali di akun yang membuat kunci KMS. Untuk tindakan lintas akun, Anda hanya dapat menggunakan ID kunci atau ARN kunci untuk mengidentifikasi kunci. Tindakan lintas akun melibatkan penggunaan peran dari akun lain (accountB), sehingga menentukan ID kunci akan menggunakan kunci dari akun lain (accounTB).

> **Important**
>
> CodePipeline hanya mendukung tombol KMS simetris. Jangan gunakan kunci KMS asimetris untuk mengenkripsi data di bucket S3 Anda.

CannedACL

Wajib: Tidak

`CannedACL`Parameter menerapkan ACL kalengan yang ditentukan ke objek yang diterapkan ke Amazon S3. Ini menimpa ACL yang ada yang diterapkan ke objek.

CacheControl

Wajib: Tidak

`CacheControl`Parameter mengontrol perilaku caching requests/responses untuk objek di bucket. Untuk daftar nilai yang valid, lihat bidang `Cache-Control`header untuk operasi HTTP. Untuk memasukkan beberapa nilai`CacheControl`, gunakan koma di antara setiap nilai. Anda dapat menambahkan spasi setelah setiap koma (opsional), seperti yang ditunjukkan dalam contoh ini untuk CLI:

```
"CacheControl": "public, max-age=0, no-transform"
```

Artefak masukan

- Jumlah Artefak: 1
- Deskripsi: File untuk penyebaran atau arsip diperoleh dari repositori sumber, di-zip, dan diunggah oleh. CodePipeline

Artefak keluaran

- Jumlah artefak: `0`
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Izin peran layanan: Tindakan penerapan S3

Untuk dukungan tindakan penerapan S3, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

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"
        }
      }
```

```
    }
  ]
}
```

Untuk dukungan tindakan penerapan S3, jika objek S3 memiliki tag, Anda juga harus menambahkan izin berikut ke pernyataan kebijakan Anda:

```
"s3:GetObjectTagging",
"s3:GetObjectVersionTagging",
"s3:PutObjectTagging"
```

Contoh konfigurasi tindakan

Berikut ini menunjukkan contoh untuk konfigurasi tindakan.

Contoh konfigurasi saat **`Extract`** diatur ke **`false`**

Contoh berikut menunjukkan konfigurasi tindakan default ketika tindakan dibuat dengan `Extract` bidang diatur ke`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"
        }
    ]
},
```

Contoh konfigurasi saat **`Extract`** diatur ke **`true`**

Contoh berikut menunjukkan konfigurasi tindakan default ketika tindakan dibuat dengan `Extract` bidang diatur ke`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"
        }
    ]
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Tutorial: Membuat pipeline yang menggunakan Amazon S3 sebagai penyedia penerapan— Tutorial ini memandu Anda melalui dua contoh untuk membuat pipeline dengan tindakan penerapan S3. Anda mengunduh file sampel, mengunggah file ke CodeCommit repositori, membuat bucket S3, dan mengonfigurasi bucket untuk hosting. Selanjutnya, Anda menggunakan CodePipeline konsol untuk membuat pipeline dan menentukan konfigurasi penerapan Amazon S3.
- Referensi tindakan sumber Amazon S3— Referensi tindakan ini memberikan informasi referensi dan contoh untuk tindakan sumber Amazon S3 di. CodePipeline

Referensi tindakan sumber Amazon S3

Memicu pipeline saat objek baru diunggah ke bucket dan kunci objek yang dikonfigurasi.

> **Note**
>
> Topik referensi ini menjelaskan tindakan sumber Amazon S3 CodePipeline di mana lokasi sumber adalah bucket Amazon S3 yang dikonfigurasi untuk pembuatan versi. Untuk informasi referensi tentang tindakan penerapan Amazon S3, lihat. CodePipeline Referensi tindakan penerapan Amazon S3

Anda dapat membuat bucket Amazon S3 untuk digunakan sebagai lokasi sumber untuk file aplikasi Anda.

> **Note**
>
> Saat membuat bucket sumber, pastikan Anda mengaktifkan pembuatan versi di bucket. Jika Anda ingin menggunakan bucket Amazon S3 yang sudah ada, lihat Menggunakan pembuatan versi untuk mengaktifkan pembuatan versi pada bucket yang ada.

Jika Anda menggunakan konsol untuk membuat atau mengedit pipeline, CodePipeline buat EventBridge aturan yang memulai pipeline saat terjadi perubahan di bucket sumber S3.

> **Note**
>
> Untuk Amazon ECR, Amazon S3, CodeCommit atau sumber, Anda juga dapat membuat penggantian sumber menggunakan entri transformasi input untuk menggunakan `revisionValue` peristiwa EventBridge in untuk pipeline Anda, yang berasal dari variabel peristiwa sumber untuk kunci objek, komit, atau ID gambar Anda. `revisionValue` Untuk informasi selengkapnya, lihat langkah opsional untuk entri transformasi masukan yang disertakan dalam prosedur di bawahTindakan sumber dan sumber daya Amazon ECR EventBridge,Menyambungkan ke tindakan sumber Amazon S3 dengan sumber yang diaktifkan untuk acara, atauCodeCommit tindakan sumber dan EventBridge.

Anda harus sudah membuat bucket sumber Amazon S3 dan mengunggah file sumber sebagai file ZIP tunggal sebelum Anda menghubungkan pipeline melalui tindakan Amazon S3.

> **Note**
>
> Jika Amazon S3 adalah penyedia sumber untuk pipeline Anda, Anda dapat mem-zip file sumber atau file ke dalam satu .zip dan mengunggah.zip ke bucket sumber Anda. Anda juga dapat mengunggah satu file yang tidak di-zip; namun, tindakan hilir yang mengharapkan file.zip akan gagal.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Izin peran layanan: Tindakan sumber S3
- Deklarasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Source`
- Pemilik: `AWS`
- Penyedia: `S3`
- Versi: `1`

Parameter konfigurasi

S3Bucket

Wajib: Ya

Nama bucket Amazon S3 tempat perubahan sumber harus dideteksi.

S3 ObjectKey

Wajib: Ya

Nama kunci objek Amazon S3 tempat perubahan sumber akan dideteksi.

AllowOverrideForS3 ObjectKey

Wajib: Tidak

`AllowOverrideForS3ObjectKey`mengontrol apakah penggantian sumber dari `StartPipelineExecution` dapat mengganti yang sudah dikonfigurasi `S3ObjectKey` dalam aksi sumber. Untuk informasi selengkapnya tentang penggantian sumber dengan Kunci Objek S3, lihat. Mulai pipeline dengan penggantian revisi sumber

> **Important**
>
> Jika Anda menghilangkan`AllowOverrideForS3ObjectKey`, CodePipeline default kemampuan untuk mengganti S3 ObjectKey dalam aksi sumber dengan menyetel parameter ini ke. `false`

Nilai yang valid untuk parameter ini:

- `true`: Jika disetel, Kunci Objek S3 yang telah dikonfigurasi sebelumnya dapat diganti dengan penggantian revisi sumber selama eksekusi pipeline.

> **Note**
>
> Jika Anda bermaksud mengizinkan semua CodePipeline pengguna kemampuan untuk mengganti Kunci Objek S3 yang telah dikonfigurasi sebelumnya saat memulai eksekusi pipeline baru, Anda harus menyetelnya. `AllowOverrideForS3ObjectKey` `true`

- `false`:

  Jika disetel, tidak CodePipeline akan mengizinkan Kunci Objek S3 diganti menggunakan penggantian revisi sumber. Ini juga merupakan nilai default untuk parameter ini.

PollForSourceChanges

Wajib: Tidak

`PollForSourceChanges`mengontrol apakah CodePipeline polling bucket sumber Amazon S3 untuk perubahan sumber. Kami menyarankan Anda menggunakan CloudWatch Acara dan CloudTrail untuk mendeteksi perubahan sumber sebagai gantinya. Untuk informasi selengkapnya tentang mengonfigurasi CloudWatch Acara, lihat Migrasi jaringan pemungutan suara dengan sumber dan jejak CloudTrail S3 (CLI) atauMigrasikan jalur pemungutan suara dengan sumber dan CloudTrail jejak S3 (templat)CloudFormation.

> **Important**
>
> Jika ingin mengonfigurasi CloudWatch Peristiwa, Anda harus mengatur `PollForSourceChanges` `false` untuk menghindari eksekusi pipeline duplikat.

Nilai yang valid untuk parameter ini:

- `true`: Jika disetel, CodePipeline polling lokasi sumber Anda untuk perubahan sumber.

> **Note**
>
> Jika Anda menghilangkan`PollForSourceChanges`, CodePipeline default untuk polling lokasi sumber Anda untuk perubahan sumber. Perilaku ini sama seperti jika `PollForSourceChanges` disertakan dan disetel ke`true`.

- `false`: Jika disetel CodePipeline , jangan polling lokasi sumber Anda untuk perubahan sumber. Gunakan pengaturan ini jika Anda ingin mengonfigurasi aturan CloudWatch Peristiwa untuk mendeteksi perubahan sumber.

Artefak masukan

- Jumlah Artefak: `0`
- Deskripsi: Artefak masukan tidak berlaku untuk jenis tindakan ini.

Artefak keluaran

- Jumlah artefak: `1`
- Deskripsi: Menyediakan artefak yang tersedia di bucket sumber yang dikonfigurasi untuk terhubung ke pipeline. Artefak yang dihasilkan dari bucket adalah artefak keluaran untuk aksi Amazon S3. Metadata objek Amazon S3 (ETag dan ID versi) ditampilkan CodePipeline sebagai revisi sumber untuk eksekusi pipeline yang dipicu.

Variabel keluaran

Ketika dikonfigurasi, tindakan ini menghasilkan variabel yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir dalam pipeline. Tindakan ini menghasilkan variabel yang dapat dilihat sebagai variabel keluaran, bahkan jika tindakan tidak memiliki namespace. Anda mengonfigurasi tindakan dengan namespace untuk membuat variabel-variabel tersebut tersedia untuk konfigurasi tindakan hilir.

Untuk informasi lebih lanjut tentang variabel di CodePipeline, lihatReferensi variabel.

BucketName

Nama bucket Amazon S3 terkait dengan perubahan sumber yang memicu pipeline.

ETag

Tag entitas untuk objek yang terkait dengan perubahan sumber yang memicu pipeline. ETag Ini adalah MD5 hash dari objek. ETag hanya mencerminkan perubahan pada isi objek, bukan metadatanya.

ObjectKey

Nama kunci objek Amazon S3 terkait dengan perubahan sumber yang memicu pipeline.

VersionId

ID versi untuk versi objek yang terkait dengan perubahan sumber yang memicu pipeline.

Izin peran layanan: Tindakan sumber S3

Untuk dukungan tindakan sumber S3, tambahkan berikut ini ke pernyataan kebijakan Anda:

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"
        }
      }
    }
  ]
}
```

Deklarasi tindakan

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": []
        }
    ]
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Tutorial: Buat pipeline sederhana (ember S3)— Tutorial ini menyediakan contoh file spesifikasi aplikasi dan contoh CodeDeploy aplikasi dan kelompok penyebaran. Gunakan tutorial ini untuk membuat pipeline dengan sumber Amazon S3 yang menyebarkan ke instans Amazon EC2.

AWS AppConfig menyebarkan referensi tindakan

AWS AppConfig adalah kemampuan AWS Systems Manager. AppConfig mendukung penerapan terkontrol ke aplikasi dari berbagai ukuran dan mencakup pemeriksaan dan pemantauan validasi bawaan. Anda dapat menggunakannya AppConfig dengan aplikasi yang dihosting di instans Amazon EC2, wadah AWS Lambda, aplikasi seluler, atau perangkat IoT.

Tindakan penerapan adalah AWS CodePipeline tindakan yang *`AppConfig`* menyebarkan konfigurasi yang disimpan di lokasi sumber pipeline Anda ke profil AppConfig aplikasi, lingkungan, dan konfigurasi yang ditentukan. Ini menggunakan preferensi yang didefinisikan dalam strategi AppConfig penyebaran.

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `AppConfig`
- Versi: 1

Parameter konfigurasi

Aplikasi

Wajib: Ya

ID AWS AppConfig aplikasi dengan rincian untuk konfigurasi dan penyebaran Anda.

Lingkungan

Wajib: Ya

ID AWS AppConfig lingkungan tempat konfigurasi dikerahkan.

ConfigurationProfile

Wajib: Ya

ID profil AWS AppConfig konfigurasi yang akan digunakan.

InputArtifactConfigurationPath

Wajib: Ya

Jalur file dari data konfigurasi dalam artefak input untuk digunakan.

DeploymentStrategy

Wajib: Tidak

Strategi AWS AppConfig penyebaran yang akan digunakan untuk penyebaran.

Artefak masukan

- Jumlah artefak: 1
- Deskripsi: Artefak masukan untuk tindakan penyebaran.

Artefak keluaran

Tidak berlaku.

Izin peran layanan: tindakan **AppConfig**

Saat CodePipeline menjalankan tindakan, kebijakan peran CodePipeline layanan memerlukan izin berikut, yang dicakup dengan tepat ke tingkat sumber daya untuk mempertahankan akses dengan hak istimewa paling sedikit.

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"
        }
    ]
}
```

Contoh konfigurasi tindakan

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"
        }
    ]
```

```
}
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- AWS AppConfig— Untuk informasi tentang AWS AppConfig penerapan, lihat AWS Systems Manager Panduan Pengguna.
- Tutorial: Buat pipeline yang digunakan AWS AppConfig sebagai penyedia penyebaran— Tutorial ini membuat Anda mulai menyiapkan file konfigurasi dan AppConfig sumber daya penyebaran sederhana, dan menunjukkan cara menggunakan konsol untuk membuat pipeline dengan tindakan AWS AppConfig penerapan.

CloudFormation menyebarkan referensi tindakan

Mengeksekusi operasi pada CloudFormation tumpukan. Tumpukan adalah kumpulan sumber AWS daya yang dapat Anda kelola sebagai satu unit. Sumber daya dalam tumpukan ditentukan oleh CloudFormation template tumpukan. Set perubahan membuat perbandingan yang dapat dilihat tanpa mengubah tumpukan asli. Untuk informasi tentang jenis CloudFormation tindakan yang dapat dilakukan pada tumpukan dan set perubahan, lihat `ActionMode` parameter.

Untuk membuat pesan kesalahan untuk CloudFormation tindakan di mana operasi tumpukan gagal, CodePipeline panggil CloudFormation `DescribeStackEvents` API. Jika peran IAM tindakan memiliki izin untuk mengakses API tersebut, detail tentang sumber daya pertama yang gagal akan disertakan dalam pesan CodePipeline kesalahan. Jika tidak, jika kebijakan peran tidak memiliki izin yang sesuai, CodePipeline akan mengabaikan mengakses API dan menampilkan pesan kesalahan umum sebagai gantinya. Untuk melakukan ini, `cloudformation:DescribeStackEvents` izin harus ditambahkan ke peran layanan atau peran IAM lainnya untuk pipeline.

Jika Anda tidak ingin detail sumber daya muncul dalam pesan kesalahan pipeline,
Anda dapat mencabut izin ini untuk peran IAM tindakan dengan menghapus izin.
`cloudformation:DescribeStackEvents`

Topik

- Tipe tindakan
- Parameter konfigurasi

- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Izin peran layanan: tindakan CloudFormation
- Deklarasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `CloudFormation`
- Versi: `1`

Parameter konfigurasi

ActionMode

Wajib: Ya

`ActionMode`adalah nama tindakan yang CloudFormation dilakukan pada tumpukan atau set perubahan. Mode aksi berikut tersedia:

- `CHANGE_SET_EXECUTE`mengeksekusi set perubahan untuk tumpukan sumber daya yang didasarkan pada serangkaian pembaruan sumber daya yang ditentukan. Dengan tindakan ini, CloudFormation mulai mengubah tumpukan.
- `CHANGE_SET_REPLACE` membuat set perubahan, jika tidak ada, berdasarkan nama tumpukan dan templat yang Anda kirimkan. Jika set perubahan ada, CloudFormation menghapusnya, dan kemudian membuat yang baru.
- `CREATE_UPDATE`membuat tumpukan jika tidak ada. Jika tumpukan ada, CloudFormation perbarui tumpukan. Gunakan tindakan ini untuk memperbarui tumpukan yang ada. Tidak sepertiREPLACE_ON_FAILURE, jika tumpukan ada dan dalam keadaan gagal, CodePipeline tidak akan menghapus dan mengganti tumpukan.
- `DELETE_ONLY` akan menghapus tumpukan. Jika Anda menentukan tumpukan yang tidak ada, tindakan berhasil diselesaikan tanpa menghapus tumpukan.

- `REPLACE_ON_FAILURE`membuat tumpukan, jika tidak ada. Jika tumpukan ada dan dalam keadaan gagal, CloudFormation hapus tumpukan, lalu buat tumpukan baru. Jika tumpukan tidak dalam keadaan gagal, CloudFormation perbarui.

  Tumpukan dalam keadaan gagal ketika salah satu jenis status berikut ditampilkan di CloudFormation:

  - `ROLLBACK_FAILED`
  - `CREATE_FAILED`
  - `DELETE_FAILED`
  - `UPDATE_ROLLBACK_FAILED`

  Gunakan tindakan ini untuk secara otomatis mengganti tumpukan yang gagal tanpa memulihkan atau memecahkan masalahnya.

  > **Important**
  >
  > Kami menyarankan Anda menggunakan `REPLACE_ON_FAILURE` untuk tujuan pengujian hanya karena mungkin menghapus tumpukan Anda.

StackName

Wajib: Ya

`StackName`adalah nama tumpukan yang ada atau tumpukan yang ingin Anda buat.

Kemampuan

Diperlukan: Kondisional

Penggunaan `Capabilities` mengakui bahwa template mungkin memiliki kemampuan untuk membuat dan memperbarui beberapa sumber daya sendiri, dan bahwa kemampuan ini ditentukan berdasarkan jenis sumber daya dalam template.

Properti ini diperlukan jika Anda memiliki sumber daya IAM di template tumpukan Anda atau Anda membuat tumpukan langsung dari template yang berisi makro. Agar CloudFormation tindakan berhasil beroperasi dengan cara ini, Anda harus secara eksplisit mengakui bahwa Anda ingin melakukannya dengan salah satu kemampuan berikut:

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`

- `CAPABILITY_AUTO_EXPAND`

Anda dapat menentukan lebih dari satu kemampuan dengan menggunakan koma (tanpa spasi) di antara kemampuan. Contoh di Deklarasi tindakan menunjukkan entri dengan properti CAPABILITY_IAM dan CAPABILITY_AUTO_EXPAND.

Untuk informasi selengkapnya`Capabilities`, lihat properti UpdateStackdi bawah Referensi AWS CloudFormation API.

ChangeSetName

Diperlukan: Kondisional

`ChangeSetName`adalah nama set perubahan yang ada atau set perubahan baru yang ingin Anda buat untuk tumpukan yang ditentukan.

Properti ini diperlukan untuk mode tindakan berikut: CHANGE_SET_REPLACE dan CHANGE_SET_EXECUTE. Untuk semua mode tindakan lainnya, properti ini diabaikan.

RoleArn

Diperlukan: Kondisional

`RoleArn`Ini adalah ARN dari peran layanan IAM yang CloudFormation mengasumsikan ketika beroperasi pada sumber daya di tumpukan yang ditentukan. `RoleArn`tidak diterapkan saat menjalankan set perubahan. Jika Anda tidak menggunakan CodePipeline untuk membuat set perubahan, pastikan bahwa set perubahan atau tumpukan memiliki peran terkait.

> **Note**
>
> Peran ini harus dalam akun yang sama dengan peran untuk tindakan yang sedang berjalan, seperti yang dikonfigurasi dalam deklarasi tindakan`RoleArn`.

Properti ini diperlukan untuk mode tindakan berikut:

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- DELETE_ONLY
- CHANGE_SET_REPLACE

> **Note**
>
> CloudFormation diberikan URL bertanda S3 ke template; oleh karena itu, ini `RoleArn` tidak memerlukan izin untuk mengakses bucket artefak. Namun, tindakan tersebut `RoleArn` memang memerlukan izin untuk mengakses bucket artefak, untuk menghasilkan URL yang ditandatangani.

TemplatePath

Diperlukan: Kondisional

`TemplatePath`mewakili file CloudFormation template. Anda menyertakan file dalam artefak input untuk tindakan ini. Nama file mengikuti format ini:

`Artifactname``::`*`TemplateFileName`*

`Artifactname` adalah nama artefak input seperti yang muncul di CodePipeline. Sebagai contoh, tahap sumber dengan nama artefak `SourceArtifact` dan nama file `template-export.json` membuat nama `TemplatePath` seperti yang ditunjukkan dalam contoh ini:

```
"TemplatePath": "SourceArtifact::template-export.json"
```

Properti ini diperlukan untuk mode tindakan berikut:

- CREATE_UPDATE
- REPLACE_ON_FAILURE
- CHANGE_SET_REPLACE

Untuk semua mode tindakan lainnya, properti ini diabaikan.

> **Note**
>
> File CloudFormation template yang berisi badan template memiliki panjang minimal 1 byte dan panjang maksimum 1 MB. Untuk tindakan CloudFormation penerapan di CodePipeline, ukuran artefak input maksimum selalu 256 MB. Untuk informasi selengkapnya, lihat Kuota di AWS CodePipeline dan CloudFormation Batas.

OutputFileName

Wajib: Tidak

Gunakan `OutputFileName` untuk menentukan nama file keluaran, seperti`CreateStackOutput.json`, yang CodePipeline menambah artefak keluaran pipeline untuk tindakan ini. File JSON berisi isi `Outputs` bagian dari CloudFormation tumpukan.

Jika Anda tidak menentukan nama, CodePipeline tidak menghasilkan file keluaran atau artefak.

ParameterOverrides

Wajib: Tidak

Parameter didefinisikan dalam template tumpukan Anda dan memungkinkan Anda untuk memberikan nilai untuk mereka pada saat pembuatan tumpukan atau pembaruan. Anda dapat menggunakan objek JSON untuk mengatur nilai parameter dalam template Anda. (Nilai-nilai ini mengesampingkan yang ditetapkan dalam file konfigurasi template.) Untuk informasi selengkapnya tentang penggunaan penggantian parameter, lihat Properti Konfigurasi (Objek JSON).

Kami menyarankan Anda menggunakan file konfigurasi template untuk sebagian besar nilai parameter Anda. Gunakan penggantian parameter hanya untuk nilai yang tidak diketahui sampai pipeline berjalan. Untuk informasi selengkapnya, lihat Menggunakan Fungsi Ganti Parameter dengan CodePipeline Pipelines di AWS CloudFormation Panduan Pengguna.

> **Note**
>
> Semua nama parameter harus ada dalam templat tumpukan.

TemplateConfiguration

Wajib: Tidak

`TemplateConfiguration` adalah file konfigurasi templat. Anda menyertakan file dalam artefak input untuk tindakan ini. Ini dapat berisi nilai parameter template dan kebijakan tumpukan. Untuk informasi selengkapnya tentang format file konfigurasi templat, lihat AWS CloudFormation Artefak.

Nama file konfigurasi template mengikuti format ini:

`Artifactname``::`*`TemplateConfigurationFileName`*

`Artifactname` adalah nama artefak input seperti yang muncul di CodePipeline. Sebagai contoh, tahap sumber dengan nama artefak `SourceArtifact` dan nama file `test-configuration.json` membuat nama `TemplateConfiguration` seperti yang ditunjukkan dalam contoh ini:

```
"TemplateConfiguration": "SourceArtifact::test-configuration.json"
```

Artefak masukan

- Jumlah artefak: `0 to 10`
- Deskripsi: Sebagai masukan, CloudFormation tindakan secara opsional menerima artefak untuk tujuan ini:
  - Untuk menyediakan file template stack untuk dieksekusi. (Lihat `TemplatePath` parameternya.)
  - Untuk menyediakan file konfigurasi template yang akan digunakan. (Lihat `TemplateConfiguration` parameternya.) Untuk informasi selengkapnya tentang format file konfigurasi templat, lihat AWS CloudFormation Artefak.
  - Untuk menyediakan artefak untuk fungsi Lambda yang akan digunakan sebagai bagian dari tumpukan. CloudFormation

Artefak keluaran

- Jumlah artefak: `0 to 1`
- Deskripsi: Jika `OutputFileName` parameter ditentukan, ada artefak keluaran yang dihasilkan oleh tindakan ini yang berisi file JSON dengan nama yang ditentukan. File JSON berisi isi bagian Output dari tumpukan. CloudFormation

  Untuk informasi selengkapnya tentang bagian output yang dapat Anda buat untuk CloudFormation tindakan Anda, lihat Keluaran.

Variabel keluaran

Ketika dikonfigurasi, tindakan ini menghasilkan variabel yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir dalam pipeline. Anda mengonfigurasi tindakan dengan namespace untuk membuat variabel-variabel tersebut tersedia untuk konfigurasi tindakan hilir.

Untuk CloudFormation tindakan, variabel dihasilkan dari nilai apa pun yang ditunjuk di `Outputs` bagian template tumpukan. Perhatikan bahwa satu-satunya mode CloudFormation tindakan yang menghasilkan output adalah mode yang menghasilkan atau memperbarui tumpukan, seperti pembuatan tumpukan, pembaruan tumpukan, dan eksekusi set perubahan. Mode tindakan terkait yang menghasilkan variabel adalah:

- `CHANGE_SET_EXECUTE`
- `CHANGE_SET_REPLACE`
- `CREATE_UPDATE`
- `REPLACE_ON_FAILURE`

Untuk informasi selengkapnya, lihat Referensi variabel. Untuk tutorial yang menunjukkan cara membuat pipeline dengan tindakan CloudFormation penerapan dalam pipeline yang menggunakan variabel CloudFormation keluaran, lihatTutorial: Buat pipeline yang menggunakan variabel dari AWS CloudFormation tindakan penerapan.

Izin peran layanan: tindakan CloudFormation

Saat CodePipeline menjalankan tindakan, kebijakan peran CodePipeline layanan memerlukan izin berikut, yang dicakup dengan tepat ke ARN sumber daya pipeline untuk mempertahankan akses dengan hak istimewa paling sedikit. Misalnya, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

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"
                    ]
                }
            }
        }
    ]
}
```

Perhatikan bahwa `cloudformation:DescribeStackEvents` izin adalah opsional. Hal ini memungkinkan CloudFormation tindakan untuk menampilkan pesan kesalahan yang lebih rinci. Izin ini dapat dicabut dari peran IAM jika Anda tidak ingin detail sumber daya muncul dalam pesan kesalahan pipeline.

Deklarasi tindakan

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"
        }
    ]
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Referensi Properti Konfigurasi - Bab referensi ini dalam Panduan AWS CloudFormation Pengguna memberikan lebih banyak deskripsi dan contoh untuk CodePipeline parameter ini.
- AWS CloudFormation Referensi API — CreateStackParameter dalam Referensi AWS CloudFormation API menjelaskan parameter tumpukan untuk CloudFormation template.

CloudFormation StackSets menyebarkan referensi tindakan

CodePipeline menawarkan kemampuan untuk melakukan CloudFormation StackSets operasi sebagai bagian dari proses CI/CD Anda. Anda menggunakan kumpulan tumpukan untuk membuat tumpukan di AWS akun di seluruh AWS Wilayah dengan menggunakan satu CloudFormation templat. Semua sumber daya yang disertakan dalam setiap tumpukan ditentukan oleh CloudFormation template set tumpukan. Saat Anda membuat kumpulan tumpukan, Anda menentukan template yang akan digunakan, serta parameter dan kemampuan apa pun yang dibutuhkan template.

Untuk informasi selengkapnya tentang konsep CloudFormation StackSets, lihat StackSets konsep di Panduan AWS CloudFormation Pengguna.

Anda mengintegrasikan pipeline Anda dengan CloudFormation StackSets melalui dua tipe tindakan berbeda yang Anda gunakan bersama-sama:

- `CloudFormationStackSet`Tindakan membuat atau memperbarui stack set atau stack instance dari template yang disimpan di lokasi sumber pipeline. Setiap kali kumpulan tumpukan dibuat atau diperbarui, ia memulai penerapan perubahan tersebut ke instance tertentu. Di konsol, Anda dapat memilih penyedia tindakan CloudFormation Stack Set saat membuat atau mengedit pipeline.
- `CloudFormationStackInstances`Tindakan ini menerapkan perubahan dari `CloudFormationStackSet` tindakan ke instance tertentu, membuat instance tumpukan baru, dan mendefinisikan penggantian parameter ke instance tertentu. Di konsol, Anda dapat memilih penyedia tindakan CloudFormation Stack Instances saat mengedit pipeline yang ada.

Anda dapat menggunakan tindakan ini untuk menyebarkan ke AWS akun target atau menargetkan unit AWS IDs organisasi Organizations.

> **Note**
>
> Untuk menyebarkan ke akun AWS Organizations target atau unit organisasi IDs dan menggunakan model izin yang dikelola layanan, Anda harus mengaktifkan akses tepercaya antara dan Organizations. AWS CloudFormation StackSets AWS Untuk informasi selengkapnya, lihat Mengaktifkan akses tepercaya dengan AWS CloudFormation Stacksets.

Topik

- Bagaimana CloudFormation StackSets tindakan bekerja
- Cara menyusun StackSets tindakan dalam pipa
- Tindakan CloudFormationStackSet
- CloudFormationStackInstances Tindakan
- Izin peran layanan: tindakan CloudFormationStackSet
- Izin peran layanan: tindakan CloudFormationStackInstances
- Model izin untuk operasi set tumpukan
- Jenis data parameter template
- Lihat juga

Bagaimana CloudFormation StackSets tindakan bekerja

`CloudFormationStackSet`Tindakan membuat atau memperbarui sumber daya tergantung pada apakah tindakan tersebut berjalan untuk pertama kalinya.

`CloudFormationStackSet`Tindakan membuat atau memperbarui kumpulan tumpukan dan menyebarkan perubahan tersebut ke instance tertentu.

> **Note**
>
> Jika Anda menggunakan tindakan ini untuk membuat pembaruan yang menyertakan penambahan instance tumpukan, instance baru akan diterapkan terlebih dahulu dan pembaruan selesai terakhir. Instans baru pertama menerima versi lama, dan kemudian pembaruan diterapkan ke semua instance.

- Buat: Ketika tidak ada instance yang ditentukan dan kumpulan tumpukan tidak ada, CloudFormationStackSettindakan membuat kumpulan tumpukan tanpa membuat instance apa pun.
- Pembaruan: Saat CloudFormationStackSettindakan dijalankan untuk kumpulan tumpukan yang sudah dibuat, tindakan akan memperbarui kumpulan tumpukan. Jika tidak ada instance yang ditentukan dan kumpulan tumpukan sudah ada, semua instance diperbarui. Jika tindakan ini digunakan untuk memperbarui instance tertentu, semua instance yang tersisa akan berpindah ke status USANG.

  Anda dapat menggunakan CloudFormationStackSettindakan untuk memperbarui kumpulan tumpukan dengan cara berikut.

  - Perbarui template pada beberapa atau semua contoh.
  - Perbarui parameter pada beberapa atau semua contoh.
  - Perbarui peran eksekusi untuk kumpulan tumpukan (ini harus cocok dengan peran eksekusi yang ditentukan dalam peran Administrator).
  - Ubah model izin (hanya jika tidak ada instance yang dibuat).
  - Aktifkan/Nonaktifkan `AutoDeployment` jika model izin set tumpukan adalah. `Service Managed`
  - Bertindak sebagai administrator yang didelegasikan di akun anggota jika model izin set tumpukan adalah. `Service Managed`
  - Perbarui peran Administrator.
  - Perbarui deskripsi pada set tumpukan.
  - Tambahkan target penerapan ke pembaruan set tumpukan untuk membuat instance tumpukan baru.

`CloudFormationStackInstances`Tindakan ini membuat instance tumpukan baru atau memperbarui instance tumpukan yang sudah ketinggalan zaman. Sebuah instance menjadi usang ketika kumpulan tumpukan diperbarui, tetapi tidak semua instance di dalamnya diperbarui.

- Buat: Jika tumpukan sudah ada, `CloudFormationStackInstances` tindakan hanya memperbarui instance dan tidak membuat instance tumpukan.
- Pembaruan: Setelah `CloudFormationStackSet` tindakan dilakukan, jika templat atau parameter telah diperbarui hanya dalam beberapa kasus, sisanya akan ditandai`OUTDATED`. Pada tahap pipeline selanjutnya, `CloudFormationStackInstances` perbarui sisa instance di tumpukan yang disetel dalam gelombang sehingga semua instance ditandai. `CURRENT` Tindakan ini juga dapat digunakan untuk menambahkan instance tambahan atau mengganti parameter pada instance baru atau yang sudah ada.

Sebagai bagian dari pembaruan, `CloudFormationStackInstances` tindakan `CloudFormationStackSet` dan dapat menentukan target penerapan baru, yang membuat instance tumpukan baru.

Sebagai bagian dari pembaruan, `CloudFormationStackInstances` tindakan `CloudFormationStackSet` dan tidak menghapus kumpulan tumpukan, instance, atau sumber daya. Saat tindakan memperbarui tumpukan tetapi tidak menentukan semua instance yang akan diperbarui, instance yang tidak ditentukan untuk pembaruan akan dihapus dari pembaruan dan disetel ke status. `OUTDATED`

Selama penerapan, instance tumpukan juga dapat menampilkan status `OUTDATED` jika penerapan ke instance gagal.

Cara menyusun StackSets tindakan dalam pipa

Sebagai praktik terbaik, Anda harus membangun pipeline Anda sehingga kumpulan tumpukan dibuat dan awalnya diterapkan ke subset atau satu instance. Setelah Anda menguji penerapan dan melihat kumpulan tumpukan yang dihasilkan, lalu tambahkan `CloudFormationStackInstances` tindakan sehingga instance yang tersisa dibuat dan diperbarui.

Gunakan konsol atau CLI untuk membuat struktur pipa yang direkomendasikan sebagai berikut:

1. Buat pipeline dengan aksi sumber (wajib) dan `CloudFormationStackSet` tindakan sebagai tindakan penerapan. Jalankan pipeline Anda.

2. Saat pipeline Anda pertama kali berjalan, `CloudFormationStackSet` tindakan akan membuat kumpulan tumpukan Anda dan setidaknya satu instance awal. Verifikasi pembuatan set tumpukan dan tinjau penerapan ke instance awal Anda. Misalnya, untuk pembuatan set tumpukan awal untuk akun Account-A `us-east-1` dimana Region yang ditentukan, instance stack dibuat dengan set stack:

Contoh tumpukan	Region	Status
StackInstanceID-1	us-east-1	SAAT INI

3. Edit pipeline Anda untuk ditambahkan `CloudFormationStackInstances` sebagai tindakan penerapan kedua untuk create/update menumpuk instance untuk target yang Anda tetapkan. Misalnya, untuk pembuatan instance tumpukan untuk akun `Account-A` tempat `eu-central-1` Wilayah `us-east-2` dan ditentukan, instance tumpukan yang tersisa dibuat dan instance awal tetap diperbarui sebagai berikut:

Contoh tumpukan	Region	Status
StackInstanceID-1	us-east-1	SAAT INI
StackInstanceID-2	us-east-2	SAAT INI
StackInstanceID-3	eu-central-1	SAAT INI

4. Jalankan pipeline sesuai kebutuhan untuk memperbarui kumpulan tumpukan dan memperbarui atau membuat instance tumpukan.

Saat Anda memulai pembaruan tumpukan di mana Anda telah menghapus target penerapan dari konfigurasi tindakan, maka instance tumpukan yang tidak ditunjuk untuk pembaruan akan dihapus dari penerapan dan dipindahkan ke status USANG. Misalnya, untuk pembaruan instans tumpukan untuk akun `Account-A` di mana `us-east-2` Wilayah dihapus dari konfigurasi tindakan, instance tumpukan yang tersisa dibuat dan instance yang dihapus disetel ke USANG sebagai berikut:

Contoh tumpukan	Region	Status
StackInstanceID-1	us-east-1	SAAT INI
StackInstanceID-2	us-east-2	KETINGGALAN JAMAN

Contoh tumpukan	Region	Status
StackInstanceID-3	eu-central-1	SAAT INI

Untuk informasi selengkapnya tentang praktik terbaik untuk menerapkan kumpulan tumpukan, lihat Praktik terbaik untuk StackSets di Panduan AWS CloudFormation Pengguna.

Tindakan `CloudFormationStackSet`

Tindakan ini membuat atau memperbarui kumpulan tumpukan dari template yang disimpan di lokasi sumber pipeline.

Setelah menentukan kumpulan tumpukan, Anda dapat membuat, memperbarui, atau menghapus tumpukan di akun target dan Wilayah yang ditentukan dalam parameter konfigurasi. Saat membuat, memperbarui, dan menghapus tumpukan, Anda dapat menentukan preferensi lain, seperti urutan Wilayah untuk operasi yang akan dilakukan, persentase toleransi kegagalan di luar operasi tumpukan berhenti, dan jumlah akun tempat operasi dilakukan pada tumpukan secara bersamaan.

Satu set tumpukan adalah sumber daya regional. Jika Anda membuat kumpulan tumpukan di satu AWS Wilayah, Anda tidak dapat mengaksesnya dari Wilayah lain.

Saat tindakan ini digunakan sebagai tindakan pembaruan ke kumpulan tumpukan, pembaruan ke tumpukan tidak diizinkan tanpa penerapan ke setidaknya satu instance tumpukan.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Contoh konfigurasi CloudFormationStackSettindakan

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`

- Penyedia: `CloudFormationStackSet`
- Versi: 1

Parameter konfigurasi

StackSetName

Wajib: Ya

Nama yang akan dikaitkan dengan set tumpukan. Nama ini harus unik di Wilayah tempat ia dibuat.

Nama mungkin hanya berisi karakter alfanumerik dan tanda hubung. Itu harus dimulai dengan karakter alfabet dan 128 karakter atau kurang.

Deskripsi

Wajib: Tidak

Deskripsi set tumpukan. Anda dapat menggunakan ini untuk menggambarkan tujuan kumpulan tumpukan atau informasi relevan lainnya.

TemplatePath

Wajib: Ya

Lokasi template yang mendefinisikan sumber daya dalam kumpulan tumpukan. Ini harus menunjuk ke template dengan ukuran maksimum 460.800 byte.

Masukkan path ke nama artefak sumber dan file template dalam format`"InputArtifactName::TemplateFileName"`, seperti yang ditunjukkan pada contoh berikut.

```
SourceArtifact::template.txt
```

Parameter

Wajib: Tidak

Daftar parameter template untuk kumpulan tumpukan Anda yang diperbarui selama penerapan.

Anda dapat memberikan parameter sebagai daftar literal atau jalur file:

- Anda dapat memasukkan parameter dalam format sintaks singkatan berikut:. `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  Untuk informasi selengkapnya tentang tipe data ini, lihatJenis data parameter template.

  Contoh berikut menunjukkan parameter bernama `BucketName` dengan nilai`amzn-s3-demo-source-bucket`.

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ```

  Contoh berikut menunjukkan entri dengan beberapa parameter:

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ParameterKey=Asset1,ParameterValue=true
  ParameterKey=Asset2,ParameterValue=true
  ```

- Anda dapat memasukkan lokasi file yang berisi daftar penggantian parameter template yang dimasukkan dalam format`"InputArtifactName::ParametersFileName"`, seperti yang ditunjukkan pada contoh berikut.

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

  Contoh berikut menunjukkan isi file untuk`parameters.txt`.

  ```
  [
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      },
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      }
  ]
  ```

Kemampuan

Wajib: Tidak

Menunjukkan bahwa template dapat membuat dan memperbarui sumber daya, tergantung pada jenis sumber daya dalam template.

Anda harus menggunakan properti ini jika Anda memiliki sumber daya IAM di template tumpukan Anda atau Anda membuat tumpukan langsung dari template yang berisi makro. Agar CloudFormation tindakan berhasil beroperasi dengan cara ini, Anda harus menggunakan salah satu dari kemampuan berikut:

- `CAPABILITY_IAM`
- `CAPABILITY_NAMED_IAM`

Anda dapat menentukan lebih dari satu kemampuan dengan menggunakan koma dan tidak ada spasi di antara kemampuan. Contoh di Contoh konfigurasi CloudFormationStackSettindakan menunjukkan entri dengan beberapa kemampuan.

PermissionModel

Wajib: Tidak

Menentukan bagaimana peran IAM dibuat dan dikelola. Jika bidang tidak ditentukan, default digunakan. Untuk informasi, lihat Model izin untuk operasi set tumpukan.

Nilai yang valid adalah:

- `SELF_MANAGED`(default): Anda harus membuat peran administrator dan eksekusi untuk menyebarkan ke akun target.
- `SERVICE_MANAGED`: CloudFormation StackSets secara otomatis membuat peran IAM yang diperlukan untuk menyebarkan ke akun yang dikelola oleh Organizations AWS . Ini membutuhkan akun untuk menjadi anggota Organisasi.

> **Note**
>
> Parameter ini hanya dapat diubah ketika tidak ada instance tumpukan di set tumpukan.

AdministrationRoleArn

> **Note**
>
> Karena CloudFormation StackSets melakukan operasi di beberapa akun, Anda harus menentukan izin yang diperlukan di akun tersebut sebelum Anda dapat membuat kumpulan tumpukan.

Wajib: Tidak

> **Note**
>
> Parameter ini opsional untuk model izin SELF_MANAGED dan tidak digunakan untuk model izin SERVICE_MANAGED.

ARN dari peran IAM di akun administrator yang digunakan untuk melakukan operasi set tumpukan.

Nama mungkin berisi karakter alfanumerik, salah satu karakter berikut: _+=, .@-, dan tidak ada spasi. Namanya tidak peka huruf besar/kecil. Nama peran ini harus memiliki panjang minimal 20 karakter dan panjang maksimum 2048 karakter. Nama peran harus unik di dalam akun. Nama peran yang ditentukan di sini harus berupa nama peran yang ada. Jika Anda tidak menentukan nama peran, itu diatur ke AWSCloudFormationStackSetAdministrationRole. Jika Anda menentukan ServiceManaged, Anda tidak boleh menentukan nama peran.

ExecutionRoleName

> **Note**
>
> Karena CloudFormation StackSets melakukan operasi di beberapa akun, Anda harus menentukan izin yang diperlukan di akun tersebut sebelum Anda dapat membuat kumpulan tumpukan.

Wajib: Tidak

> **Note**
>
> Parameter ini opsional untuk model izin SELF_MANAGED dan tidak digunakan untuk model izin SERVICE_MANAGED.

Nama peran IAM dalam akun target yang digunakan untuk melakukan operasi set tumpukan. Nama mungkin berisi karakter alfanumerik, salah satu karakter berikut: _+=, .@-, dan tidak ada spasi. Namanya tidak peka huruf besar/kecil. Nama peran ini harus memiliki panjang minimal 1 karakter dan panjang maksimum 64 karakter. Nama peran harus unik di dalam akun. Nama peran yang ditentukan di sini harus berupa nama peran yang ada. Jangan tentukan peran ini jika Anda menggunakan peran eksekusi yang disesuaikan. Jika Anda tidak menentukan nama peran, itu diatur ke`AWSCloudFormationStackSetExecutionRole`. Jika Anda menyetel Service_Managed ke true, Anda tidak boleh menentukan nama peran.

OrganizationsAutoDeployment

Wajib: Tidak

> **Note**
>
> Parameter ini opsional untuk model izin SERVICE_MANAGED dan tidak digunakan untuk model izin SELF_MANAGED.

Menjelaskan apakah CloudFormation StackSets secara otomatis menyebarkan ke akun AWS Organizations yang ditambahkan ke organisasi target atau unit organisasi (OU). Jika `OrganizationsAutoDeployment` ditentukan, jangan tentukan `DeploymentTargets` dan`Regions`.

> **Note**
>
> Jika tidak ada input yang disediakan`OrganizationsAutoDeployment`, maka nilai defaultnya adalah`Disabled`.

Nilai yang valid adalah:

- `Enabled`. Diperlukan: Tidak.

StackSets secara otomatis menyebarkan instance tumpukan tambahan ke akun AWS Organizations yang ditambahkan ke organisasi target atau unit organisasi (OU) di Wilayah yang ditentukan. Jika akun dihapus dari organisasi target atau OU, CloudFormation StackSets menghapus instance tumpukan dari akun di Wilayah yang ditentukan.

- `Disabled`. Diperlukan: Tidak.

  StackSets tidak secara otomatis menyebarkan instance tumpukan tambahan ke akun AWS Organizations yang ditambahkan ke organisasi target atau unit organisasi (OU) di Wilayah yang ditentukan.

- `EnabledWithStackRetention`. Diperlukan: Tidak.

  Sumber daya tumpukan dipertahankan ketika akun dihapus dari organisasi target atau OU.

DeploymentTargets

Wajib: Tidak

> **Note**
>
> Untuk model izin SERVICE_MANAGED, Anda dapat memberikan ID root organisasi atau Unit IDs organisasi untuk target penerapan. Untuk model izin SELF_MANAGED, Anda hanya dapat menyediakan akun.

> **Note**
>
> Ketika parameter ini dipilih, Anda juga harus memilih Wilayah.

Daftar AWS akun atau unit organisasi IDs tempat instance stack set harus dibuat/diperbarui.

- Akun:

  Anda dapat memberikan akun sebagai daftar literal atau jalur file:

  - Literal: Masukkan parameter dalam format sintaks singkatan`account_ID,account_ID`, seperti yang ditunjukkan pada contoh berikut.

    ```
    111111222222,333333444444
    ```

- Jalur file: Lokasi file yang berisi daftar AWS akun tempat instance kumpulan tumpukan harus dibuat/diperbarui, dimasukkan dalam format. `InputArtifactName::AccountsFileName` Jika Anda menggunakan jalur file untuk menentukan salah satu akun atau OrganizationalUnitIds, format file harus dalam JSON, seperti yang ditunjukkan pada contoh berikut.

  ```
  SourceArtifact::accounts.txt
  ```

  Contoh berikut menunjukkan isi file untuk`accounts.txt`.

  ```
  [
      "111111222222"
  ]
  ```

  Contoh berikut menunjukkan isi file `accounts.txt` saat mencantumkan lebih dari satu akun:

  ```
  [
      "111111222222","333333444444"
  ]
  ```

- OrganizationalUnitIds:

  > **Note**
  >
  > Parameter ini opsional untuk model izin SERVICE_MANAGED dan tidak digunakan untuk model izin SELF_MANAGED. Jangan gunakan ini jika Anda memilih OrganizationsAutoDeployment.

  Unit AWS organisasi untuk memperbarui instance tumpukan terkait.

  Anda dapat menyediakan unit organisasi IDs sebagai daftar literal atau jalur file:

  - Literal: Masukkan array string dipisahkan dengan koma, seperti yang ditunjukkan pada contoh berikut.

    ```
    ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
    ```

- Jalur file: Lokasi file yang berisi daftar tempat untuk membuat atau memperbarui instance kumpulan tumpukan. OrganizationalUnitIds Jika Anda menggunakan jalur file untuk menentukan salah satu akun atau OrganizationalUnitIds, format file harus dalam JSON, seperti yang ditunjukkan pada contoh berikut.

  Masukkan jalur ke file dalam formatInputArtifactName::OrganizationalUnitIdsFileName.

  ```
  SourceArtifact::OU-IDs.txt
  ```

  Contoh berikut menunjukkan isi file untukOU-IDs.txt:

  ```
  [
      "ou-examplerootid111-exampleouid111","ou-examplerootid222-exampleouid222"
  ]
  ```

Daerah

Wajib: Tidak

> **Note**
>
> Ketika parameter ini dipilih, Anda juga harus memilih DeploymentTargets.

Daftar AWS Wilayah tempat instance kumpulan tumpukan dibuat atau diperbarui. Wilayah diperbarui dalam urutan di mana mereka dimasukkan.

Masukkan daftar AWS Wilayah yang valid dalam formatRegion1,Region2, seperti yang ditunjukkan pada contoh berikut.

```
us-west-2,us-east-1
```

FailureTolerancePercentage

Wajib: Tidak

Persentase akun per Wilayah yang operasi tumpukan ini dapat gagal sebelum CloudFormation menghentikan operasi di Wilayah tersebut. Jika operasi dihentikan di Wilayah, CloudFormation

tidak mencoba operasi di Wilayah berikutnya. Saat menghitung jumlah akun berdasarkan persentase yang ditentukan, CloudFormation bulatkan ke nomor bulat berikutnya.

MaxConcurrentPercentage

Wajib: Tidak

Persentase maksimum akun untuk melakukan operasi ini pada satu waktu. Saat menghitung jumlah akun berdasarkan persentase yang ditentukan, CloudFormation bulatkan ke nomor bulat berikutnya. Jika pembulatan ke bawah akan menghasilkan nol, CloudFormation tetapkan angka sebagai satu sebagai gantinya. Meskipun Anda menggunakan pengaturan ini untuk menentukan maksimum, untuk penerapan besar jumlah aktual akun yang ditindaklanjuti secara bersamaan mungkin lebih rendah karena pelambatan layanan.

RegionConcurrencyType

Wajib: Tidak

Anda dapat menentukan apakah kumpulan tumpukan harus diterapkan Wilayah AWS secara berurutan atau paralel dengan mengonfigurasi parameter penerapan konkurensi Wilayah. Ketika konkurensi Wilayah ditentukan untuk menyebarkan tumpukan di beberapa Wilayah AWS secara paralel, ini dapat menghasilkan waktu penerapan keseluruhan yang lebih cepat.

- Paralel: Penerapan set tumpukan akan dilakukan pada saat yang sama, selama kegagalan penerapan Wilayah tidak melebihi toleransi kegagalan yang ditentukan.
- Berurutan: Penerapan set tumpukan akan dilakukan satu per satu, selama kegagalan penerapan Wilayah tidak melebihi toleransi kegagalan yang ditentukan. Deployment berurutan adalah pilihan default.

ConcurrencyMode

Wajib: Tidak

Mode konkurensi memungkinkan Anda memilih bagaimana tingkat konkurensi berperilaku selama operasi set tumpukan, baik dengan toleransi kegagalan yang ketat atau lunak. Toleransi Kegagalan Ketat menurunkan kecepatan penerapan karena kegagalan operasi set tumpukan terjadi karena konkurensi menurun untuk setiap kegagalan. Soft Failure Tolerance memprioritaskan kecepatan penerapan sambil tetap memanfaatkan CloudFormation kemampuan keselamatan.

- `STRICT_FAILURE_TOLERANCE`: Opsi ini secara dinamis menurunkan tingkat konkurensi untuk memastikan jumlah akun yang gagal tidak pernah melebihi toleransi kegagalan tertentu. Ini adalah perilaku default.

- `SOFT_FAILURE_TOLERANCE`: Opsi ini memisahkan toleransi kegagalan dari konkurensi yang sebenarnya. Hal ini memungkinkan operasi stack set berjalan pada tingkat konkurensi yang ditetapkan, terlepas dari jumlah kegagalan.

CallAs

Wajib: Tidak

> **Note**
>
> Parameter ini opsional untuk model `SERVICE_MANAGED` izin dan tidak digunakan untuk model `SELF_MANAGED` izin.

Menentukan apakah Anda bertindak di akun manajemen organisasi atau sebagai administrator yang didelegasikan dalam akun anggota.

> **Note**
>
> Jika parameter ini disetel ke`DELEGATED_ADMIN`, pastikan bahwa peran IAM pipeline memiliki `organizations:ListDelegatedAdministrators` izin. Jika tidak, tindakan akan gagal saat berjalan dengan kesalahan yang mirip dengan berikut ini:`Account used is not a delegated administrator`.

- `SELF`: Penyebaran set tumpukan akan menggunakan izin yang dikelola layanan saat masuk ke akun manajemen.
- `DELEGATED_ADMIN`: Penyebaran set tumpukan akan menggunakan izin yang dikelola layanan saat masuk ke akun administrator yang didelegasikan.

Artefak masukan

Anda harus menyertakan setidaknya satu artefak masukan yang berisi template untuk kumpulan tumpukan dalam `CloudFormationStackSet` tindakan. Anda dapat menyertakan lebih banyak artefak masukan untuk daftar target penyebaran, akun, dan parameter.

- Jumlah artefak: `1 to 3`
- Deskripsi: Anda dapat memasukkan artefak untuk menyediakan:

- File template tumpukan. (Lihat `TemplatePath` parameternya.)
- File parameter. (Lihat `Parameters` parameternya.)
- File akun. (Lihat `DeploymentTargets` parameternya.)

Artefak keluaran

- Jumlah artefak: `0`
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Variabel keluaran

Jika Anda mengonfigurasi tindakan ini, ini menghasilkan variabel yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir di pipeline. Anda mengonfigurasi tindakan dengan namespace untuk membuat variabel-variabel tersebut tersedia untuk konfigurasi tindakan hilir.

- StackSetId: ID dari set tumpukan.
- OperationId: ID operasi stack set.

Untuk informasi selengkapnya, lihat Referensi variabel.

Contoh konfigurasi CloudFormationStackSettindakan

Contoh berikut menunjukkan konfigurasi tindakan untuk CloudFormationStackSettindakan tersebut.

Contoh untuk model izin yang dikelola sendiri

Contoh berikut menunjukkan CloudFormationStackSettindakan di mana target penyebaran yang dimasukkan adalah ID AWS akun.

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"
}
```

Contoh untuk model izin yang dikelola layanan

Contoh berikut menunjukkan CloudFormationStackSettindakan untuk model izin yang dikelola layanan di mana opsi untuk penerapan otomatis ke AWS Organizations diaktifkan dengan penyimpanan tumpukan.

YAML

```
Name: Deploy
ActionTypeId:
  Category: Deploy
  Owner: AWS
  Provider: CloudFormationStackSet
  Version: '1'
RunOrder: 1
Configuration:
  Capabilities: 'CAPABILITY_IAM,CAPABILITY_NAMED_IAM'
  OrganizationsAutoDeployment: EnabledWithStackRetention
  PermissionModel: SERVICE_MANAGED
  StackSetName: stacks-orgs
  TemplatePath: 'SourceArtifact::template.json'
OutputArtifacts: []
InputArtifacts:
  - Name: SourceArtifact
Region: eu-central-1
Namespace: DeployVariables
```

JSON

```
{
    "Name": "Deploy",
    "ActionTypeId": {
        "Category": "Deploy",
        "Owner": "AWS",
        "Provider": "CloudFormationStackSet",
        "Version": "1"
    },
    "RunOrder": 1,
    "Configuration": {
        "Capabilities": "CAPABILITY_IAM,CAPABILITY_NAMED_IAM",
        "OrganizationsAutoDeployment": "EnabledWithStackRetention",
        "PermissionModel": "SERVICE_MANAGED",
        "StackSetName": "stacks-orgs",
```

```
        "TemplatePath": "SourceArtifact::template.json"
    },
    "OutputArtifacts": [],
    "InputArtifacts": [
        {
            "Name": "SourceArtifact"
        }
    ],
    "Region": "eu-central-1",
    "Namespace": "DeployVariables"
}
```

CloudFormationStackInstances Tindakan

Tindakan ini membuat instance baru dan menyebarkan kumpulan tumpukan ke instance tertentu. Instans tumpukan adalah referensi ke tumpukan di akun target dalam Wilayah. Sebuah instance stack bisa ada tanpa stack; misalnya, jika pembuatan stack tidak berhasil, instance stack menunjukkan alasan kegagalan pembuatan stack. Sebuah instans templat dikaitkan dengan hanya satu set tumpukan.

Setelah pembuatan awal kumpulan tumpukan, Anda dapat menambahkan instance tumpukan baru dengan menggunakan`CloudFormationStackInstances`. Nilai parameter template dapat diganti pada tingkat instance tumpukan selama membuat atau memperbarui operasi instance set tumpukan.

Setiap set tumpukan memiliki satu template dan set parameter template. Saat Anda memperbarui parameter template atau template, Anda memperbaruinya untuk seluruh rangkaian. Kemudian semua status instance disetel ke `OUTDATED` sampai perubahan diterapkan ke instance itu.

Untuk mengganti nilai parameter pada instance tertentu, misalnya, jika templat berisi parameter untuk `stage` dengan nilai`prod`, Anda dapat mengganti nilai parameter tersebut menjadi atau. `beta` `gamma`

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Contoh konfigurasi tindakan

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `CloudFormationStackInstances`
- Versi: `1`

Parameter konfigurasi

StackSetName

Wajib: Ya

Nama yang akan dikaitkan dengan set tumpukan. Nama ini harus unik di Wilayah tempat ia dibuat.

Nama mungkin hanya berisi karakter alfanumerik dan tanda hubung. Itu harus dimulai dengan karakter alfabet dan 128 karakter atau kurang.

DeploymentTargets

Wajib: Tidak

> **Note**
>
> Untuk model izin SERVICE_MANAGED, Anda dapat memberikan ID root organisasi atau Unit IDs organisasi untuk target penerapan. Untuk model izin SELF_MANAGED, Anda hanya dapat menyediakan akun.

> **Note**
>
> Ketika parameter ini dipilih, Anda juga harus memilih Wilayah.

Daftar AWS akun atau unit organisasi IDs tempat instance stack set harus dibuat/diperbarui.

- Akun:

  Anda dapat memberikan akun sebagai daftar literal atau jalur file:

- Literal: Masukkan parameter dalam format sintaks singkatan`account_ID,account_ID`, seperti yang ditunjukkan pada contoh berikut.

```
111111222222,333333444444
```

- Jalur file: Lokasi file yang berisi daftar AWS akun tempat instance kumpulan tumpukan harus dibuat/diperbarui, dimasukkan dalam format. `InputArtifactName::AccountsFileName` Jika Anda menggunakan jalur file untuk menentukan salah satu akun atau OrganizationalUnitIds, format file harus dalam JSON, seperti yang ditunjukkan pada contoh berikut.

```
SourceArtifact::accounts.txt
```

Contoh berikut menunjukkan isi file untuk`accounts.txt`:

```
[
    "111111222222"
]
```

Contoh berikut menunjukkan isi file `accounts.txt` saat mencantumkan lebih dari satu akun:

```
[
    "111111222222","333333444444"
]
```

- OrganizationalUnitIds:

> **Note**
>
> Parameter ini opsional untuk model izin SERVICE_MANAGED dan tidak digunakan untuk model izin SELF_MANAGED. Jangan gunakan ini jika Anda memilih OrganizationsAutoDeployment.

Unit AWS organisasi untuk memperbarui instance tumpukan terkait.

Anda dapat menyediakan unit organisasi IDs sebagai daftar literal atau jalur file.

- Literal: Masukkan array string dipisahkan dengan koma, seperti yang ditunjukkan pada contoh berikut.

  ```
  ou-examplerootid111-exampleouid111,ou-examplerootid222-exampleouid222
  ```

- Jalur file: Lokasi file yang berisi daftar tempat untuk membuat atau memperbarui instance kumpulan tumpukan. OrganizationalUnitIds Jika Anda menggunakan jalur file untuk menentukan salah satu akun atau OrganizationalUnitIds, format file harus dalam JSON, seperti yang ditunjukkan pada contoh berikut.

  Masukkan jalur ke file dalam formatInputArtifactName::OrganizationalUnitIdsFileName.

  ```
  SourceArtifact::OU-IDs.txt
  ```

  Contoh berikut menunjukkan isi file untukOU-IDs.txt:

  ```
  [
      "ou-examplerootid111-exampleouid111","ou-examplerootid222-exampleouid222"
  ]
  ```

Daerah

Wajib: Ya

> **Note**
>
> Ketika parameter ini dipilih, Anda juga harus memilih DeploymentTargets.

Daftar AWS Wilayah tempat instance kumpulan tumpukan dibuat atau diperbarui. Wilayah diperbarui dalam urutan di mana mereka dimasukkan.

Masukkan daftar AWS Wilayah yang valid dalam format:Region1,Region2, seperti yang ditunjukkan pada contoh berikut.

```
us-west-2,us-east-1
```

ParameterOverrides

Wajib: Tidak

Daftar parameter set tumpukan yang ingin Anda timpa dalam instance tumpukan yang dipilih. Nilai parameter yang diganti diterapkan ke semua instance tumpukan di akun dan Wilayah yang ditentukan.

Anda dapat memberikan parameter sebagai daftar literal atau jalur file:

- Anda dapat memasukkan parameter dalam format sintaks singkatan berikut:. `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  `ParameterKey=string,ParameterValue=string,UsePreviousValue=boolean,ResolvedV`
  Untuk informasi selengkapnya tentang tipe data ini, lihatJenis data parameter template.

  Contoh berikut menunjukkan parameter bernama `BucketName` dengan nilai`amzn-s3-demo-source-bucket`.

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ```

  Contoh berikut menunjukkan entri dengan beberapa parameter.

  ```
  ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket
  ParameterKey=Asset1,ParameterValue=true
  ParameterKey=Asset2,ParameterValue=true
  ```

- Anda dapat memasukkan lokasi file yang berisi daftar penggantian parameter template yang dimasukkan dalam format`InputArtifactName::ParameterOverridessFileName`, seperti yang ditunjukkan pada contoh berikut.

  ```
  SourceArtifact::parameter-overrides.txt
  ```

  Contoh berikut menunjukkan isi file untuk`parameter-overrides.txt`.

  ```
  [
      {
          "ParameterKey": "KeyName",
          "ParameterValue": "true"
      },
  ```

```
    {
        "ParameterKey": "KeyName",
        "ParameterValue": "true"
    }
]
```

FailureTolerancePercentage

Wajib: Tidak

Persentase akun per Wilayah yang operasi tumpukan ini dapat gagal sebelum CloudFormation menghentikan operasi di Wilayah tersebut. Jika operasi dihentikan di Wilayah, CloudFormation tidak mencoba operasi di Wilayah berikutnya. Saat menghitung jumlah akun berdasarkan persentase yang ditentukan, CloudFormation bulatkan ke nomor bulat berikutnya.

MaxConcurrentPercentage

Wajib: Tidak

Persentase maksimum akun untuk melakukan operasi ini pada satu waktu. Saat menghitung jumlah akun berdasarkan persentase yang ditentukan, CloudFormation bulatkan ke nomor bulat berikutnya. Jika pembulatan ke bawah akan menghasilkan nol, CloudFormation tetapkan angka sebagai satu sebagai gantinya. Meskipun Anda menentukan maksimum, untuk penerapan besar jumlah aktual akun yang ditindaklanjuti secara bersamaan mungkin lebih rendah karena pelambatan layanan.

RegionConcurrencyType

Wajib: Tidak

Anda dapat menentukan apakah kumpulan tumpukan harus diterapkan Wilayah AWS secara berurutan atau paralel dengan mengonfigurasi parameter penerapan konkurensi wilayah. Ketika konkurensi Wilayah ditentukan untuk menyebarkan tumpukan di beberapa Wilayah AWS secara paralel, ini dapat menghasilkan waktu penerapan keseluruhan yang lebih cepat.

- Paralel: Penerapan set tumpukan akan dilakukan pada saat yang sama, selama kegagalan penerapan Wilayah tidak melebihi toleransi kegagalan yang ditentukan.
- Berurutan: Penerapan set tumpukan akan dilakukan satu per satu, selama kegagalan penerapan Wilayah tidak melebihi toleransi kegagalan yang ditentukan. Deployment berurutan adalah pilihan default.

ConcurrencyMode

Wajib: Tidak

Mode konkurensi memungkinkan Anda memilih bagaimana tingkat konkurensi berperilaku selama operasi set tumpukan, baik dengan toleransi kegagalan yang ketat atau lunak. Toleransi Kegagalan Ketat menurunkan kecepatan penerapan karena kegagalan operasi set tumpukan terjadi karena konkurensi menurun untuk setiap kegagalan. Soft Failure Tolerance memprioritaskan kecepatan penerapan sambil tetap memanfaatkan CloudFormation kemampuan keselamatan.

- `STRICT_FAILURE_TOLERANCE`: Opsi ini secara dinamis menurunkan tingkat konkurensi untuk memastikan jumlah akun yang gagal tidak pernah melebihi toleransi kegagalan tertentu. Ini adalah perilaku default.
- `SOFT_FAILURE_TOLERANCE`: Opsi ini memisahkan toleransi kegagalan dari konkurensi yang sebenarnya. Hal ini memungkinkan operasi stack set berjalan pada tingkat konkurensi yang ditetapkan, terlepas dari jumlah kegagalan.

CallAs

Wajib: Tidak

> **Note**
>
> Parameter ini opsional untuk model `SERVICE_MANAGED` izin dan tidak digunakan untuk model `SELF_MANAGED` izin.

Menentukan apakah Anda bertindak di akun manajemen organisasi atau sebagai administrator yang didelegasikan dalam akun anggota.

> **Note**
>
> Jika parameter ini disetel ke`DELEGATED_ADMIN`, pastikan bahwa peran IAM pipeline memiliki `organizations:ListDelegatedAdministrators` izin. Jika tidak, tindakan akan gagal saat berjalan dengan kesalahan yang mirip dengan berikut ini:`Account used is not a delegated administrator`.

- `SELF`: Penyebaran set tumpukan akan menggunakan izin yang dikelola layanan saat masuk ke akun manajemen.
- `DELEGATED_ADMIN`: Penyebaran set tumpukan akan menggunakan izin yang dikelola layanan saat masuk ke akun administrator yang didelegasikan.

Artefak masukan

`CloudFormationStackInstances`dapat berisi artefak yang mencantumkan target dan parameter penerapan.

- Jumlah artefak: `0 to 2`
- Deskripsi: Sebagai masukan, aksi kumpulan tumpukan secara opsional menerima artefak untuk tujuan ini:
  - Untuk menyediakan file parameter yang akan digunakan. (Lihat `ParameterOverrides` parameternya.)
  - Untuk menyediakan file akun target untuk digunakan. (Lihat `DeploymentTargets` parameternya.)

Artefak keluaran

- Jumlah artefak: `0`
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Variabel keluaran

Ketika dikonfigurasi, tindakan ini menghasilkan variabel yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir dalam pipeline. Anda mengonfigurasi tindakan dengan namespace untuk membuat variabel-variabel tersebut tersedia untuk konfigurasi tindakan hilir.

- StackSetId: ID dari set tumpukan.
- OperationId: ID operasi stack set.

Untuk informasi selengkapnya, lihat Referensi variabel.

Contoh konfigurasi tindakan

Contoh berikut menunjukkan konfigurasi tindakan untuk CloudFormationStackInstancestindakan tersebut.

Contoh untuk model izin yang dikelola sendiri

Contoh berikut menunjukkan CloudFormationStackInstancestindakan di mana target penyebaran yang dimasukkan adalah Akun 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"
}
```

Contoh untuk model izin yang dikelola layanan

Contoh berikut menunjukkan CloudFormationStackInstancestindakan untuk model izin yang dikelola layanan di mana target penerapan adalah ID unit AWS organisasi Organizations. `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"
}
```

Izin peran layanan: tindakan **CloudFormationStackSet**

Untuk CloudFormation StackSets tindakan, izin minimum berikut diperlukan.

Untuk `CloudFormationStackSet` tindakan tersebut, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackSet",
        "cloudformation:UpdateStackSet",
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:DescribeStackSet",
        "cloudformation:ListStackInstances"
    ],
    "Resource": "resource_ARN"
},
```

Izin peran layanan: tindakan **CloudFormationStackInstances**

Untuk `CloudFormationStackInstances` tindakan tersebut, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

```
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:CreateStackInstances",
        "cloudformation:DescribeStackSetOperation"
    ],
    "Resource": "resource_ARN"
},
```

Model izin untuk operasi set tumpukan

Karena CloudFormation StackSets melakukan operasi di beberapa akun, Anda harus menentukan izin yang diperlukan di akun tersebut sebelum Anda dapat membuat kumpulan tumpukan. Anda dapat menentukan izin melalui izin yang dikelola sendiri atau izin yang dikelola layanan.

Dengan izin yang dikelola sendiri, Anda membuat dua peran IAM yang diperlukan oleh StackSets - peran administrator seperti AWSCloud FormationStackSetAdministrationRole di akun tempat Anda menentukan kumpulan tumpukan dan peran eksekusi seperti AWSCloud FormationStackSetExecutionRole di setiap akun tempat Anda menerapkan instance kumpulan tumpukan. Menggunakan model izin ini, StackSets dapat menyebarkan ke AWS akun mana pun di mana pengguna memiliki izin untuk membuat peran IAM. Untuk informasi selengkapnya, lihat Memberikan izin yang dikelola sendiri di AWS CloudFormation Panduan Pengguna.

> **Note**
>
> Karena CloudFormation StackSets melakukan operasi di beberapa akun, Anda harus menentukan izin yang diperlukan di akun tersebut sebelum Anda dapat membuat kumpulan tumpukan.

Dengan izin yang dikelola layanan, Anda dapat menerapkan instance tumpukan ke akun yang dikelola oleh Organizations. AWS Dengan menggunakan model izin ini, Anda tidak perlu membuat peran IAM yang diperlukan karena StackSets membuat peran IAM atas nama Anda. Dengan model ini, Anda juga dapat mengaktifkan penerapan otomatis ke akun yang ditambahkan ke organisasi di masa mendatang. Lihat Mengaktifkan akses tepercaya dengan AWS Organizations di Panduan AWS CloudFormation Pengguna.

Jenis data parameter template

Parameter template yang digunakan dalam operasi stack set termasuk tipe data berikut. Untuk informasi selengkapnya, lihat DescribeStackSet.

ParameterKey

- Deskripsi: Kunci yang terkait dengan parameter. Jika Anda tidak menentukan kunci dan nilai untuk parameter tertentu, AWS CloudFormation gunakan nilai default yang ditentukan dalam template.
- Contoh:

```
"ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket"
```

ParameterValue

- Deskripsi: Nilai masukan yang terkait dengan parameter.
- Contoh:

```
"ParameterKey=BucketName,ParameterValue=amzn-s3-demo-source-bucket"
```

UsePreviousValue

- Selama pembaruan tumpukan, gunakan nilai parameter yang ada yang digunakan tumpukan untuk kunci parameter yang diberikan. Jika Anda menentukan`true`, jangan tentukan nilai parameter.
- Contoh:

```
"ParameterKey=Asset1,UsePreviousValue=true"
```

Setiap set tumpukan memiliki satu template dan set parameter template. Saat Anda memperbarui parameter template atau template, Anda memperbaruinya untuk seluruh rangkaian. Kemudian semua status instance disetel ke USANG hingga perubahan diterapkan ke instance itu.

Untuk mengganti nilai parameter pada instance tertentu, misalnya, jika templat berisi parameter untuk `stage` dengan nilai`prod`, Anda dapat mengganti nilai parameter tersebut menjadi atau. `beta` `gamma`

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Jenis parameter - Bab referensi di Panduan AWS CloudFormation Pengguna ini memberikan lebih banyak deskripsi dan contoh untuk parameter CloudFormation templat.
- Praktik terbaik — Untuk informasi selengkapnya tentang praktik terbaik untuk menerapkan kumpulan tumpukan, lihat https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-bestpractices.html di Panduan AWS CloudFormation Pengguna.
- AWS CloudFormation Referensi API - Anda dapat mereferensikan CloudFormation tindakan berikut di Referensi AWS CloudFormation API untuk informasi selengkapnya tentang parameter yang digunakan dalam operasi kumpulan tumpukan:

- CreateStackSetTindakan membuat set tumpukan.
- UpdateStackSetTindakan ini memperbarui kumpulan tumpukan dan instance tumpukan terkait di akun dan Wilayah yang ditentukan. Bahkan jika operasi set tumpukan yang dibuat dengan memperbarui kumpulan tumpukan gagal (sepenuhnya atau sebagian, di bawah atau di atas toleransi kegagalan yang ditentukan), kumpulan tumpukan diperbarui dengan perubahan ini. CreateStackInstances Panggilan selanjutnya pada set tumpukan yang ditentukan menggunakan set tumpukan yang diperbarui.
- CreateStackInstancesTindakan ini membuat instance tumpukan untuk semua wilayah tertentu dalam semua akun yang ditentukan pada model izin yang dikelola sendiri, atau dalam semua target penerapan yang ditentukan pada model izin yang dikelola layanan. Anda dapat mengganti parameter untuk instance yang dibuat oleh tindakan ini. Jika instance sudah ada, CreateStackInstances panggilan UpdateStackInstances dengan parameter input yang sama. Saat Anda menggunakan tindakan ini untuk membuat instance, tindakan ini tidak mengubah status instance tumpukan lainnya.
- UpdateStackInstancesTindakan ini memperbarui instance tumpukan dengan kumpulan tumpukan untuk semua wilayah tertentu dalam semua akun tertentu pada model izin yang dikelola sendiri, atau dalam semua target penerapan yang ditentukan pada model izin yang dikelola layanan. Anda dapat mengganti parameter untuk instance yang diperbarui oleh tindakan ini. Saat Anda menggunakan tindakan ini untuk memperbarui subset instance, tindakan ini tidak mengubah status instance tumpukan lainnya.
- DescribeStackSetOperationTindakan mengembalikan deskripsi operasi stack set yang ditentukan.
- DescribeStackSetTindakan mengembalikan deskripsi set stack yang ditentukan.

AWS CodeBuild membangun dan menguji referensi tindakan

Memungkinkan Anda menjalankan build dan pengujian sebagai bagian dari pipeline Anda. Saat Anda menjalankan tindakan CodeBuild build atau test, perintah yang ditentukan dalam buildspec dijalankan di dalam container. CodeBuild Semua artefak yang ditetapkan sebagai artefak masukan untuk suatu CodeBuild tindakan tersedia di dalam wadah yang menjalankan perintah. CodeBuild dapat memberikan tindakan build atau test. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS CodeBuild.

Saat Anda menggunakan CodePipeline wizard di konsol untuk membuat proyek build, proyek CodeBuild build akan menunjukkan penyedia sumbernya CodePipeline. Saat membuat proyek build

di CodeBuild konsol, Anda tidak dapat menentukan CodePipeline sebagai penyedia sumber, tetapi menambahkan tindakan build ke pipeline akan menyesuaikan sumber di CodeBuild konsol. Untuk informasi selengkapnya, lihat ProjectSource di dalam Referensi API AWS CodeBuild .

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Izin peran layanan: tindakan CodeBuild
- Deklarasi tindakan (CodeBuild contoh)
- Lihat juga

Tipe tindakan

- Kategori: `Build` atau `Test`
- Pemilik: `AWS`
- Penyedia: `CodeBuild`
- Versi: `1`

Parameter konfigurasi

ProjectName

Wajib: Ya

`ProjectName`adalah nama proyek pembangunan di CodeBuild.

PrimarySource

Diperlukan: Kondisional

Nilai `PrimarySource` parameter harus menjadi nama salah satu artefak input untuk tindakan. CodeBuild mencari file buildspec dan menjalankan perintah buildspec di direktori yang berisi versi unzip artefak ini.

Parameter ini diperlukan jika beberapa artefak masukan ditentukan untuk suatu CodeBuild tindakan. Ketika hanya ada satu artefak sumber untuk aksi, artefak default ke `PrimarySource` artefak itu.

BatchEnabled

Wajib: Tidak

Nilai Boolean dari `BatchEnabled` parameter memungkinkan tindakan untuk menjalankan beberapa build dalam eksekusi build yang sama.

Ketika opsi ini diaktifkan, `CombineArtifacts` opsi tersedia.

Untuk contoh pipeline dengan build batch diaktifkan, lihat CodePipeline integrasi dengan CodeBuild dan build batch.

BuildspecOverride

Wajib: Tidak

Definisi buildspec sebaris atau deklarasi file buildspec yang mengganti definisi terbaru yang ditentukan dalam proyek build, hanya untuk build ini. Buildspec yang didefinisikan pada proyek tidak diubah.

Jika nilai ini diatur, itu bisa menjadi salah satu dari yang berikut:

- Definisi buildspec sebaris. Untuk informasi selengkapnya, lihat referensi sintaks di sintaks Buildspec.
- Jalur ke file buildspec alternatif relatif terhadap nilai variabel `CODEBUILD_SRC_DIR` lingkungan bawaan atau jalur ke bucket S3. Ember harus Wilayah AWS sama dengan proyek pembangunan. Tentukan file buildspec menggunakan ARN (misalnya,). `arn:aws:s3:::my-codebuild-sample2/buildspec.yml` Jika nilai ini tidak disediakan atau diatur ke string kosong, kode sumber harus berisi file buildspec di direktori root nya. Untuk informasi selengkapnya tentang menambahkan jalur, lihat Nama File Buildspec dan Lokasi Penyimpanan.

> **Note**
>
> Karena properti ini memungkinkan Anda untuk mengubah perintah build yang akan berjalan di container, Anda harus mencatat bahwa prinsipal IAM dengan kemampuan untuk memanggil API ini dan menyetel parameter ini dapat mengganti pengaturan default. Selain itu, kami menganjurkan agar Anda menggunakan lokasi buildspec yang dapat dipercaya seperti file di repositori sumber Anda atau bucket Amazon S3.

CombineArtifacts

Wajib: Tidak

Nilai Boolean `CombineArtifacts` parameter menggabungkan semua artefak build dari build batch menjadi file artefak tunggal untuk aksi build.

Untuk menggunakan opsi ini, `BatchEnabled` parameter harus diaktifkan.

EnvironmentVariables

Wajib: Tidak

Nilai parameter ini digunakan untuk mengatur variabel lingkungan untuk CodeBuild tindakan di pipeline Anda. Nilai untuk `EnvironmentVariables` parameter mengambil bentuk array JSON dari objek variabel lingkungan. Lihat parameter contoh diDeklarasi tindakan (CodeBuild contoh).

Setiap objek memiliki tiga bagian, yang semuanya adalah string:

- `name`: Nama atau kunci variabel lingkungan.
- `value`: Nilai variabel lingkungan. Saat menggunakan `SECRETS_MANAGER` tipe `PARAMETER_STORE` atau, nilai ini harus berupa nama parameter yang telah Anda simpan di AWS Systems Manager Parameter Store atau rahasia yang telah Anda simpan di AWS Secrets Manager, masing-masing.

> **Note**
>
> Kami sangat tidak menyarankan penggunaan variabel lingkungan untuk menyimpan nilai sensitif, terutama AWS kredensyal. Saat Anda menggunakan CodeBuild konsol atau AWS CLI, variabel lingkungan ditampilkan dalam teks biasa. Untuk nilai sensitif, kami sarankan Anda menggunakan `SECRETS_MANAGER` tipe sebagai gantinya.

- `type`: (Opsional) Jenis variabel lingkungan. Nilai yang valid adalah `PARAMETER_STORE`, `SECRETS_MANAGER`, atau `PLAINTEXT`. Ketika tidak ditentukan, ini default ke. `PLAINTEXT`

> **Note**
>
> Saat Anda memasukkan`name`,`value`, dan `type` untuk konfigurasi variabel lingkungan Anda, terutama jika variabel lingkungan berisi sintaks variabel CodePipeline keluaran, jangan melebihi batas 1000 karakter untuk bidang nilai konfigurasi. Kesalahan validasi dikembalikan ketika batas ini terlampaui.

Untuk informasi selengkapnya, lihat EnvironmentVariabledi Referensi AWS CodeBuild API. Untuk contoh CodeBuild tindakan dengan variabel lingkungan yang menyelesaikan ke nama GitHub cabang, lihat. Contoh: Gunakan BranchName variabel dengan variabel CodeBuild lingkungan

Artefak masukan

- Jumlah artefak: `1 to 5`
- Deskripsi: CodeBuild mencari file buildspec dan menjalankan perintah buildspec dari direktori artefak sumber utama. Ketika salah satu sumber input ditentukan atau ketika lebih dari satu sumber input ditentukan untuk CodeBuild tindakan, artefak tunggal, atau artefak utama dalam kasus beberapa sumber input, harus diatur menggunakan parameter konfigurasi `PrimarySource` tindakan di. CodePipeline

  Setiap artefak input diekstraksi ke direktorinya sendiri, lokasi yang disimpan dalam variabel lingkungan. Direktori untuk artefak sumber utama tersedia dengan`$CODEBUILD_SRC_DIR`. Direktori untuk semua artefak input lainnya tersedia dengan. `$CODEBUILD_SRC_DIR_yourInputArtifactName`

> **Note**
>
> Artefak yang dikonfigurasi dalam CodeBuild proyek Anda menjadi artefak input yang digunakan oleh CodeBuild aksi di pipeline Anda.

Artefak keluaran

- Jumlah artefak: `0 to 5`
- Deskripsi: Ini dapat digunakan untuk membuat artefak yang didefinisikan dalam file CodeBuild buildspec tersedia untuk tindakan selanjutnya dalam pipeline. Ketika hanya satu artefak keluaran yang didefinisikan, artefak ini dapat didefinisikan langsung di bawah `artifacts` bagian file buildspec. Ketika lebih dari satu artefak keluaran ditentukan, semua artefak yang direferensikan harus didefinisikan sebagai artefak sekunder dalam file buildspec. Nama-nama artefak keluaran CodePipeline harus cocok dengan pengidentifikasi artefak dalam file buildspec.

> **Note**
>
> Artefak yang dikonfigurasi dalam CodeBuild proyek Anda menjadi artefak CodePipeline masukan dalam aksi pipeline Anda.

Jika `CombineArtifacts` parameter dipilih untuk build batch, lokasi artefak keluaran berisi artefak gabungan dari beberapa build yang dijalankan dalam eksekusi yang sama.

Variabel keluaran

Tindakan ini akan menghasilkan sebagai variabel semua variabel lingkungan yang diekspor sebagai bagian dari build. Untuk detail selengkapnya tentang cara mengekspor variabel lingkungan, lihat EnvironmentVariabledi Panduan AWS CodeBuild API.

Untuk informasi selengkapnya tentang penggunaan variabel CodeBuild lingkungan di CodePipeline, lihat contoh diCodeBuild variabel keluaran aksi. Untuk daftar variabel lingkungan yang dapat Anda gunakan CodeBuild, lihat Variabel lingkungan di lingkungan build di Panduan AWS CodeBuild Pengguna.

Izin peran layanan: tindakan CodeBuild

Untuk CodeBuild dukungan, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "codebuild:BatchGetBuilds",
                "codebuild:StartBuild",
                "codebuild:BatchGetBuildBatches",
                "codebuild:StartBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:*:111122223333:project/[[ProjectName]]"
```

```
            ],
            "Effect": "Allow"
        }
    ]
}
```

Deklarasi tindakan (CodeBuild contoh)

YAML

```
Name: Build
Actions:
  - Name: PackageExport
    ActionTypeId:
      Category: Build
      Owner: AWS
      Provider: CodeBuild
      Version: '1'
    RunOrder: 1
    Configuration:
      BatchEnabled: 'true'
      CombineArtifacts: 'true'
      ProjectName: my-build-project
      PrimarySource: MyApplicationSource1
      EnvironmentVariables:
 '[{"name":"TEST_VARIABLE","value":"TEST_VALUE","type":"PLAINTEXT"},
{"name":"ParamStoreTest","value":"PARAMETER_NAME","type":"PARAMETER_STORE"}]'
    OutputArtifacts:
      - Name: MyPipeline-BuildArtifact
    InputArtifacts:
      - Name: MyApplicationSource1
      - Name: MyApplicationSource2
```

JSON

```
{
    "Name": "Build",
    "Actions": [
        {
            "Name": "PackageExport",
            "ActionTypeId": {
```

```
                "Category": "Build",
                "Owner": "AWS",
                "Provider": "CodeBuild",
                "Version": "1"
            },
            "RunOrder": 1,
            "Configuration": {
                "BatchEnabled": "true",
                "CombineArtifacts": "true",
                "ProjectName": "my-build-project",
                "PrimarySource": "MyApplicationSource1",
                "EnvironmentVariables": "[{\"name\":\"TEST_VARIABLE\",\"value\":
\"TEST_VALUE\",\"type\":\"PLAINTEXT\"},{\"name\":\"ParamStoreTest\",\"value\":
\"PARAMETER_NAME\",\"type\":\"PARAMETER_STORE\"}]"
            },
            "OutputArtifacts": [
                {
                    "Name": "MyPipeline-BuildArtifact"
                }
            ],
            "InputArtifacts": [
                {
                    "Name": "MyApplicationSource1"
                },
                {
                    "Name": "MyApplicationSource2"
                }
            ]
        }
    ]
}
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- AWS CodeBuild Panduan Pengguna — Untuk contoh pipeline dengan CodeBuild tindakan, lihat Menggunakan CodePipeline dengan CodeBuild untuk Menguji Kode dan Menjalankan Build. Untuk contoh proyek dengan beberapa CodeBuild artefak input dan output, lihat CodePipelineIntegrasi dengan CodeBuild dan Beberapa Sumber Input dan Contoh Artefak Output dan Beberapa Sumber Input dan Contoh Artefak Output.

- Tutorial: Membuat pipeline yang membangun dan menguji aplikasi Android Anda AWS Device Farm— Tutorial ini menyediakan contoh file buildspec dan contoh aplikasi untuk membuat pipeline dengan GitHub sumber yang membangun dan menguji aplikasi Android dengan dan. CodeBuild AWS Device Farm
- Referensi Spesifikasi Bangun untuk CodeBuild — Topik referensi ini memberikan definisi dan contoh untuk memahami file CodeBuild buildspec. Untuk daftar variabel lingkungan yang dapat Anda gunakan CodeBuild, lihat Variabel lingkungan di lingkungan build di Panduan AWS CodeBuild Pengguna.

AWS CodePipeline memanggil referensi tindakan

Anda menggunakan tindakan CodePipeline pemanggilan untuk menyederhanakan pemicu eksekusi pipa hilir dan meneruskan variabel pipeline dan revisi sumber antar pipeline.

> **Note**
>
> Tindakan ini hanya didukung untuk pipeline tipe V2.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Izin kebijakan peran layanan untuk tindakan CodePipeline pemanggilan
- Deklarasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Invoke`
- Pemilik: `AWS`
- Penyedia: `CodePipeline`

- Versi: 1

Parameter konfigurasi

PipelineName

Wajib: Ya

Nama pipa yang akan, setelah berjalan, memulai pipa target saat ini. Anda harus sudah membuat pipeline pemanggilan. Tindakan akan memulai pipeline `s3-pipeline-test` (target) saat pipeline (pemanggilan) bernama `my-s3-pipeline` memulai eksekusi.

SourceRevisions

Wajib: Tidak

Revisi sumber yang Anda inginkan untuk digunakan pipeline target saat dimulai oleh pipeline pemanggilan. Misalnya, aksi sumber S3 menyediakan variabel keluaran seperti ID Versi S3 dan Kunci Objek. Anda dapat menentukan nilai revisi yang akan digunakan saat pipeline dipanggil.

Untuk CLI, Anda menentukan revisi sumber sebagai string JSON serial. Untuk informasi selengkapnya tentang penggunaan penggantian revisi sumber, lihat SourceRevisionOverridedi Panduan API. CodePipeline

Pemetaan menggunakan format string seperti yang ditunjukkan pada contoh berikut:

```
[{"actionName":"Source","revisionType":"S3_OBJECT_VERSION_ID","revision
Value":"zq8mjNEXAMPLE"}]
```

Variabel

Wajib: Tidak

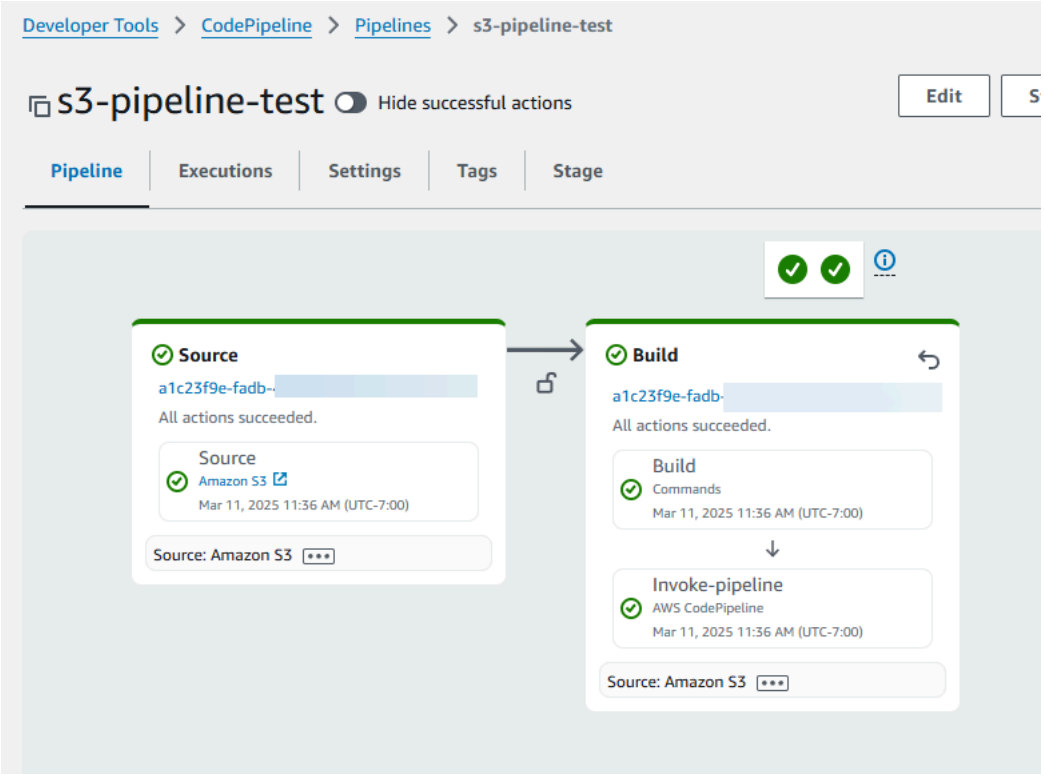
Nama dan nilai variabel yang Anda ingin tindakan untuk mendukung.

Untuk CLI, Anda menentukan variabel sebagai string JSON serial. Untuk informasi selengkapnya tentang penggunaan variabel pipeline, lihat PipelineVariabledi Panduan CodePipeline API.

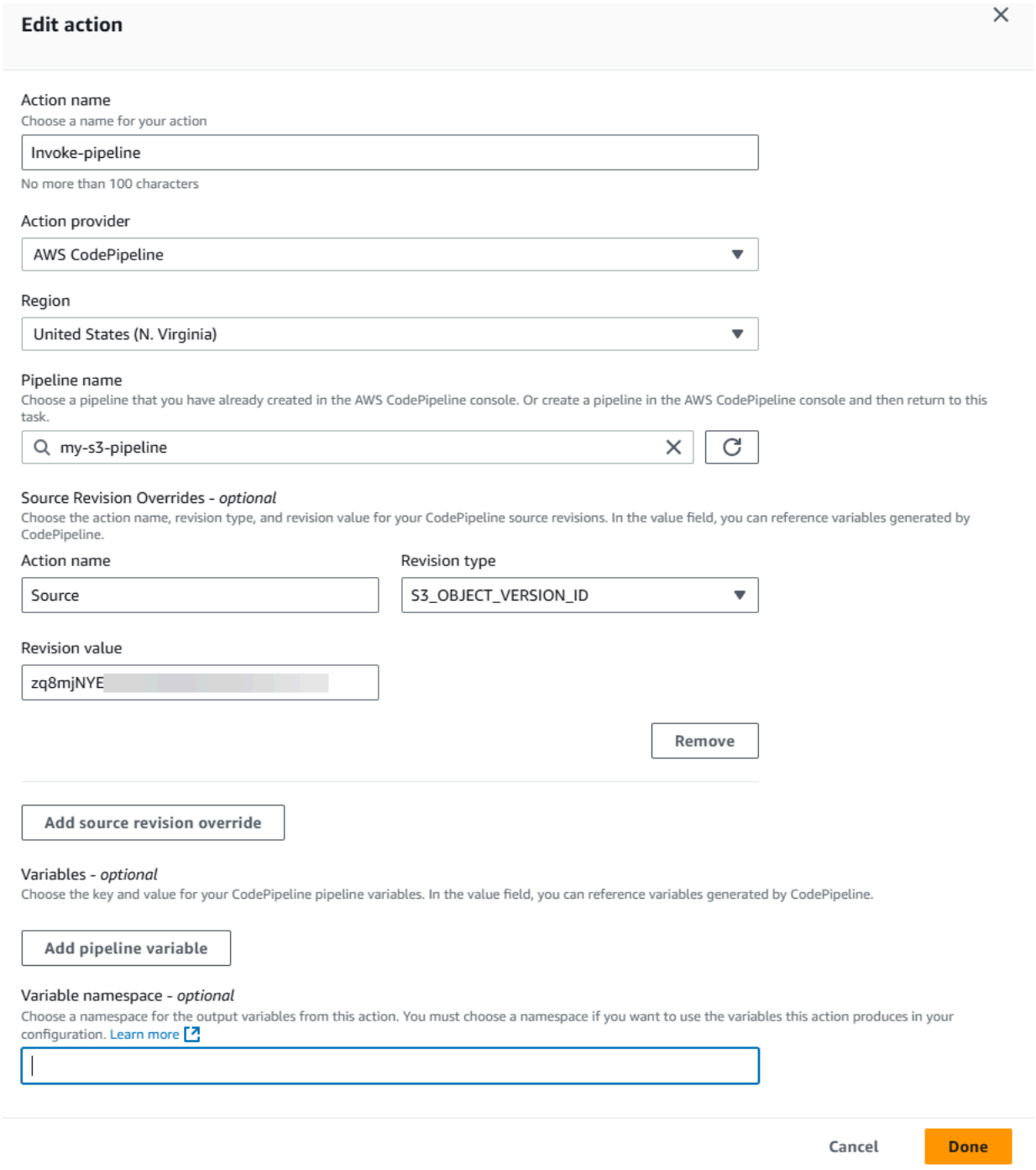
Pemetaan menggunakan format string seperti yang ditunjukkan pada contoh berikut:

```
[{"name":"VAR1","value":"VALUE1"}]
```

Gambar berikut menunjukkan contoh tindakan yang ditambahkan ke pipeline di konsol.



Gambar berikut menunjukkan contoh halaman Edit untuk tindakan tersebut. Dalam contoh berikut, pipeline bernama `s3-pipeline-test` memiliki tindakan pemanggilan pipeline yang dikonfigurasi seperti yang ditunjukkan untuk konsol. Tindakan akan memulai `s3-pipeline-test` pipeline ketika pipeline bernama `my-s3-pipeline` menyelesaikan eksekusi. Contoh menunjukkan bahwa penggantian revisi sumber untuk penggantian sumber S3_OBJECT_VERSION_ID dengan nilai revisi tertentu dari. `zq8mjNYEexample`



Artefak masukan

- Jumlah artefak: `0`
- Deskripsi: Artefak masukan tidak berlaku untuk jenis tindakan ini.

Artefak keluaran

- Jumlah artefak: `0`
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Izin kebijakan peran layanan untuk tindakan CodePipeline pemanggilan

Saat CodePipeline menjalankan tindakan, kebijakan peran CodePipeline layanan memerlukan `codepipeline:StartPipelineExecution` izin, yang dicakup dengan tepat ke ARN sumber daya pipeline untuk mempertahankan akses dengan hak istimewa paling sedikit.

```
{
           "Sid": "StatementForPipelineInvokeAction",
           "Effect": "Allow",
           "Action": "codepipeline:StartPipelineExecution",
           "Resource": [
               "arn:aws:codepipeline:{{region}}:{{AccountId}}:{{pipelineName}}"
           ]
       }
```

Deklarasi tindakan

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\"}]"
    }
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Mulai pipeline dengan penggantian revisi sumber— Bagian ini menjelaskan memulai pipa dengan revisi sumber secara manual atau melalui transformator input EventBridge peristiwa.

CodeCommit referensi tindakan sumber

Memulai pipeline saat komit baru dibuat pada CodeCommit repositori dan cabang yang dikonfigurasi.

Jika Anda menggunakan konsol untuk membuat atau mengedit pipeline, CodePipeline buat EventBridge aturan yang memulai pipeline saat terjadi perubahan di repositori.

> **Note**
>
> Untuk Amazon ECR, Amazon S3, CodeCommit atau sumber, Anda juga dapat membuat penggantian sumber menggunakan entri transformasi masukan untuk menggunakan `revisionValue` peristiwa EventBridge in untuk pipeline Anda, yang berasal dari variabel peristiwa sumber untuk kunci objek, komit, atau ID gambar Anda. `revisionValue` Untuk informasi selengkapnya, lihat langkah opsional untuk entri transformasi masukan yang

disertakan dalam prosedur di bawahTindakan sumber dan sumber daya Amazon ECR EventBridge,Menyambungkan ke tindakan sumber Amazon S3 dengan sumber yang diaktifkan untuk acara, atauCodeCommit tindakan sumber dan EventBridge.

Anda harus sudah membuat CodeCommit repositori sebelum menghubungkan pipeline melalui tindakan. CodeCommit

Setelah perubahan kode terdeteksi, Anda memiliki opsi berikut untuk meneruskan kode ke tindakan selanjutnya:

- Default - Mengkonfigurasi tindakan CodeCommit sumber untuk menampilkan file ZIP dengan salinan dangkal dari komit Anda.
- Full clone — Mengonfigurasi aksi sumber untuk menampilkan referensi URL Git ke repositori untuk tindakan selanjutnya.

  Saat ini, referensi URL Git hanya dapat digunakan oleh CodeBuild tindakan hilir untuk mengkloning repo dan metadata Git terkait. Mencoba meneruskan referensi URL Git ke CodeBuild non-tindakan menghasilkan kesalahan.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Izin peran layanan: tindakan CodeCommit
- Contoh konfigurasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Source`
- Pemilik: `AWS`
- Penyedia: `CodeCommit`

- Versi: 1

Parameter konfigurasi

RepositoryName

Wajib: Ya

Nama repositori tempat perubahan sumber akan dideteksi.

BranchName

Wajib: Ya

Nama cabang tempat perubahan sumber harus dideteksi.

PollForSourceChanges

Wajib: Tidak

`PollForSourceChanges`mengontrol apakah CodePipeline polling CodeCommit repositori untuk perubahan sumber. Kami menyarankan Anda menggunakan CloudWatch Acara untuk mendeteksi perubahan sumber sebagai gantinya. Untuk informasi selengkapnya tentang mengonfigurasi CloudWatch Acara, lihat Migrasi jalur pemungutan suara (CodeCommit sumber) (CLI) atauMigrasikan jalur pemungutan suara (CodeCommit sumber) (templat)CloudFormation.

> **Important**
>
> Jika ingin mengonfigurasi aturan CloudWatch Peristiwa, Anda harus mengatur `PollForSourceChanges false` untuk menghindari eksekusi pipeline duplikat.

Nilai yang valid untuk parameter ini:

- `true`: Jika disetel, CodePipeline polling repositori Anda untuk perubahan sumber.

> **Note**
>
> Jika Anda menghilangkan`PollForSourceChanges`, CodePipeline default untuk polling repositori Anda untuk perubahan sumber. Perilaku ini sama seperti jika `PollForSourceChanges` disertakan dan disetel ke`true`.

- `false`: Jika disetel, CodePipeline tidak melakukan polling repositori Anda untuk perubahan sumber. Gunakan setelan ini jika Anda ingin mengonfigurasi aturan CloudWatch Peristiwa untuk mendeteksi perubahan sumber.

OutputArtifactFormat

Wajib: Tidak

Format artefak keluaran. Nilai dapat berupa `CODEBUILD_CLONE_REF` atau`CODE_ZIP`. Jika tidak ditentukan, defaultnya adalah`CODE_ZIP`.

> **Important**
>
> `CODEBUILD_CLONE_REF`Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir. Jika Anda memilih opsi ini, Anda perlu menambahkan `codecommit:GitPull` izin ke peran CodeBuild layanan Anda seperti yang ditunjukkan pada[Tambahkan CodeBuild GitClone izin untuk tindakan CodeCommit sumber](). Anda juga perlu menambahkan `codecommit:GetRepository` izin ke peran CodePipeline layanan Anda seperti yang ditunjukkan pada[Menambahkan izin ke peran CodePipeline layanan](). Untuk tutorial yang menunjukkan cara menggunakan opsi klon Penuh, lihat[Tutorial: Gunakan klon lengkap dengan sumber CodeCommit pipa]().

Artefak masukan

- Jumlah artefak: `0`
- Deskripsi: Artefak masukan tidak berlaku untuk jenis tindakan ini.

Artefak keluaran

- Jumlah artefak: `1`
- Deskripsi: Artefak keluaran dari tindakan ini adalah file ZIP yang berisi konten repositori dan cabang yang dikonfigurasi pada komit yang ditentukan sebagai revisi sumber untuk eksekusi pipeline. Artefak yang dihasilkan dari repositori adalah artefak keluaran untuk tindakan tersebut. CodeCommit ID komit kode sumber ditampilkan CodePipeline sebagai revisi sumber untuk eksekusi pipeline yang dipicu.

Variabel keluaran

Ketika dikonfigurasi, tindakan ini menghasilkan variabel yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir dalam pipeline. Tindakan ini menghasilkan variabel yang dapat dilihat sebagai variabel keluaran, bahkan jika tindakan tidak memiliki namespace. Anda mengonfigurasi tindakan dengan namespace untuk membuat variabel-variabel tersebut tersedia untuk konfigurasi tindakan hilir.

Untuk informasi selengkapnya, lihat Referensi variabel.

CommitId

ID CodeCommit komit yang memicu eksekusi pipeline. Komit IDs adalah SHA penuh dari komit.

CommitMessage

Pesan deskripsi, jika ada, terkait dengan komit yang memicu eksekusi pipeline.

RepositoryName

Nama CodeCommit repositori tempat komit yang memicu pipeline dibuat.

BranchName

Nama cabang untuk CodeCommit repositori tempat perubahan sumber dilakukan.

AuthorDate

Tanggal ketika komit ditulis, dalam format stempel waktu.

CommitterDate

Tanggal ketika komit dilakukan, dalam format stempel waktu.

Izin peran layanan: tindakan CodeCommit

Saat CodePipeline menjalankan tindakan, kebijakan peran CodePipeline layanan memerlukan izin berikut, yang dicakup dengan tepat ke ARN sumber daya pipeline untuk mempertahankan akses dengan hak istimewa paling sedikit. Misalnya, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

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]]"
            ]
        }
    ]
}
```

Contoh konfigurasi tindakan

Contoh untuk format artefak keluaran default

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"
}
```

Contoh untuk format artefak keluaran klon penuh

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"
}
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Tutorial: Buat pipeline sederhana (CodeCommit repositori)— Tutorial ini menyediakan contoh file spesifikasi aplikasi dan contoh CodeDeploy aplikasi dan kelompok penyebaran. Gunakan tutorial ini untuk membuat pipeline dengan CodeCommit sumber yang menyebarkan ke instans Amazon EC2.

AWS CodeDeploy menyebarkan referensi tindakan

Anda menggunakan AWS CodeDeploy tindakan untuk menyebarkan kode aplikasi ke armada penyebaran Anda. Armada penerapan Anda dapat terdiri dari instans Amazon EC2, instans lokal, atau keduanya.

> **Note**
>
> Topik referensi ini menjelaskan tindakan CodeDeploy penerapan CodePipeline di mana platform penerapan adalah Amazon EC2. Untuk informasi referensi tentang Amazon Elastic Container Service untuk tindakan CodeDeploy blue/green penerapan CodePipeline, lihatAmazon Elastic Container Service dan CodeDeploy referensi aksi penerapan biru-hijau.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Izin peran layanan: tindakan AWS CodeDeploy
- Deklarasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `CodeDeploy`
- Versi: 1

Parameter konfigurasi

ApplicationName

Wajib: Ya

Nama aplikasi yang Anda buat CodeDeploy.

DeploymentGroupName

Wajib: Ya

Grup penyebaran yang Anda buat. CodeDeploy

Artefak masukan

- Jumlah artefak: 1
- Deskripsi: AppSpec File yang CodeDeploy digunakan untuk menentukan:
  - Apa yang harus diinstal ke instans Anda dari revisi aplikasi Anda di Amazon S3 atau. GitHub
  - Peristiwa siklus hidup mana yang akan dijalankan sebagai respons terhadap peristiwa siklus hidup penerapan.

Untuk informasi selengkapnya tentang AppSpec file, lihat Referensi CodeDeploy AppSpec File.

Artefak keluaran

- Jumlah artefak: `0`
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Izin peran layanan: tindakan AWS CodeDeploy

Untuk AWS CodeDeploy dukungan, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

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

Deklarasi tindakan

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"
        }
    ]
```

```
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Tutorial: Buat pipeline sederhana (ember S3)— Tutorial ini memandu Anda melalui pembuatan bucket sumber, instans EC2, dan CodeDeploy sumber daya untuk menyebarkan aplikasi sampel. Anda kemudian membuat pipeline dengan tindakan CodeDeploy penerapan yang menerapkan kode yang dikelola di bucket S3 ke instans Amazon EC2.
- Tutorial: Buat pipeline sederhana (CodeCommit repositori)— Tutorial ini memandu Anda melalui pembuatan repositori CodeCommit sumber Anda, instans EC2, dan CodeDeploy sumber daya untuk menyebarkan aplikasi sampel. Anda kemudian membuat pipeline dengan tindakan CodeDeploy penerapan yang menerapkan kode dari CodeCommit repositori ke instans Amazon EC2.
- CodeDeploy AppSpec Referensi File — Bab referensi dalam Panduan AWS CodeDeploy Pengguna ini memberikan informasi referensi dan contoh untuk CodeDeploy AppSpec file.

CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri

Tindakan sumber untuk koneksi didukung oleh AWS CodeConnections. CodeConnections memungkinkan Anda untuk membuat dan mengelola koneksi antara AWS sumber daya dan repositori pihak ketiga seperti. GitHub Memulai pipeline saat komit baru dibuat pada repositori kode sumber pihak ketiga. Tindakan sumber mengambil perubahan kode saat pipeline dijalankan secara manual atau saat peristiwa webhook dikirim dari penyedia sumber.

Anda dapat mengonfigurasi tindakan dalam pipeline Anda untuk menggunakan konfigurasi Git yang memungkinkan Anda memulai pipeline dengan pemicu. Untuk mengonfigurasi konfigurasi pemicu pipeline untuk memfilter dengan pemicu, lihat detail selengkapnya diTambahkan pemicu dengan tipe acara permintaan push atau pull kode.

> **Note**
>
> Fitur ini tidak tersedia di Asia Pasifik (Hong Kong), Asia Pasifik (Hyderabad), Asia Pasifik (Jakarta), Asia Pasifik (Melbourne), Asia Pasifik (Osaka), Afrika (Cape Town), Timur Tengah (Bahrain), Timur Tengah (UEA), Eropa (Spanyol), Eropa (Zurich), Israel (Tel Aviv), atau (AS-Barat) Wilayah. AWS GovCloud Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline. Untuk pertimbangan dengan tindakan ini di Wilayah Eropa (Milan), lihat catatan diCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Koneksi dapat mengaitkan AWS sumber daya Anda dengan repositori pihak ketiga berikut:

- Bitbucket Cloud (melalui opsi penyedia Bitbucket di CodePipeline konsol atau `Bitbucket` penyedia di CLI)

  > **Note**
  >
  > Anda dapat membuat koneksi ke repositori Bitbucket Cloud. Jenis penyedia Bitbucket yang diinstal, seperti Bitbucket Server, tidak didukung.

- > **Note**
  >
  > Jika Anda menggunakan ruang kerja Bitbucket, Anda harus memiliki akses administrator untuk membuat koneksi.

- GitHub dan GitHub Enterprise Cloud (melalui opsi penyedia GitHub (melalui GitHub Aplikasi) di CodePipeline konsol atau `GitHub` penyedia di CLI)

  > **Note**
  >
  > Jika repositori Anda ada di GitHub organisasi, Anda harus menjadi pemilik organisasi untuk membuat koneksi. Jika Anda menggunakan repositori yang tidak ada dalam organisasi, Anda harus menjadi pemilik repositori.

- GitHub Server Perusahaan (melalui opsi penyedia Server GitHub Perusahaan di CodePipeline konsol atau `GitHub Enterprise Server` penyedia di CLI)

- GitLab.com (melalui opsi GitLabpenyedia di CodePipeline konsol atau `GitLab` penyedia di CLI)

> **Note**
>
> Anda dapat membuat koneksi ke repositori tempat Anda memiliki peran Pemilik GitLab, dan kemudian koneksi dapat digunakan dengan repositori dengan sumber daya seperti. CodePipeline Untuk repositori dalam grup, Anda tidak perlu menjadi pemilik grup.

- Instalasi yang dikelola sendiri untuk GitLab (Edisi Perusahaan atau Edisi Komunitas) (melalui opsi penyedia yang GitLab dikelola sendiri di CodePipeline konsol atau `GitLabSelfManaged` penyedia di CLI)

> **Note**
>
> Setiap koneksi mendukung semua repositori yang Anda miliki dengan penyedia itu. Anda hanya perlu membuat koneksi baru untuk setiap jenis penyedia.

Koneksi memungkinkan pipeline Anda mendeteksi perubahan sumber melalui aplikasi penginstalan penyedia pihak ketiga. Misalnya, webhook digunakan untuk berlangganan jenis GitHub acara dan dapat diinstal pada organisasi, repositori, atau Aplikasi. GitHub Koneksi Anda menginstal webhook repositori di GitHub Aplikasi Anda yang berlangganan peristiwa jenis push. GitHub

Setelah perubahan kode terdeteksi, Anda memiliki opsi berikut untuk meneruskan kode ke tindakan selanjutnya:

- Default: Seperti tindakan CodePipeline sumber lain yang ada, `CodeStarSourceConnection` dapat menampilkan file ZIP dengan salinan dangkal dari komit Anda.
- Klon lengkap: juga `CodeStarSourceConnection` dapat dikonfigurasi untuk menampilkan referensi URL ke repo untuk tindakan selanjutnya.

  Saat ini, referensi URL Git hanya dapat digunakan oleh CodeBuild tindakan hilir untuk mengkloning repo dan metadata Git terkait. Mencoba meneruskan referensi URL Git ke CodeBuild non-tindakan menghasilkan kesalahan.

CodePipeline meminta Anda untuk menambahkan aplikasi penginstalan AWS Konektor ke akun pihak ketiga saat Anda membuat koneksi. Anda harus sudah membuat akun dan repositori penyedia pihak ketiga Anda sebelum Anda dapat terhubung melalui tindakan. `CodeStarSourceConnection`

> **Note**
>
> Untuk membuat atau melampirkan kebijakan ke peran Anda dengan izin yang diperlukan untuk menggunakan AWS CodeStar koneksi, lihat Referensi izin koneksi. Bergantung pada kapan peran CodePipeline layanan dibuat, Anda mungkin perlu memperbarui izinnya untuk mendukung AWS CodeStar koneksi. Untuk petunjuk, lihat Menambahkan izin ke peran CodePipeline layanan.

> **Note**
>
> Untuk menggunakan koneksi di Eropa (Milan) Wilayah AWS, Anda harus:
>
> 1. Instal aplikasi khusus Wilayah
> 2. Aktifkan Wilayah
>
> Aplikasi khusus Wilayah ini mendukung koneksi di Wilayah Eropa (Milan). Ini diterbitkan di situs penyedia pihak ketiga, dan terpisah dari aplikasi yang ada yang mendukung koneksi untuk Wilayah lain. Dengan menginstal aplikasi ini, Anda memberi wewenang kepada penyedia pihak ketiga untuk membagikan data Anda dengan layanan untuk Wilayah ini saja, dan Anda dapat mencabut izin kapan saja dengan mencopot pemasangan aplikasi. Layanan tidak akan memproses atau menyimpan data Anda kecuali Anda mengaktifkan Wilayah. Dengan mengaktifkan Wilayah ini, Anda memberikan izin layanan kami untuk memproses dan menyimpan data Anda.
> Meskipun Wilayah tidak diaktifkan, penyedia pihak ketiga masih dapat membagikan data Anda dengan layanan kami jika aplikasi khusus Wilayah tetap diinstal, jadi pastikan untuk menghapus instalan aplikasi setelah Anda menonaktifkan Wilayah. Untuk informasi selengkapnya, lihat Mengaktifkan Wilayah.

Topik

- Tipe tindakan
- Parameter konfigurasi

- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Izin peran layanan: tindakan CodeConnections
- Deklarasi tindakan
- Menginstal aplikasi instalasi dan membuat koneksi
- Lihat juga

Tipe tindakan

- Kategori: `Source`
- Pemilik: `AWS`
- Penyedia: `CodeStarSourceConnection`
- Versi: `1`

Parameter konfigurasi

ConnectionArn

Wajib: Ya

Koneksi ARN yang dikonfigurasi dan diautentikasi untuk penyedia sumber.

FullRepositoryId

Wajib: Ya

Pemilik dan nama repositori tempat perubahan sumber dideteksi.

Contoh: `some-user/my-repo`

> **Important**
>
> Anda harus mempertahankan kasus yang benar untuk FullRepositoryIdnilainya. Misalnya, jika nama pengguna Anda `some-user` dan nama repo adalah`My-Repo`, nilai yang disarankan FullRepositoryIdadalah`some-user/My-Repo`.

BranchName

Wajib: Ya

Nama cabang tempat perubahan sumber harus dideteksi.

OutputArtifactFormat

Wajib: Tidak

Menentukan format artefak keluaran. Bisa salah satu `CODEBUILD_CLONE_REF` atau`CODE_ZIP`. Jika tidak ditentukan, defaultnya adalah`CODE_ZIP`.

> **Important**
>
> `CODEBUILD_CLONE_REF`Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir. Jika Anda memilih opsi ini, Anda perlu memperbarui izin untuk peran layanan CodeBuild proyek Anda seperti yang ditunjukkan pada[Tambahkan CodeBuild GitClone izin untuk koneksi ke Bitbucket,, GitHub Enterprise Server GitHub, atau .com GitLab](). Untuk tutorial yang menunjukkan cara menggunakan opsi klon Penuh, lihat[Tutorial: Gunakan klon lengkap dengan sumber GitHub pipa]().

DetectChanges

Wajib: Tidak

Kontrol secara otomatis memulai pipeline Anda ketika komit baru dibuat pada repositori dan cabang yang dikonfigurasi. Jika tidak ditentukan, nilai default adalah`true`, dan bidang tidak ditampilkan secara default. Nilai yang valid untuk parameter ini:

- `true`: CodePipeline secara otomatis memulai pipeline Anda pada komit baru.
- `false`: CodePipeline tidak memulai pipeline Anda pada komit baru.

Artefak masukan

- Jumlah artefak: `0`
- Deskripsi: Artefak masukan tidak berlaku untuk jenis tindakan ini.

Artefak keluaran

- Jumlah artefak: 1
- Deskripsi: Artefak yang dihasilkan dari repositori adalah artefak keluaran untuk tindakan tersebut. `CodeStarSourceConnection` ID komit kode sumber ditampilkan CodePipeline sebagai revisi sumber untuk eksekusi pipeline yang dipicu. Anda dapat mengonfigurasi artefak keluaran dari tindakan ini di:
  - File ZIP yang berisi isi repositori dan cabang yang dikonfigurasi pada komit yang ditentukan sebagai revisi sumber untuk eksekusi pipeline.
  - File JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung.

> **Important**
>
> Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.
> Jika Anda memilih opsi ini, Anda perlu memperbarui izin untuk peran layanan CodeBuild proyek Anda seperti yang ditunjukkan padaPemecahan masalah CodePipeline. Untuk tutorial yang menunjukkan cara menggunakan opsi klon Penuh, lihatTutorial: Gunakan klon lengkap dengan sumber GitHub pipa.

Variabel keluaran

Ketika dikonfigurasi, tindakan ini menghasilkan variabel yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir dalam pipeline. Tindakan ini menghasilkan variabel yang dapat dilihat sebagai variabel keluaran, bahkan jika tindakan tidak memiliki namespace. Anda mengonfigurasi tindakan dengan namespace untuk membuat variabel-variabel tersebut tersedia untuk konfigurasi tindakan hilir.

Untuk informasi selengkapnya, lihat Referensi variabel.

AuthorDate

Tanggal ketika komit ditulis, dalam format stempel waktu.

BranchName

Nama cabang untuk repositori tempat perubahan sumber dilakukan.

CommitId

ID komit yang memicu eksekusi pipeline.

CommitMessage

Pesan deskripsi, jika ada, terkait dengan komit yang memicu eksekusi pipeline.

ConnectionArn

Koneksi ARN yang dikonfigurasi dan diautentikasi untuk penyedia sumber.

FullRepositoryName

Nama repositori tempat komit yang memicu pipeline dibuat.

Izin peran layanan: tindakan CodeConnections

Untuk CodeConnections, izin berikut diperlukan untuk membuat pipeline dengan sumber yang menggunakan koneksi, seperti Bitbucket Cloud.

```
{
    "Effect": "Allow",
    "Action": [
        "codeconnections:UseConnection"
    ],
    "Resource": "resource_ARN"
},
```

Untuk informasi selengkapnya tentang izin IAM untuk koneksi, lihat Referensi izin koneksi.

Deklarasi tindakan

Dalam contoh berikut, artefak keluaran diatur ke format ZIP default `CODE_ZIP` untuk koneksi dengan 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"
        }
    ]
},
```

Menginstal aplikasi instalasi dan membuat koneksi

Pertama kali Anda menggunakan konsol untuk menambahkan koneksi baru ke repositori pihak ketiga, Anda harus mengotorisasi CodePipeline akses ke repositori Anda. Anda memilih atau membuat aplikasi instalasi yang membantu Anda terhubung ke akun tempat Anda membuat repositori kode pihak ketiga.

Ketika Anda menggunakan AWS CLI atau CloudFormation template, Anda harus menyediakan koneksi ARN dari koneksi yang telah melalui jabat tangan instalasi. Jika tidak, pipa tidak dipicu.

> **Note**
>
> Untuk tindakan `CodeStarSourceConnection` sumber, Anda tidak perlu menyiapkan webhook atau default ke polling. Tindakan koneksi mengelola deteksi perubahan sumber Anda untuk Anda.

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- AWS::CodeStarConnections::Connection— Referensi CloudFormation template untuk sumber daya AWS CodeStar Connections menyediakan parameter dan contoh untuk koneksi dalam CloudFormation template.
- AWS CodeStarReferensi API AWS CodeStar Koneksi — Referensi API Koneksi menyediakan informasi referensi untuk tindakan koneksi yang tersedia.
- Untuk melihat langkah-langkah untuk membuat pipeline dengan tindakan sumber yang didukung oleh koneksi, lihat berikut ini:
  - Untuk Bitbucket Cloud, gunakan opsi Bitbucket di konsol atau `CodestarSourceConnection` tindakan di CLI. Lihat Koneksi Bitbucket Cloud.
  - Untuk GitHub dan GitHub Enterprise Cloud, gunakan opsi GitHubpenyedia di konsol atau `CodestarSourceConnection` tindakan di CLI. Lihat GitHub koneksi.
  - Untuk GitHub Enterprise Server, gunakan opsi penyedia GitHub Enterprise Server di konsol atau `CodestarSourceConnection` tindakan di CLI. Lihat GitHub Koneksi Enterprise Server.
  - GitLabUntuk.com, gunakan opsi GitLabpenyedia di konsol atau `CodestarSourceConnection` tindakan dengan `GitLab` penyedia di CLI. Lihat GitLabkoneksi .com.

- Untuk melihat tutorial Memulai yang membuat pipeline dengan sumber Bitbucket dan CodeBuild tindakan, lihat Memulai koneksi.
- Untuk tutorial yang menunjukkan cara menghubungkan ke GitHub repositori dan menggunakan opsi klon Penuh dengan tindakan hilir CodeBuild , lihat. Tutorial: Gunakan klon lengkap dengan sumber GitHub pipa

Referensi tindakan perintah

Tindakan Commands memungkinkan Anda menjalankan perintah shell dalam instance komputasi virtual. Saat Anda menjalankan tindakan, perintah yang ditentukan dalam konfigurasi tindakan dijalankan dalam wadah terpisah. Semua artefak yang ditetapkan sebagai artefak masukan untuk suatu CodeBuild tindakan tersedia di dalam wadah yang menjalankan perintah. Tindakan ini memungkinkan Anda untuk menentukan perintah tanpa terlebih dahulu membuat CodeBuild proyek. Untuk informasi selengkapnya, lihat ActionDeclarationdan OutputArtifactdi Referensi AWS CodePipeline API.

Important

Tindakan ini menggunakan CodeBuild komputasi CodePipeline terkelola untuk menjalankan perintah di lingkungan build. Menjalankan tindakan perintah akan dikenakan biaya terpisah di AWS CodeBuild.

Note

Tindakan Commands hanya tersedia untuk pipeline tipe V2.

Topik

- Pertimbangan untuk tindakan Perintah
- Izin kebijakan peran layanan
- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran

- Variabel-variabel lingkungan
- Izin peran layanan: Tindakan perintah
- Deklarasi tindakan (contoh)
- Lihat juga

Pertimbangan untuk tindakan Perintah

Pertimbangan berikut berlaku untuk tindakan Perintah.

- Tindakan perintah menggunakan CodeBuild sumber daya yang mirip dengan CodeBuild tindakan, sambil mengizinkan perintah lingkungan shell dalam instance komputasi virtual tanpa perlu mengaitkan atau membuat proyek build.

  > **Note**
  >
  > Menjalankan tindakan perintah akan dikenakan biaya terpisah di AWS CodeBuild.

- Karena tindakan Command CodePipeline menggunakan CodeBuild resource, build yang dijalankan oleh action akan dikaitkan dengan batas build untuk akun Anda. CodeBuild Build yang dijalankan oleh tindakan Command akan dihitung terhadap batas build bersamaan seperti yang dikonfigurasi untuk akun tersebut.
- Batas waktu untuk build dengan tindakan Commands adalah 55 menit, berdasarkan CodeBuild build.
- Instance komputasi menggunakan lingkungan build yang terisolasi di CodeBuild.

  > **Note**
  >
  > Karena lingkungan build terisolasi digunakan di tingkat akun, instance dapat digunakan kembali untuk eksekusi pipeline lainnya.

- Semua format didukung kecuali format multi-baris. Anda harus menggunakan format satu baris saat memasukkan perintah.
- Tindakan perintah didukung untuk tindakan lintas akun. Untuk menambahkan tindakan perintah lintas akun, tambahkan `actionRoleArn` dari akun target Anda dalam deklarasi tindakan.

- Untuk tindakan ini, CodePipeline akan mengambil peran layanan pipeline dan menggunakan peran itu untuk memungkinkan akses ke sumber daya saat runtime. Disarankan untuk mengonfigurasi peran layanan sehingga izin dicakup ke tingkat tindakan.
- Izin yang ditambahkan ke peran CodePipeline layanan dirinci dalamMenambahkan izin ke peran CodePipeline layanan.
- Izin yang diperlukan untuk melihat log di konsol dirinci diIzin diperlukan untuk melihat log komputasi di konsol CodePipeline.
- Tidak seperti tindakan lain di CodePipeline, Anda tidak menyetel bidang dalam konfigurasi tindakan; Anda mengatur bidang konfigurasi tindakan di luar konfigurasi tindakan.

Izin kebijakan peran layanan

Saat CodePipeline menjalankan tindakan, CodePipeline buat grup log menggunakan nama pipeline sebagai berikut. Ini memungkinkan Anda untuk mengurangi izin untuk mencatat sumber daya menggunakan nama pipeline.

```
/aws/codepipeline/MyPipelineName
```

Jika Anda menggunakan peran layanan yang ada, untuk menggunakan tindakan Perintah, Anda perlu menambahkan izin berikut untuk peran layanan.

- log: CreateLogGroup
- log: CreateLogStream
- log: PutLogEvents

Dalam pernyataan kebijakan peran layanan, cakupkan izin ke tingkat pipeline seperti yang ditunjukkan pada contoh berikut.

```
{
    "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:*"
    ]
}
```

Untuk melihat log di konsol menggunakan halaman dialog detail tindakan, izin untuk melihat log harus ditambahkan ke peran konsol. Untuk informasi selengkapnya, lihat contoh kebijakan izin konsol diIzin diperlukan untuk melihat log komputasi di konsol CodePipeline.

Tipe tindakan

- Kategori: `Compute`
- Pemilik: `AWS`
- Penyedia: `Commands`
- Versi: `1`

Parameter konfigurasi

Commands

Wajib: Ya

Anda dapat memberikan perintah shell untuk menjalankan `Commands` tindakan. Di konsol, perintah dimasukkan pada baris terpisah. Di CLI, perintah dimasukkan sebagai string terpisah.

> **Note**
>
> Format multi-baris tidak didukung dan akan menghasilkan pesan kesalahan. Format baris tunggal harus digunakan untuk memasukkan perintah di bidang Perintah.

> **Important**
>
> ComputeType Nilai EnvironmentType dan cocok dengan yang ada di CodeBuild. Kami mendukung subset dari jenis yang tersedia. Untuk informasi selengkapnya, lihat Membangun Jenis Komputasi Lingkungan.

EnvironmentType

Wajib: Tidak

Gambar OS untuk lingkungan build yang mendukung tindakan Command. Berikut ini adalah nilai yang valid untuk lingkungan build:

- LINUX_CONTAINER
- WINDOWS_SERVER_2022_CONTAINER

Pemilihan untuk kemudian EnvironmentTypeakan memungkinkan jenis komputasi untuk OS itu di ComputeTypelapangan. Untuk informasi selengkapnya tentang jenis CodeBuild komputasi yang tersedia untuk tindakan ini, lihat referensi mode komputasi dan tipe lingkungan build di CodeBuild Panduan Pengguna.

> **Note**
>
> Jika tidak ditentukan, komputasi default menjadi berikut untuk lingkungan build:
>
> - Jenis komputasi: GENERAL1 BUILD_ _SMALL
> - Jenis lingkungan: LINUX_CONTAINER

ComputeType

Wajib: Tidak

Berdasarkan pilihan untuk EnvironmentType, jenis komputasi dapat disediakan. Berikut ini adalah nilai yang tersedia untuk komputasi; Namun, perhatikan bahwa opsi yang tersedia dapat bervariasi menurut OS.

- BANGUN_ _KECIL GENERAL1
- BANGUN_ _SEDANG GENERAL1
- BANGUN_ _BESAR GENERAL1

> **Important**
>
> Beberapa jenis komputasi tidak kompatibel dengan jenis lingkungan tertentu. Misalnya, WINDOWS_SERVER_2022_CONTAINER tidak kompatibel dengan BUILD_ _SMALL. GENERAL1 Menggunakan kombinasi yang tidak kompatibel menyebabkan tindakan gagal dan menghasilkan kesalahan runtime.

OutputVariabel

Wajib: Tidak

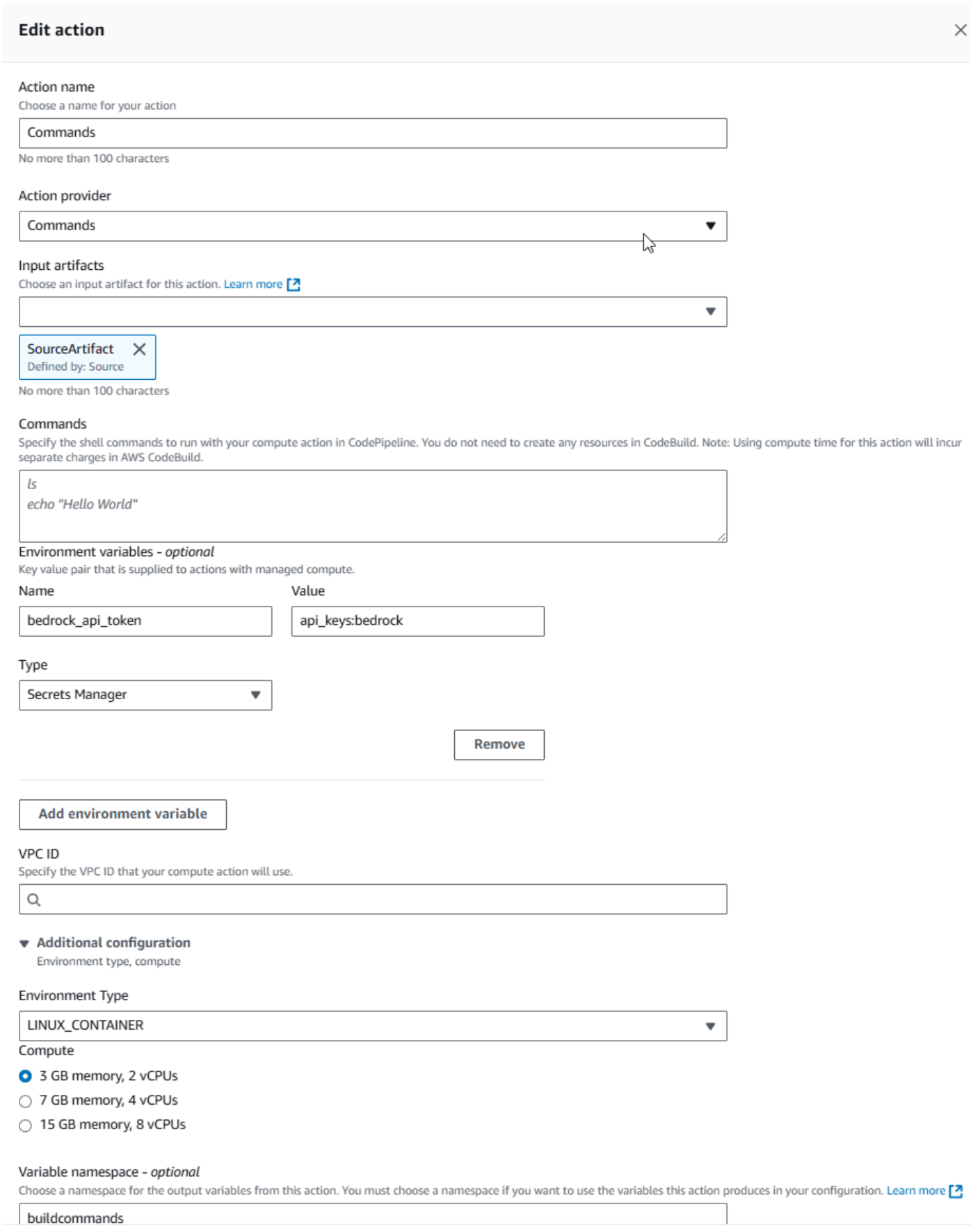
Tentukan nama variabel di lingkungan Anda yang ingin Anda ekspor. Untuk referensi variabel CodeBuild lingkungan, lihat Variabel lingkungan di lingkungan build di Panduan CodeBuild Pengguna.

Berkas

Wajib: Tidak

Anda dapat menyediakan file yang ingin Anda ekspor sebagai artefak keluaran untuk tindakan tersebut.

Format yang didukung untuk file sama dengan pola CodeBuild file. Misalnya, masukkan `**/` untuk semua file. Untuk informasi selengkapnya, lihat Membangun referensi spesifikasi untuk CodeBuild di Panduan CodeBuild Pengguna.


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

Wajib: Tidak

ID VPC untuk sumber daya Anda.

Subnet

Wajib: Tidak

Subnet untuk VPC. Bidang ini diperlukan ketika perintah Anda perlu terhubung ke sumber daya di VPC.

SecurityGroupIds

Wajib: Tidak

Kelompok keamanan untuk VPC. Bidang ini diperlukan ketika perintah Anda perlu terhubung ke sumber daya di VPC.

Berikut ini adalah contoh JSON dari tindakan dengan bidang konfigurasi yang ditampilkan untuk lingkungan dan jenis komputasi, bersama dengan variabel lingkungan contoh.

```
{
           "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
        }
```

Artefak masukan

- Jumlah artefak: `1 to 10`

Artefak keluaran

- Jumlah artefak: `0 to 1`

Variabel-variabel lingkungan

Key

Kunci dalam pasangan variabel lingkungan kunci-nilai, seperti. `BEDROCK_TOKEN`

Nilai

Nilai untuk pasangan kunci-nilai, seperti. `apiTokens:bedrockToken` Nilai dapat diparameterisasi dengan variabel keluaran dari tindakan pipa atau variabel pipa.

Saat menggunakan `SECRETS_MANAGER` tipe, nilai ini harus berupa nama rahasia yang telah Anda simpan di AWS Secrets Manager.

Tipe

Menentukan jenis penggunaan untuk nilai variabel lingkungan. Nilai dapat berupa `PLAINTEXT` atau `SECRETS_MANAGER`, salah satu. Jika nilainya`SECRETS_MANAGER`, berikan referensi Rahasia dalam `EnvironmentVariable` nilainya. Ketika tidak ditentukan, ini default ke. `PLAINTEXT`

> **Note**
>
> Kami sangat tidak menyarankan penggunaan variabel lingkungan plaintext untuk menyimpan nilai sensitif, terutama kredenal. AWS Saat Anda menggunakan CodeBuild konsol atau AWS CLI, variabel lingkungan plaintext ditampilkan dalam teks biasa. Untuk nilai sensitif, kami sarankan Anda menggunakan `SECRETS_MANAGER` tipe sebagai gantinya.

> **Note**
>
> Saat Anda memasukkan`name`,`value`, dan `type` untuk konfigurasi variabel lingkungan Anda, terutama jika variabel lingkungan berisi sintaks variabel CodePipeline keluaran, jangan melebihi batas 1000 karakter untuk bidang nilai konfigurasi. Kesalahan validasi dikembalikan ketika batas ini terlampaui.

Untuk contoh deklarasi tindakan yang menunjukkan variabel lingkungan, lihatParameter konfigurasi.

> **Note**
>
> - `SECRETS_MANAGER`Jenis ini hanya didukung untuk tindakan Command.
> - Rahasia yang direferensikan dalam tindakan Perintah akan disunting di log build yang mirip dengan. CodeBuild Tetapi pengguna pipeline yang memiliki akses Edit ke pipeline masih berpotensi mengakses nilai rahasia ini dengan memodifikasi perintah.
> - Untuk menggunakannya SecretsManager, Anda harus menambahkan izin berikut ke peran layanan pipeline Anda:
>
> ```
> {
>             "Effect": "Allow",
>             "Action": [
>                 "secretsmanager:GetSecretValue"
>             ],
>             "Resource": [
>                 "SECRET_ARN"
>             ]
>         }
> ```

Izin peran layanan: Tindakan perintah

Untuk dukungan Perintah, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

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*"
            ]
        }
    ]
}
```

Deklarasi tindakan (contoh)

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"
}
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Tutorial: Buat pipeline yang menjalankan perintah dengan komputasi (tipe V2)— Tutorial ini menyediakan contoh pipeline dengan tindakan Commands.

AWS Device Farm referensi tindakan uji

Dalam pipeline, Anda dapat mengonfigurasi tindakan pengujian yang digunakan AWS Device Farm untuk menjalankan dan menguji aplikasi di perangkat. Device Farm menggunakan kumpulan pengujian perangkat dan kerangka pengujian untuk menguji aplikasi pada perangkat tertentu. Untuk informasi tentang jenis framework pengujian yang didukung oleh tindakan Device Farm, lihat Bekerja dengan Jenis Pengujian di AWS Device Farm.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Izin peran layanan: tindakan AWS Device Farm
- Deklarasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Test`
- Pemilik: `AWS`
- Penyedia: `DeviceFarm`
- Versi: `1`

Parameter konfigurasi

AppType

Wajib: Ya

OS dan jenis aplikasi yang Anda uji. Berikut ini adalah daftar nilai yang valid:

- `iOS`
- `Android`
- `Web`

ProjectId

Wajib: Ya

ID proyek Device Farm.

Untuk menemukan ID proyek Anda, di konsol Device Farm, pilih project Anda. Di browser, salin URL proyek baru Anda. URL berisi ID proyek. ID proyek adalah nilai dalam URL setelahnya`projects/`. Dalam contoh berikut, ID proyek adalah`eec4905f-98f8-40aa-9afc-4c1cfexample`.

```
https://<region-URL>/devicefarm/home?region=us-west-2#/projects/
eec4905f-98f8-40aa-9afc-4c1cfexample/runs
```

Aplikasi

Wajib: Ya

Nama dan lokasi file aplikasi di artefak input Anda. Misalnya: `s3-ios-test-1.ipa`

TestSpec

Bersyarat: Ya

Lokasi file definisi spesifikasi pengujian di artefak input Anda. Ini diperlukan untuk uji mode kustom.

DevicePoolArn

Wajib: Ya

Perangkat Device Farm menggabungkan ARN.

Untuk mendapatkan kumpulan perangkat yang tersedia ARNs untuk proyek, termasuk ARN untuk Perangkat Teratas, gunakan AWS CLI untuk memasukkan perintah berikut:

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

TestType

Wajib: Ya

Menentukan kerangka pengujian yang didukung untuk pengujian Anda. Berikut ini adalah daftar nilai yang valid untuk`TestType`:

- APPIUM_JAVA_JUNIT
- APPIUM_JAVA_TESTNG
- APPIUM_NODE
- APPIUM_RUBY
- APPIUM_PYTHON
- APPIUM_WEB_JAVA_JUNIT
- APPIUM_WEB_JAVA_TESTNG
- APPIUM_WEB_NODE
- APPIUM_WEB_RUBY
- APPIUM_WEB_PYTHON
- BUILTIN_FUZZ
- INSTRUMENTASI
- XCTEST
- XCTEST_UI

> **Note**
>
> Jenis pengujian berikut tidak didukung oleh tindakan di CodePipeline:`WEB_PERFORMANCE_PROFILE`,`REMOTE_ACCESS_RECORD`, dan`REMOTE_ACCESS_REPLAY`.

Untuk informasi tentang jenis pengujian Device Farm, lihat Bekerja dengan Jenis Pengujian di AWS Device Farm.

RadioBluetoothEnabled

Wajib: Tidak

Nilai Boolean yang menunjukkan apakah akan mengaktifkan Bluetooth di awal pengujian.

RecordAppPerformanceData

Wajib: Tidak

Nilai Boolean yang menunjukkan apakah akan merekam data kinerja perangkat seperti CPU, FPS, dan kinerja memori selama pengujian.

RecordVideo

Wajib: Tidak

Nilai Boolean yang menunjukkan apakah akan merekam video selama pengujian.

RadioWifiEnabled

Wajib: Tidak

Nilai Boolean yang menunjukkan apakah akan mengaktifkan Wi-Fi di awal pengujian.

RadioNfcEnabled

Wajib: Tidak

Nilai Boolean yang menunjukkan apakah akan mengaktifkan NFC di awal pengujian.

RadioGpsEnabled

Wajib: Tidak

Nilai Boolean yang menunjukkan apakah akan mengaktifkan GPS di awal pengujian.

Uji

Wajib: Tidak

Nama dan jalur file definisi pengujian di lokasi sumber Anda. Jalur relatif terhadap akar artefak input untuk pengujian Anda.

FuzzEventCount

Wajib: Tidak

Jumlah peristiwa antarmuka pengguna untuk uji fuzz yang akan dilakukan, antara 1 dan 10.000.

FuzzEventThrottle

Wajib: Tidak

Jumlah milidetik untuk uji fuzz untuk menunggu sebelum melakukan acara antarmuka pengguna berikutnya, antara 1 dan 1.000.

FuzzRandomizerSeed

Wajib: Tidak

Benih untuk uji fuzz yang akan digunakan untuk mengacak peristiwa antarmuka pengguna. Menggunakan nomor yang sama untuk tes bulu halus berikutnya menghasilkan urutan peristiwa yang identik.

CustomHostMachineArtifacts

Wajib: Tidak

Lokasi pada mesin host tempat artefak kustom akan disimpan.

CustomDeviceArtifacts

Wajib: Tidak

Lokasi pada perangkat tempat artefak khusus akan disimpan.

UnmeteredDevicesOnly

Wajib: Tidak

Nilai Boolean yang menunjukkan apakah hanya akan menggunakan perangkat yang tidak diukur saat menjalankan pengujian pada langkah ini.

JobTimeoutMinutes

Wajib: Tidak

Jumlah menit uji coba akan dijalankan per perangkat sebelum waktu habis.

Lintang

Wajib: Tidak

Garis lintang perangkat dinyatakan dalam derajat sistem koordinat geografis.

Bujur

Wajib: Tidak

Bujur perangkat dinyatakan dalam derajat sistem koordinat geografis.

Artefak masukan

- Jumlah artefak: 1
- Deskripsi: Set artefak yang akan dibuat tersedia untuk tindakan uji. Device Farm mencari aplikasi yang dibangun dan definisi pengujian untuk digunakan.

Artefak keluaran

- Jumlah Artefak: 0
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Izin peran layanan: tindakan AWS Device Farm

Saat CodePipeline menjalankan tindakan, kebijakan peran CodePipeline layanan memerlukan izin berikut, yang dicakup dengan tepat ke ARN sumber daya pipeline untuk mempertahankan akses dengan hak istimewa paling sedikit. Misalnya, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

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

Deklarasi tindakan

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"
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Bekerja dengan Jenis Pengujian di Device Farm — Bab referensi dalam Panduan Pengembang Device Farm ini memberikan deskripsi selengkapnya tentang kerangka kerja pengujian Aplikasi Android, iOS, dan Web yang didukung oleh Device Farm.
- Tindakan di Device Farm — Panggilan dan parameter API di Referensi API Device Farm dapat membantu Anda bekerja dengan proyek Device Farm.
- Tutorial: Membuat pipeline yang membangun dan menguji aplikasi Android Anda AWS Device Farm— Tutorial ini menyediakan contoh file spesifikasi build dan contoh aplikasi untuk membuat pipeline dengan GitHub sumber yang membangun dan menguji aplikasi Android dengan CodeBuild dan Device Farm.
- Tutorial: Buat pipeline yang menguji aplikasi iOS Anda AWS Device Farm— Tutorial ini menyediakan contoh aplikasi untuk membuat pipeline dengan sumber Amazon S3 yang menguji aplikasi iOS yang dibangun dengan Device Farm.

Referensi aksi penyebaran Elastic Beanstalk

Elastic Beanstalk adalah AWS platform di dalamnya yang digunakan untuk menyebarkan dan menskalakan aplikasi web. Anda menggunakan tindakan Elastic Beanstalk untuk menyebarkan kode aplikasi ke lingkungan penerapan Anda.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Izin peran layanan: tindakan ElasticBeanstalk penerapan
- Deklarasi tindakan
- Lihat juga

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `ElasticBeanstalk`
- Versi: `1`

Parameter konfigurasi

ApplicationName

Wajib: Ya

Nama aplikasi yang Anda buat di Elastic Beanstalk.

EnvironmentName

Wajib: Ya

Nama lingkungan yang Anda buat di Elastic Beanstalk. Lingkungan adalah kumpulan AWS sumber daya yang menjalankan versi aplikasi. Setiap lingkungan hanya berlangsung satu versi aplikasi pada satu waktu, namun, Anda dapat menjalankan versi aplikasi yang sama atau versi aplikasi yang berbeda di banyak lingkungan secara bersamaan.

Artefak masukan

- Jumlah artefak: 1

- Deskripsi: Artefak masukan untuk aksi.

Artefak keluaran

- Jumlah artefak: `0`
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Izin peran layanan: tindakan **`ElasticBeanstalk`** penerapan

Untuk Elastic Beanstalk, berikut ini adalah izin minimum yang diperlukan untuk membuat pipeline dengan tindakan penerapan. `ElasticBeanstalk`

```
{
    "Effect": "Allow",
    "Action": [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
        "cloudformation:*",
        "rds:*",
        "sqs:*",
        "ecs:*"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Anda harus mengganti wildcard dalam kebijakan sumber daya dengan sumber daya untuk akun yang ingin Anda batasi aksesnya. Untuk informasi selengkapnya tentang membuat kebijakan yang memberikan akses hak istimewa paling sedikit, lihat. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege

Deklarasi tindakan

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"
        }
    ]
},
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Menyebarkan aplikasi Flask ke Elastic Beanstalk - Tutorial ini memandu Anda melalui pembuatan aplikasi dan sumber daya lingkungan Anda di Elastic Beanstalk menggunakan contoh aplikasi Flask. Anda kemudian dapat membangun pipeline Anda dengan tindakan penyebaran Elastic Beanstalk yang menyebarkan aplikasi Anda dari repositori sumber ke lingkungan Elastic Beanstalk Anda.

Amazon Inspector **`InspectorScan`** memanggil referensi tindakan

Amazon Inspector adalah layanan manajemen kerentanan yang secara otomatis menemukan beban kerja dan terus memindai mereka untuk kerentanan perangkat lunak dan paparan jaringan yang tidak diinginkan. `InspectorScan`Tindakan dalam CodePipeline mengotomatiskan mendeteksi dan memperbaiki kerentanan keamanan dalam kode sumber terbuka Anda. Tindakan ini adalah tindakan komputasi terkelola dengan kemampuan pemindaian keamanan. Anda dapat menggunakan InspectorScan dengan kode sumber aplikasi di repositori pihak ketiga Anda, seperti GitHub atau Bitbucket Cloud, atau dengan gambar untuk aplikasi kontainer. Tindakan Anda akan memindai dan melaporkan tingkat kerentanan dan peringatan yang Anda konfigurasikan.

> **Important**
>
> Tindakan ini menggunakan CodeBuild komputasi CodePipeline terkelola untuk menjalankan perintah di lingkungan build. Menjalankan tindakan akan dikenakan biaya terpisah. AWS CodeBuild

Topik

- ID tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Izin peran layanan: tindakan InspectorScan
- Deklarasi tindakan
- Lihat juga

ID tipe tindakan

- Kategori: `Invoke`
- Pemilik: `AWS`
- Penyedia: `InspectorScan`
- Versi: 1

Contoh:

```
{
    "Category": "Invoke",
    "Owner": "AWS",
    "Provider": "InspectorScan",
    "Version": "1"
},
```

Parameter konfigurasi

InspectorRunMode

(Wajib) String yang menunjukkan mode pemindaian. Nilai yang valid adalah `SourceCodeScan` | `ECRImageScan`.

ECRRepositoryNama

Nama repositori Amazon ECR tempat gambar didorong.

ImageTag

Tag yang digunakan untuk gambar.

Parameter untuk tindakan ini memindai tingkat kerentanan yang Anda tentukan. Level berikut untuk ambang kerentanan tersedia:

CriticalThreshold

Jumlah kerentanan tingkat keparahan kritis yang ditemukan di sumber Anda di luar yang CodePipeline seharusnya gagal dalam tindakan.

HighThreshold

Jumlah kerentanan tingkat keparahan tinggi yang ditemukan di sumber Anda di luar yang CodePipeline seharusnya gagal dalam tindakan.

MediumThreshold

Jumlah kerentanan tingkat keparahan sedang yang ditemukan di sumber Anda di luar yang CodePipeline seharusnya gagal dalam tindakan.

LowThreshold

Jumlah kerentanan tingkat keparahan rendah yang ditemukan di sumber Anda di luar yang CodePipeline seharusnya gagal dalam tindakan.

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

Artefak masukan

- Jumlah artefak: 1
- Deskripsi: Kode sumber untuk memindai kerentanan. Jika pemindaian adalah untuk repositori ECR, artefak input ini tidak diperlukan.

Artefak keluaran

- Jumlah artefak: 1
- Deskripsi: Rincian kerentanan sumber Anda dalam bentuk file Software Bill of Materials (SBOM).

Variabel keluaran

Ketika dikonfigurasi, tindakan ini menghasilkan variabel yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir dalam pipeline. Tindakan ini menghasilkan variabel yang dapat dilihat sebagai variabel keluaran, bahkan jika tindakan tidak memiliki namespace. Anda mengonfigurasi tindakan dengan namespace untuk membuat variabel-variabel tersebut tersedia untuk konfigurasi tindakan hilir.

Untuk informasi selengkapnya, lihat Referensi variabel.

HighestScannedSeverity

Output tingkat keparahan tertinggi dari pemindaian. Nilai yang valid adalah `medium | high | critical`.

Izin peran layanan: tindakan **`InspectorScan`**

Untuk dukungan `InspectorScan` tindakan, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

```
{
        "Effect": "Allow",
        "Action": "inspector-scan:ScanSbom",
        "Resource": "*"
    },
    {
        "Effect": "Allow",
```

```
        "Action": [
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
        ],
        "Resource": "resource_ARN"
    },
```

Selain itu, jika belum ditambahkan untuk tindakan Perintah, tambahkan izin berikut ke peran layanan Anda untuk melihat CloudWatch log.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "resource_ARN"
},
```

> **Note**
>
> Cakupkan izin ke tingkat sumber daya pipeline dengan menggunakan izin berbasis sumber daya dalam pernyataan kebijakan peran layanan.

Deklarasi tindakan

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"
                    },
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Untuk informasi selengkapnya tentang Amazon Inspector, lihat Panduan Pengguna Amazon Inspector.

AWS Lambda memanggil referensi tindakan

Memungkinkan Anda menjalankan fungsi Lambda sebagai tindakan di pipeline Anda. Menggunakan objek peristiwa yang merupakan input ke fungsi ini, fungsi memiliki akses ke konfigurasi tindakan, lokasi artefak input, lokasi artefak keluaran, dan informasi lain yang diperlukan untuk mengakses artefak. Untuk contoh peristiwa yang diteruskan ke fungsi pemanggilan Lambda, lihat. Contoh acara JSON Sebagai bagian dari implementasi fungsi Lambda, harus ada panggilan ke atau. `PutJobSuccessResult API` `PutJobFailureResult API` Jika tidak, eksekusi tindakan ini akan hang sampai waktu tindakan habis. Jika Anda menentukan artefak keluaran untuk tindakan, artefak tersebut harus diunggah ke bucket S3 sebagai bagian dari implementasi fungsi.

> **Important**
>
> Jangan mencatat peristiwa JSON yang CodePipeline dikirim ke Lambda karena ini dapat mengakibatkan kredensi pengguna masuk Log. CloudWatch CodePipeline Peran tersebut menggunakan acara JSON untuk meneruskan kredensi sementara ke Lambda di lapangan. `artifactCredentials` Untuk contoh acara, lihatContoh acara JSON.

Tipe tindakan

- Kategori: `Invoke`
- Pemilik: `AWS`
- Penyedia: `Lambda`
- Versi: `1`

Parameter konfigurasi

FunctionName

Wajib: Ya

`FunctionName`adalah nama fungsi yang dibuat di Lambda.

UserParameters

Wajib: Tidak

String yang dapat diproses sebagai input oleh fungsi Lambda.

Artefak masukan

- Jumlah Artefak: `0 to 5`
- Deskripsi: Kumpulan artefak yang akan tersedia untuk fungsi Lambda.

Artefak keluaran

- Jumlah Artefak: `0 to 5`
- Deskripsi: Kumpulan artefak yang dihasilkan sebagai output oleh fungsi Lambda.

Variabel keluaran

Tindakan ini akan menghasilkan sebagai variabel semua pasangan kunci-nilai yang disertakan dalam `outputVariables` bagian permintaan PutJobSuccessResult API.

Untuk informasi lebih lanjut tentang variabel di CodePipeline, lihatReferensi variabel.

Contoh konfigurasi tindakan

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"
        }
    ]
},
```

Contoh acara JSON

Tindakan Lambda mengirimkan peristiwa JSON yang berisi ID pekerjaan, konfigurasi tindakan pipeline, lokasi artefak input dan output, dan informasi enkripsi apa pun untuk artefak. Pekerja kerja mengakses detail ini untuk menyelesaikan tindakan Lambda. Untuk informasi lebih lanjut, lihat detail pekerjaan. Berikut adalah contoh kasusnya.

```
{
    "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"
            }
        }
    }
}
```

Acara JSON memberikan rincian pekerjaan berikut untuk tindakan Lambda di: CodePipeline

- `id`: ID unik yang dihasilkan sistem dari pekerjaan.
- `accountId`: ID AWS akun yang terkait dengan pekerjaan.
- `data`: Informasi lain yang diperlukan bagi pekerja kerja untuk menyelesaikan pekerjaan.
  - `actionConfiguration`: Parameter tindakan untuk tindakan Lambda. Untuk definisi, lihatParameter konfigurasi.
  - `inputArtifacts`: Artefak dipasok ke aksi.
    - `location`: Lokasi toko artefak.
      - `s3Location`: Informasi lokasi artefak masukan untuk tindakan.

        - `bucketName`: Nama penyimpanan artefak pipa untuk aksi (misalnya, ember Amazon S3 codepipeline-us-east bernama -2-1234567890).
        - `objectKey`: Nama aplikasi (misalnya,`CodePipelineDemoApplication.zip`).
      - `type`: Jenis artefak di lokasi. Saat ini, `S3` adalah satu-satunya jenis artefak yang valid.
    - `revision`: ID revisi artefak. Bergantung pada jenis objek, ini bisa berupa ID komit (GitHub) atau ID revisi (Amazon Simple Storage Service). Untuk informasi selengkapnya, lihat ArtifactRevision.
    - `name`: Nama artefak yang akan dikerjakan, sepertiMyApp.
- `outputArtifacts`: Output dari tindakan.
    - `location`: Lokasi toko artefak.
      - `s3Location`: Informasi lokasi artefak keluaran untuk tindakan tersebut.
        - `bucketName`: Nama penyimpanan artefak pipa untuk aksi (misalnya, ember Amazon S3 codepipeline-us-east bernama -2-1234567890).
        - `objectKey`: Nama aplikasi (misalnya,`CodePipelineDemoApplication.zip`).
      - `type`: Jenis artefak di lokasi. Saat ini, `S3` adalah satu-satunya jenis artefak yang valid.
    - `revision`: ID revisi artefak. Bergantung pada jenis objek, ini bisa berupa ID komit (GitHub) atau ID revisi (Amazon Simple Storage Service). Untuk informasi selengkapnya, lihat ArtifactRevision.
    - `name`: Nama output dari artefak, sepertiMyApp.
- `artifactCredentials`: Kredensi AWS sesi yang digunakan untuk mengakses artefak input dan output di bucket Amazon S3. Kredensial ini adalah kredensial sementara yang dikeluarkan oleh (). AWS Security Token Service AWS STS
    - `secretAccessKey`: Kunci akses rahasia untuk sesi tersebut.
    - `sessionToken`: Token untuk sesi.
    - `accessKeyId`: Kunci akses rahasia untuk sesi tersebut.
- `continuationToken`: Token yang dihasilkan oleh tindakan. Tindakan masa depan menggunakan token ini untuk mengidentifikasi instance tindakan yang sedang berjalan. Ketika tindakan selesai, tidak ada token kelanjutan yang harus diberikan.
- `encryptionKey`: Kunci enkripsi yang digunakan untuk mengenkripsi data di toko artefak, seperti kunci. AWS KMS Jika ini tidak ditentukan, kunci default untuk Amazon Simple Storage Service digunakan.

- `id`: ID yang digunakan untuk mengidentifikasi kunci. Untuk AWS KMS kunci, Anda dapat menggunakan ID kunci, kunci ARN, atau alias ARN.
- `type`: Jenis kunci enkripsi, seperti AWS KMS kunci.

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- AWS CloudFormation Panduan Pengguna — Untuk informasi selengkapnya tentang tindakan Lambda dan CloudFormation artefak untuk saluran pipa, lihat Menggunakan Fungsi Pengganti Parameter dengan Saluran Pipa, Mengotomatiskan Penerapan Aplikasi Berbasis Lambda, dan CodePipeline Artefak.AWS CloudFormation
- Memanggil AWS Lambda fungsi dalam pipeline di CodePipeline— Prosedur ini menyediakan contoh fungsi Lambda dan menunjukkan cara menggunakan konsol untuk membuat pipeline dengan tindakan pemanggilan Lambda.

AWS OpsWorks menyebarkan referensi tindakan

Anda menggunakan AWS OpsWorks tindakan untuk menerapkan dengan OpsWorks menggunakan pipeline Anda.

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `OpsWorks`
- Versi: `1`

Parameter konfigurasi

Aplikasi

Wajib: Ya

OpsWorks Tumpukan. Tumpukan adalah wadah untuk infrastruktur aplikasi Anda.

Tumpukan

Wajib: Ya

OpsWorks Aplikasi. Aplikasi ini mewakili kode yang ingin Anda terapkan dan jalankan.

Lapisan

Wajib: Tidak

OpsWorks Tumpukan. Lapisan menentukan konfigurasi dan sumber daya untuk satu set instance.

Artefak masukan

- Jumlah artefak: `1`
- Deskripsi: Ini adalah artefak masukan untuk tindakan Anda.

Artefak keluaran

- Jumlah artefak: `0 to 1`
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Izin peran layanan: tindakan AWS OpsWorks

Untuk AWS OpsWorks dukungan, tambahkan yang berikut ini ke pernyataan kebijakan Anda:

```
{
    "Effect": "Allow",
    "Action": [
        "opsworks:CreateDeployment",
        "opsworks:DescribeApps",
        "opsworks:DescribeCommands",
        "opsworks:DescribeDeployments",
        "opsworks:DescribeInstances",
        "opsworks:DescribeStacks",
        "opsworks:UpdateApp",
        "opsworks:UpdateStack"
    ],
    "Resource": "resource_ARN"
},
```

Contoh konfigurasi tindakan

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"
    }
}
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- AWS OpsWorks Panduan Pengguna - Untuk informasi tentang penerapan dengan AWS OpsWorks, lihat Panduan AWS OpsWorks Pengguna.

AWS Service Catalog menyebarkan referensi tindakan

Anda menggunakan AWS Service Catalog tindakan untuk menerapkan template menggunakan pipeline Anda. Ini adalah template sumber daya yang telah Anda buat di Service Catalog.

Tipe tindakan

- Kategori: `Deploy`
- Pemilik: `AWS`
- Penyedia: `ServiceCatalog`
- Versi: `1`

Parameter konfigurasi

TemplateFilePath

Wajib: Ya

Jalur file untuk template sumber daya Anda di lokasi sumber Anda.

ProductVersionName

Wajib: Ya

Versi produk di Service Catalog.

ProductType

Wajib: Ya

Jenis produk di Service Catalog.

ProductId

Wajib: Ya

ID produk di Service Catalog.

ProductVersionDescription

Wajib: Tidak

Deskripsi versi produk di Service Catalog.

Artefak masukan

- Jumlah artefak: 1
- Deskripsi: Ini adalah artefak masukan untuk tindakan Anda.

Artefak keluaran

- Jumlah artefak: 0
- Deskripsi: Artefak keluaran tidak berlaku untuk jenis tindakan ini.

Izin peran layanan: Tindakan Service Catalog

Untuk dukungan Service Catalog, tambahkan berikut ini ke pernyataan kebijakan Anda:

```
{
    "Effect": "Allow",
    "Action": [
        "servicecatalog:ListProvisioningArtifacts",
        "servicecatalog:CreateProvisioningArtifact",
        "servicecatalog:DescribeProvisioningArtifact",
        "servicecatalog:DeleteProvisioningArtifact",
        "servicecatalog:UpdateProduct"
    ],
    "Resource": "resource_ARN"
},
{
    "Effect": "Allow",
    "Action": [
        "cloudformation:ValidateTemplate"
    ],
    "Resource": "resource_ARN"
}
```

Contoh konfigurasi tindakan berdasarkan jenis file konfigurasi

Contoh berikut menunjukkan konfigurasi yang valid untuk tindakan penerapan yang menggunakan Service Catalog, untuk pipeline yang dibuat di konsol tanpa file konfigurasi terpisah:

```
"configuration": {
  "TemplateFilePath": "S3_template.json",
  "ProductVersionName": "devops S3 v2",
  "ProductType": "CLOUD_FORMATION_TEMPLATE",
  "ProductVersionDescription": "Product version description",
  "ProductId": "prod-example123456"
}
```

Contoh berikut menunjukkan konfigurasi yang valid untuk tindakan penerapan yang menggunakan Service Catalog, untuk pipeline yang dibuat di konsol dengan file `sample_config.json` konfigurasi terpisah:

```
"configuration": {
  "ConfigurationFilePath": "sample_config.json",
  "ProductId": "prod-example123456"
}
```

Contoh konfigurasi tindakan

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"
    }
}
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Panduan Pengguna Service Catalog — Untuk informasi tentang sumber daya dan templat di Service Catalog, lihat Panduan Pengguna Service Catalog.
- Tutorial: Buat pipeline yang di-deploy ke Service Catalog— Tutorial tutorial ini menunjukkan cara membuat dan mengkonfigurasi pipeline untuk menyebarkan template produk Anda ke Service Catalog dan memberikan perubahan yang telah Anda buat di repositori sumber Anda.

AWS Step Functions memanggil referensi tindakan

AWS CodePipeline Tindakan yang melakukan hal berikut:

- Memulai eksekusi mesin AWS Step Functions status dari pipeline Anda.

- Menyediakan status awal ke mesin status melalui properti dalam konfigurasi tindakan atau file yang terletak di artefak pipa untuk diteruskan sebagai input.
- Secara opsional menetapkan awalan ID eksekusi untuk mengidentifikasi eksekusi yang berasal dari tindakan.
- Mendukung mesin status Standar dan Ekspres.

> **Note**
>
> Tindakan Step Functions berjalan di Lambda, dan karena itu memiliki kuota ukuran artefak yang sama dengan kuota ukuran artefak untuk fungsi Lambda. Untuk informasi selengkapnya, lihat Kuota Lambda di Panduan Pengembang Lambda.

Tipe tindakan

- Kategori: `Invoke`
- Pemilik: `AWS`
- Penyedia: `StepFunctions`
- Versi: `1`

Parameter konfigurasi

StateMachineArn

Wajib: Ya

Nama Sumber Daya Amazon (ARN) untuk mesin status yang akan dipanggil.

ExecutionNamePrefix

Wajib: Tidak

Secara default, ID eksekusi tindakan digunakan sebagai nama eksekusi mesin negara. Jika awalan disediakan, itu ditambahkan ke ID eksekusi tindakan dengan tanda hubung dan bersama-sama digunakan sebagai nama eksekusi mesin negara.

```
myPrefix-1624a1d1-3699-43f0-8e1e-6bafd7fde791
```

Untuk mesin keadaan ekspres, nama hanya boleh berisi 0-9, A-Z, a-z, - dan _.

InputType

Wajib: Tidak

- Literal (default): Ketika ditentukan, nilai di bidang Input diteruskan langsung ke input mesin status.

  Contoh entri untuk bidang Input saat Literal dipilih:

  ```
  {"action": "test"}
  ```

- FilePath: Isi file dalam artefak input yang ditentukan oleh bidang Input digunakan sebagai input untuk eksekusi mesin negara. Artefak input diperlukan saat InputTypediatur ke FilePath.

  Contoh entri untuk bidang Input saat FilePathdipilih:

  ```
  assets/input.json
  ```

Input

Diperlukan: Kondisional

- Literal: Ketika InputTypediatur ke Literal (default), bidang ini opsional.

  Jika disediakan, bidang Input digunakan secara langsung sebagai input untuk eksekusi mesin status. Jika tidak, mesin status dipanggil dengan objek JSON kosong. `{}`

- FilePath: Kapan InputTypediatur ke FilePath, bidang ini diperlukan.

  Artefak input juga diperlukan saat InputTypediatur ke FilePath.

  Isi file dalam artefak input yang ditentukan digunakan sebagai input untuk eksekusi mesin negara.

Artefak masukan

- Jumlah artefak: `0 to 1`
- Deskripsi: Jika InputTypediatur ke FilePath, artefak ini diperlukan dan digunakan untuk sumber input untuk eksekusi mesin status.

Artefak keluaran

- Jumlah artefak: `0 to 1`
- Deskripsi:
  - Mesin Negara Standar: Jika disediakan, artefak keluaran diisi dengan output dari mesin negara. Ini diperoleh dari `output` properti respons Step Functions DescribeExecution API setelah eksekusi mesin status berhasil diselesaikan.
  - Mesin Negara Ekspres: Tidak didukung.

Variabel keluaran

Tindakan ini menghasilkan variabel keluaran yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir dalam pipeline.

Untuk informasi selengkapnya, lihat Referensi variabel.

StateMachineArn

ARN dari mesin negara.

ExecutionArn

ARN dari eksekusi mesin negara. Hanya mesin negara standar.

Izin peran layanan: tindakan **`StepFunctions`**

Untuk `StepFunctions` tindakan, berikut ini adalah izin minimum yang diperlukan untuk membuat pipeline dengan tindakan pemanggilan Step Functions.

```
{
    "Effect": "Allow",
    "Action": [
        "states:DescribeStateMachine",
        "states:DescribeExecution",
        "states:StartExecution"
    ],
    "Resource": "resource_ARN"
},
```

Contoh konfigurasi tindakan

Contoh untuk input default

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"
    }
}
```

Contoh untuk masukan literal

YAML

```
Name: ActionName
ActionTypeId:
  Category: Invoke
  Owner: AWS
  Version: 1
  Provider: StepFunctions
OutputArtifacts:
  - Name: myOutputArtifact
Configuration:
  StateMachineArn: arn:aws:states:us-east-1:111122223333:stateMachine:HelloWorld-
StateMachine
  ExecutionNamePrefix: my-prefix
  Input: '{"action": "test"}'
```

JSON

```
{
    "Name": "ActionName",
    "ActionTypeId": {
        "Category": "Invoke",
        "Owner": "AWS",
        "Version": 1,
        "Provider": "StepFunctions"
    },
    "OutputArtifacts": [
        {
            "Name": "myOutputArtifact"
        }
    ],
    "Configuration": {
        "StateMachineArn": "arn:aws:states:us-
east-1:111122223333:stateMachine:HelloWorld-StateMachine",
        "ExecutionNamePrefix": "my-prefix",
        "Input": "{\"action\": \"test\"}"
    }
}
```

Contoh untuk file input

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"
    }
}
```

Perilaku

Selama rilis, CodePipeline jalankan mesin status yang dikonfigurasi menggunakan input seperti yang ditentukan dalam konfigurasi tindakan.

Kapan InputTypediatur ke Literal, konten bidang konfigurasi tindakan Input digunakan sebagai input untuk mesin status. Ketika input literal tidak disediakan, eksekusi mesin negara menggunakan objek `{}` JSON kosong. Untuk informasi selengkapnya tentang menjalankan eksekusi mesin status tanpa input, lihat Step Functions StartExecution API.

Ketika InputTypediatur ke FilePath, tindakan membuka ritsleting artefak input dan menggunakan konten file yang ditentukan dalam bidang konfigurasi tindakan Input sebagai input untuk mesin status. Kapan FilePathditentukan, bidang Input diperlukan dan artefak input harus ada; jika tidak, tindakan gagal.

Setelah eksekusi awal berhasil, perilaku akan menyimpang untuk dua jenis mesin status, standar dan ekspres.

Mesin negara standar

Jika eksekusi mesin status standar berhasil dimulai, CodePipeline polling `DescribeExecution` API hingga eksekusi mencapai status terminal. Jika eksekusi berhasil diselesaikan, tindakan berhasil; jika tidak, itu gagal.

Jika artefak keluaran dikonfigurasi, artefak akan berisi nilai pengembalian mesin status. Ini diperoleh dari `output` properti respons Step Functions DescribeExecution API setelah eksekusi mesin status berhasil diselesaikan. Perhatikan bahwa ada batasan panjang keluaran yang diberlakukan pada API ini.

Penanganan kesalahan

- Jika tindakan gagal memulai eksekusi mesin negara, eksekusi tindakan gagal.

- Jika eksekusi mesin status gagal mencapai status terminal sebelum tindakan CodePipeline Step Functions mencapai batas waktu (default 7 hari), eksekusi tindakan gagal. Mesin negara mungkin terus berlanjut meskipun kegagalan ini. Untuk informasi selengkapnya tentang batas waktu eksekusi mesin status di Step Functions, lihat Alur Kerja Standar vs Ekspres.

  > **Note**
  >
  > Anda dapat meminta peningkatan kuota untuk batas waktu tindakan pemanggilan untuk akun dengan tindakan tersebut. Namun, peningkatan kuota berlaku untuk semua tindakan jenis ini di semua Wilayah untuk akun tersebut.

- Jika eksekusi mesin status mencapai status terminal FAILED, TIMED_OUT, atau ABORTED, eksekusi tindakan gagal.

Mesin negara ekspres

Jika eksekusi mesin keadaan ekspres berhasil dimulai, eksekusi aksi pemanggilan selesai dengan sukses.

Pertimbangan untuk tindakan yang dikonfigurasi untuk mesin keadaan ekspres:

- Anda tidak dapat menunjuk artefak keluaran.
- Tindakan tidak menunggu eksekusi mesin negara selesai.
- Setelah eksekusi tindakan dimulai CodePipeline, eksekusi tindakan berhasil bahkan jika eksekusi mesin status gagal.

Penanganan kesalahan

- Jika CodePipeline gagal memulai eksekusi mesin status, eksekusi tindakan gagal. Kalau tidak, tindakan segera berhasil. Tindakan berhasil CodePipeline terlepas dari berapa lama eksekusi mesin negara untuk menyelesaikan atau hasilnya.

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- AWS Step Functions Panduan Pengembang — Untuk informasi tentang mesin status, eksekusi, dan input untuk mesin status, lihat Panduan AWS Step Functions Pengembang.
- Tutorial: Gunakan AWS Step Functions tindakan pemanggilan dalam pipeline— Tutorial ini membantu Anda memulai dengan contoh mesin status standar dan menunjukkan cara menggunakan konsol untuk memperbarui pipeline dengan menambahkan tindakan pemanggilan Step Functions.

Referensi struktur aturan

Bagian ini adalah referensi untuk konfigurasi aturan saja. Untuk gambaran konseptual dari struktur pipa, lihatCodePipeline referensi struktur pipa.

Setiap penyedia aturan CodePipeline menggunakan satu set bidang konfigurasi wajib dan opsional dalam struktur pipa. Bagian ini memberikan informasi referensi berikut oleh penyedia aturan:

- Nilai yang valid untuk `RuleType` bidang yang termasuk dalam blok aturan struktur pipa, seperti `Owner` dan`Provider`.
- Deskripsi dan informasi referensi lainnya untuk `Configuration` parameter (wajib dan opsional) termasuk dalam bagian aturan struktur pipa.
- Contoh bidang konfigurasi aturan JSON dan YAMAL yang valid.

Informasi referensi tersedia untuk penyedia aturan berikut:

Topik

- CloudWatchAlarm
- CodeBuild aturan
- Commands
- DeploymentWindow
- LambdaInvoke
- VariableCheck

CloudWatchAlarm

Saat Anda membuat kondisi, Anda dapat menambahkan `CloudWatchAlarm` aturan. Bagian ini memberikan referensi untuk parameter aturan. Untuk informasi selengkapnya tentang aturan dan ketentuan, lihatBagaimana kondisi panggung bekerja?.

Anda harus sudah membuat alarm di Amazon CloudWatch sebagai sumber daya terpisah.

Topik

- Jenis aturan
- Parameter konfigurasi

- Contoh konfigurasi aturan
- Lihat juga

Jenis aturan

- Kategori: `Rule`
- Pemilik: `AWS`
- Penyedia: `CloudWatchAlarm`
- Versi: `1`

Parameter konfigurasi

AlarmName

Wajib: Ya

Nama CloudWatch alarm. Ini adalah sumber daya terpisah yang dibuat di CloudWatch.

AlarmStates

Wajib: Tidak

CloudWatch Alarm yang diinginkan menyatakan aturan untuk dipantau. Nilai yang valid adalah ALARM, OK, dan INSUFFICIENT_DATA.

WaitTime

Wajib: Tidak

Waktu tunggu dalam hitungan menit untuk memungkinkan perubahan status sebelum menjalankan hasil aturan. Misalnya, konfigurasikan 20 menit untuk menunggu status alarm berubah menjadi OK sebelum menerapkan hasil aturan.

Contoh konfigurasi aturan

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"
                        }
                    ]
                }
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan aturan ini.

- Menciptakan kondisi Kegagalan— Bagian ini memberikan langkah-langkah untuk membuat kondisi On Failure dengan aturan alarm.

CodeBuild aturan

Saat Anda membuat kondisi, Anda dapat menambahkan CodeBuild aturan. Bagian ini memberikan referensi untuk parameter aturan. Untuk informasi selengkapnya tentang aturan dan ketentuan, lihatBagaimana kondisi panggung bekerja?.

Anda dapat menggunakan CodeBuild aturan untuk membuat kondisi di mana keberhasilan menjalankan proyek build Anda memenuhi kriteria aturan, seperti build run yang berhasil untuk kondisi beforeEntry.

> **Note**
>
> Untuk kondisi beforeEntry yang dikonfigurasi dengan hasil Lewati, hanya aturan berikut yang tersedia: `LambdaInvoke` dan. `VariableCheck`

Topik

- Izin kebijakan peran layanan
- Jenis aturan
- Parameter konfigurasi
- Contoh konfigurasi aturan
- Lihat juga

Izin kebijakan peran layanan

Untuk izin aturan ini, tambahkan berikut ini ke pernyataan kebijakan peran CodePipeline layanan Anda. Cakupan izin ke tingkat sumber daya.

```
{
    "Effect": "Allow",
    "Action": [
        "codebuild:BatchGetBuilds",
        "codebuild:StartBuild"
    ],
    "Resource": "resource_ARN"
},
```

Jenis aturan

- Kategori: `Rule`
- Pemilik: `AWS`
- Penyedia: `CodeBuild`
- Versi: `1`

Parameter konfigurasi

ProjectName

Wajib: Ya

`ProjectName`adalah nama proyek pembangunan di CodeBuild.

PrimarySource

Diperlukan: Kondisional

Nilai `PrimarySource` parameter harus menjadi nama salah satu artefak input untuk tindakan. CodeBuild mencari file buildspec dan menjalankan perintah buildspec di direktori yang berisi versi unzip artefak ini.

Parameter ini diperlukan jika beberapa artefak masukan ditentukan untuk suatu CodeBuild tindakan. Ketika hanya ada satu artefak sumber untuk aksi, artefak default ke `PrimarySource` artefak itu.

BatchEnabled

Wajib: Tidak

Nilai Boolean dari `BatchEnabled` parameter memungkinkan tindakan untuk menjalankan beberapa build dalam eksekusi build yang sama.

Ketika opsi ini diaktifkan, `CombineArtifacts` opsi tersedia.

Untuk contoh pipeline dengan build batch diaktifkan, lihat CodePipeline integrasi dengan CodeBuild dan build batch.

CombineArtifacts

Wajib: Tidak

Nilai Boolean `CombineArtifacts` parameter menggabungkan semua artefak build dari build batch menjadi file artefak tunggal untuk aksi build.

Untuk menggunakan opsi ini, `BatchEnabled` parameter harus diaktifkan.

EnvironmentVariables

Wajib: Tidak

Nilai parameter ini digunakan untuk mengatur variabel lingkungan untuk CodeBuild tindakan di pipeline Anda. Nilai untuk `EnvironmentVariables` parameter mengambil bentuk array JSON dari objek variabel lingkungan. Lihat parameter contoh diDeklarasi tindakan (CodeBuild contoh).

Setiap objek memiliki tiga bagian, yang semuanya adalah string:

- `name`: Nama atau kunci variabel lingkungan.
- `value`: Nilai variabel lingkungan. Saat menggunakan `SECRETS_MANAGER` tipe `PARAMETER_STORE` atau, nilai ini harus berupa nama parameter yang telah Anda simpan di AWS Systems Manager Parameter Store atau rahasia yang telah Anda simpan di AWS Secrets Manager, masing-masing.

  > **Note**
  >
  > Kami sangat tidak menyarankan penggunaan variabel lingkungan untuk menyimpan nilai sensitif, terutama AWS kredensyal. Saat Anda menggunakan CodeBuild konsol atau AWS CLI, variabel lingkungan ditampilkan dalam teks biasa. Untuk nilai sensitif, kami sarankan Anda menggunakan `SECRETS_MANAGER` tipe sebagai gantinya.

- `type`: (Opsional) Jenis variabel lingkungan. Nilai yang valid adalah `PARAMETER_STORE`, `SECRETS_MANAGER`, atau `PLAINTEXT`. Ketika tidak ditentukan, ini default ke. `PLAINTEXT`

> **Note**
>
> Saat Anda memasukkan`name`,`value`, dan `type` untuk konfigurasi variabel lingkungan Anda, terutama jika variabel lingkungan berisi sintaks variabel CodePipeline keluaran, jangan melebihi batas 1000 karakter untuk bidang nilai konfigurasi. Kesalahan validasi dikembalikan ketika batas ini terlampaui.

Untuk informasi selengkapnya, lihat EnvironmentVariabledi Referensi AWS CodeBuild API. Untuk contoh CodeBuild tindakan dengan variabel lingkungan yang menyelesaikan ke nama GitHub cabang, lihat. Contoh: Gunakan BranchName variabel dengan variabel CodeBuild lingkungan

Contoh konfigurasi aturan

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"
```

```
}
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan aturan ini.

- Untuk informasi selengkapnya tentang aturan dan ketentuan, lihat Kondisi RuleTypeId, dan RuleExecutiondi Panduan CodePipeline API.

Commands

Saat Anda membuat kondisi, Anda dapat menambahkan `Commands` aturan. Bagian ini memberikan referensi untuk parameter aturan. Untuk informasi selengkapnya tentang aturan dan ketentuan, lihatBagaimana kondisi panggung bekerja?.

Anda dapat menggunakan `Commands` aturan untuk membuat kondisi di mana perintah berhasil memenuhi kriteria aturan, seperti output dan jalur file untuk perintah yang berhasil untuk kondisi beforeEntry.

> **Note**
>
> Untuk kondisi beforeEntry yang dikonfigurasi dengan hasil Lewati, hanya aturan berikut yang tersedia: `LambdaInvoke` dan. `VariableCheck`

Topik

- Pertimbangan untuk aturan Perintah
- Izin kebijakan peran layanan
- Jenis aturan
- Parameter konfigurasi
- Contoh konfigurasi aturan
- Lihat juga

Pertimbangan untuk aturan Perintah

Pertimbangan berikut berlaku untuk aturan Perintah.

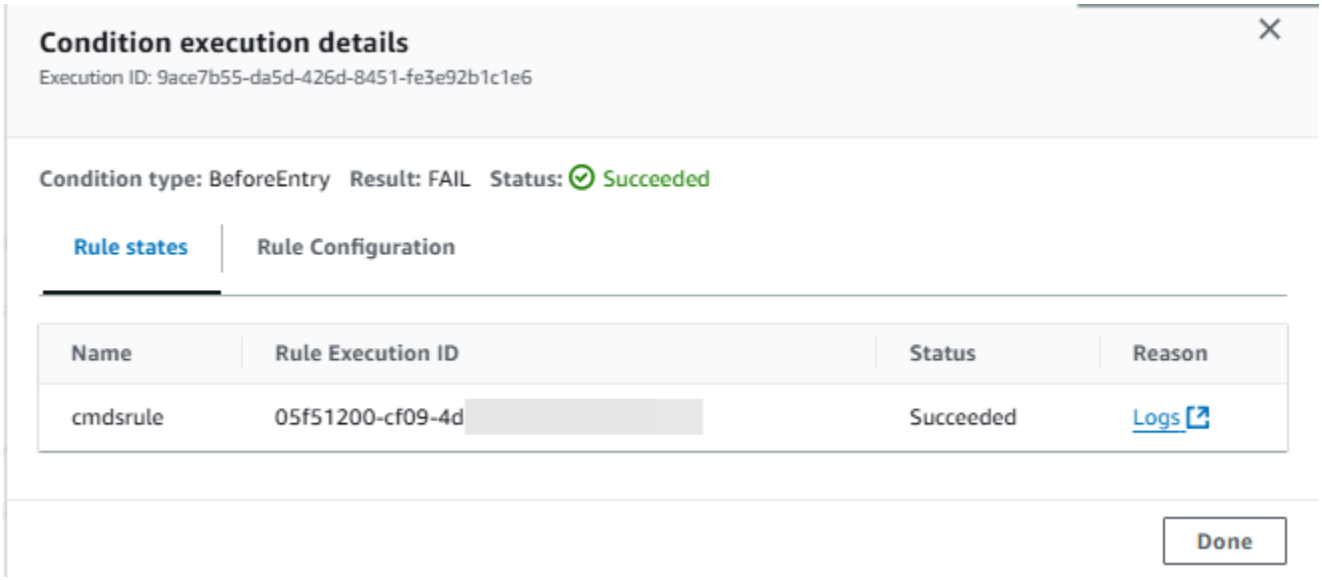
- Aturan perintah menggunakan CodeBuild sumber daya yang mirip dengan CodeBuild tindakan, sementara mengizinkan perintah lingkungan shell dalam instance komputasi virtual tanpa perlu mengaitkan atau membuat proyek build.

> **Note**
>
> Menjalankan aturan perintah akan dikenakan biaya terpisah di AWS CodeBuild.

- Karena aturan Command CodePipeline menggunakan CodeBuild resource, build yang dijalankan oleh action akan dikaitkan dengan batas build untuk akun Anda. CodeBuild Build yang dijalankan oleh aturan Command akan dihitung terhadap batas build bersamaan seperti yang dikonfigurasi untuk akun tersebut.
- Batas waktu untuk build dengan aturan Command adalah 55 menit, berdasarkan CodeBuild build.
- Instance komputasi menggunakan lingkungan build yang terisolasi di CodeBuild.

> **Note**
>
> Karena lingkungan build terisolasi digunakan di tingkat akun, instance dapat digunakan kembali untuk eksekusi pipeline lainnya.

- Semua format didukung kecuali format multi-baris. Anda harus menggunakan format satu baris saat memasukkan perintah.
- Untuk aturan ini, CodePipeline akan mengambil peran layanan pipeline dan menggunakan peran itu untuk memungkinkan akses ke sumber daya saat runtime. Disarankan untuk mengonfigurasi peran layanan sehingga izin dicakup ke tingkat tindakan.
- Izin yang ditambahkan ke peran CodePipeline layanan dirinci dalam[Menambahkan izin ke peran CodePipeline layanan](#).
- Izin yang diperlukan untuk melihat log di konsol dirinci di[Izin diperlukan untuk melihat log komputasi di konsol CodePipeline](#). Di layar contoh berikut, gunakan tautan Log untuk melihat log agar aturan Command berhasil di CloudWatch log.



- Tidak seperti tindakan lain di CodePipeline, Anda tidak menyetel bidang dalam konfigurasi tindakan; Anda mengatur bidang konfigurasi tindakan di luar konfigurasi tindakan.

Izin kebijakan peran layanan

Saat CodePipeline menjalankan aturan, CodePipeline buat grup log menggunakan nama pipeline sebagai berikut. Ini memungkinkan Anda untuk mengurangi izin untuk mencatat sumber daya menggunakan nama pipeline.

```
/aws/codepipeline/MyPipelineName
```

Jika Anda menggunakan peran layanan yang ada, untuk menggunakan tindakan Perintah, Anda perlu menambahkan izin berikut untuk peran layanan.

- log: CreateLogGroup

- log: CreateLogStream
- log: PutLogEvents

Dalam pernyataan kebijakan peran layanan, cakupkan izin ke tingkat pipeline seperti yang ditunjukkan pada contoh berikut.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "arn:aws:logs:*:YOUR_AWS_ACCOUNT_ID:log-group:/aws/
codepipeline/YOUR_PIPELINE_NAME:*"
}
```

Untuk melihat log di konsol menggunakan halaman dialog detail tindakan, izin untuk melihat log harus ditambahkan ke peran konsol. Untuk informasi selengkapnya, lihat contoh kebijakan izin konsol diIzin diperlukan untuk melihat log komputasi di konsol CodePipeline.

Jenis aturan

- Kategori: `Rule`
- Pemilik: `AWS`
- Penyedia: `Commands`
- Versi: `1`

Parameter konfigurasi

Commands

Wajib: Ya

Anda dapat memberikan perintah shell untuk menjalankan `Commands` aturan. Di konsol, perintah dimasukkan pada baris terpisah. Di CLI, perintah dimasukkan sebagai string terpisah.

> **Note**
>
> Format multi-baris tidak didukung dan akan menghasilkan pesan kesalahan. Format baris tunggal harus digunakan untuk memasukkan perintah di bidang Perintah.

Rincian berikut memberikan komputasi default yang digunakan untuk aturan Command. Untuk informasi selengkapnya, lihat Mode komputasi lingkungan build dan referensi tipe di Panduan CodeBuild Pengguna.

- CodeBuild gambar: aws/codebuild/amazonlinux 2-x86_64-standar: 5.0
- Jenis komputasi: Linux Kecil
- Nilai ComputeType Lingkungan: BUILD_ _SMALL GENERAL1
- Nilai tipe lingkungan: LINUX_CONTAINER

Contoh konfigurasi aturan

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"
        }
    ]
}
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan aturan ini.

- Untuk informasi selengkapnya tentang aturan dan ketentuan, lihat Kondisi RuleTypeId, dan RuleExecutiondi Panduan CodePipeline API.

DeploymentWindow

Saat Anda membuat kondisi, Anda dapat menambahkan `DeploymentWindow` aturan. Bagian ini memberikan referensi untuk parameter aturan. Untuk informasi selengkapnya tentang aturan dan ketentuan, lihatBagaimana kondisi panggung bekerja?.

Topik

- Jenis aturan
- Parameter konfigurasi

- Contoh konfigurasi aturan
- Lihat juga

Jenis aturan

- Kategori: `Rule`
- Pemilik: `AWS`
- Penyedia: `DeploymentWindow`
- Versi: `1`

Parameter konfigurasi

Cron

Wajib: Ya

Ekspresi yang mendefinisikan hari dan waktu ketika penerapan akan diizinkan. Ekspresi cron terdiri dari 6 bidang wajib dan satu bidang opsional dipisahkan oleh spasi putih. Bidang ekspresi cron memungkinkan Anda menentukan pola jadwal dengan ekspresi cron sebagai berikut.

Nama bidang	Nilai yang diizinkan	Karakter khusus yang diizinkan
Detik	N/A	*
Menit	0-59	, - * /
Jam	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
Bulan	1-12 atau JAN-DES	, - * /
Hari-minggu	1-7 atau MGG-SBT	, - *? /L #
Tahun (Opsional)	kosong, 1970-2199	, - * /

- Karakter '*' digunakan untuk menentukan semua nilai. Misalnya, “*” di bidang menit berarti “setiap menit”.
- Yang '?' karakter diperbolehkan untuk day-of-month dan day-of-week bidang. Ini digunakan untuk menentukan 'tidak ada nilai spesifik'. Ini berguna ketika Anda perlu menentukan sesuatu di salah satu dari dua bidang, tetapi tidak yang lain.
- Karakter '-' digunakan untuk menentukan rentang Misalnya “10-12" di bidang jam berarti “jam 10, 11 dan 12".
- Karakter ',' digunakan untuk menentukan nilai tambahan. Misalnya “MON, WED, FRI” di day-of-week lapangan berarti “hari-hari Senin, Rabu, dan Jumat”.
- Karakter '/' digunakan untuk menentukan kenaikan. Misalnya “0/15" di bidang detik berarti “detik 0, 15, 30, dan 45". Dan “5/15" di bidang detik berarti “detik 5, 20, 35, dan 50". Menentukan '*' sebelum '/' setara dengan menentukan 0 adalah nilai untuk memulai.
- Karakter 'L' diperbolehkan untuk day-of-week bidang day-of-month dan. Karakter ini pendek untuk “terakhir”, tetapi memiliki arti yang berbeda di masing-masing dari dua bidang. Misalnya, nilai “L” di day-of-month lapangan berarti “hari terakhir bulan” - hari 31 untuk Januari, hari 28 untuk Februari pada tahun non-kabisat. Jika digunakan di day-of-week lapangan dengan sendirinya, itu berarti “7" atau “SAT”. Tetapi jika digunakan di day-of-week bidang setelah nilai lain, itu berarti “<specified_day>hari terakhir bulan” - misalnya “6L” berarti “Jumat terakhir setiap bulan”. Anda juga dapat menentukan offset dari hari terakhir bulan itu, seperti “L-3" yang berarti third-to-last hari bulan kalender.
- Karakter 'W' diperbolehkan untuk day-of-month bidang tersebut. Karakter ini digunakan untuk menentukan hari kerja (Senin-Jumat) terdekat hari yang diberikan. Sebagai contoh, jika Anda menentukan “15W” sebagai nilai untuk day-of-month bidang, artinya adalah: “hari kerja terdekat dengan tanggal 15 bulan”. Jadi jika tanggal 15 adalah hari Sabtu, pemicunya akan menyala pada hari Jumat tanggal 14. Jika tanggal 15 adalah hari Minggu, pemicunya akan menyala pada hari Senin tanggal 16. Jika tanggal 15 adalah hari Selasa, maka akan menyala pada hari Selasa tanggal 15.
- Karakter 'L' dan 'W' juga dapat digabungkan untuk day-of-month ekspresi menghasilkan 'LW', yang diterjemahkan menjadi “hari kerja terakhir bulan ini”.
- Karakter '#' diperbolehkan untuk day-of-week bidang. Karakter ini digunakan untuk menentukan <specified_day>hari “ke-n” dalam sebulan. Misalnya, nilai “6 #3" di day-of-week lapangan berarti Jumat ketiga setiap bulan (hari 6 = Jumat dan "#3" = yang ke-3 dalam sebulan).
- Karakter hukum dan nama bulan dan hari dalam seminggu tidak peka huruf besar/kecil.

TimeZone

Wajib: Tidak

Zona waktu untuk jendela penyebaran. Ekspresi reguler cocok dengan pola dalam format berikut:

- Format Wilayah/Kota. Nilai cocok dengan zona waktu dalam format Region/City atau Region/City_City. Misalnya, `America/New_York` atau `Europe/Berlin`.
- Format UTC. Nilai cocok dengan string UTC opsional diikuti oleh offset dalam format+HH: MM atau -HH: MM. Sebagai contoh,,, atau. `UTC` `UTC+05:30` `UTC-03:00` Ini adalah format default jika parameter tidak diatur sebaliknya.
- Format singkatan. Nilai cocok dengan singkatan 3 hingga 5 karakter untuk zona waktu. Misalnya, `EST` atau `IST`.

  Untuk tabel nilai TimeZone ID yang valid, lihat https://docs.oracle.com/middleware/1221/ .html wcs/tag-ref/MISC/TimeZones. Perhatikan bahwa singkatan tertentu adalah singkatan duplikat, seperti CST untuk Waktu Standar Tengah, Waktu Standar Tiongkok, dan Waktu Standar Kuba.

Contoh konfigurasi aturan

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"
            }
        ]
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan aturan ini.

- Menciptakan kondisi Sukses— Bagian ini menyediakan langkah-langkah untuk membuat kondisi On Success dengan aturan jendela penerapan.
- Untuk informasi selengkapnya tentang aturan dan ketentuan, lihat Kondisi RuleTypeId, dan RuleExecutiondi Panduan CodePipeline API.

LambdaInvoke

Saat Anda membuat kondisi, Anda dapat menambahkan `LambdaInvoke` aturan. Bagian ini memberikan referensi untuk parameter aturan. Untuk informasi selengkapnya tentang aturan dan ketentuan, lihatBagaimana kondisi panggung bekerja?.

Anda harus sudah membuat fungsi di Lambda sebagai sumber daya terpisah.

Topik

- Jenis aturan
- Parameter konfigurasi
- Contoh konfigurasi aturan
- Lihat juga

Jenis aturan

- Kategori: `Rule`
- Pemilik: `AWS`
- Penyedia: `LambdaInvoke`
- Versi: 1

Parameter konfigurasi

FunctionName

Wajib: Ya

Nama fungsi Lambda.

UserParameters

Wajib: Tidak

Ini adalah parameter yang disediakan sebagai input untuk fungsi dalam format pasangan nilai kunci.

Contoh konfigurasi aturan

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"
    }
]
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan aturan ini.

- Menciptakan kondisi Kegagalan— Bagian ini memberikan langkah-langkah untuk membuat kondisi On Failure dengan aturan alarm.

VariableCheck

Saat Anda membuat kondisi, Anda dapat menambahkan `VariableCheck` aturan. Bagian ini memberikan referensi untuk parameter aturan. Untuk informasi selengkapnya tentang aturan dan ketentuan, lihatBagaimana kondisi panggung bekerja?.

Anda dapat menggunakan `VariableCheck` aturan untuk membuat kondisi di mana variabel output diperiksa terhadap ekspresi yang disediakan. Aturan melewati pemeriksaan ketika nilai variabel memenuhi kriteria aturan, seperti nilai yang sama atau lebih besar dari variabel output tertentu.

Topik

- Jenis aturan
- Parameter konfigurasi
- Contoh konfigurasi aturan
- Lihat juga

Jenis aturan

- Kategori: `Rule`
- Pemilik: `AWS`
- Penyedia: `VariableCheck`
- Versi: `1`

Parameter konfigurasi

Operator

Wajib: Ya

Operator yang menunjukkan operasi mana yang harus dilakukan untuk pemeriksaan variabel.

Dalam contoh berikut, variabel output untuk nama repositori akan diperiksa apakah sama dengan. `MyDemoRepo`

```
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
```

Operator berikut tersedia untuk membuat ekspresi sebagai berikut.

- Sama dengan - Pilih operator ini untuk memeriksa apakah variabel sama dengan nilai string.

  Parameter CLI: `EQ`
- Berisi - Pilih operator ini untuk memeriksa apakah variabel berisi nilai string sebagai substring.

  Parameter CLI: `CONTAINS`

- Cocokkan - Pilih operator ini untuk memeriksa apakah variabel cocok dengan ekspresi regex yang diberikan sebagai nilai string.

  Semua ekspresi reguler CodePipeline sesuai dengan sintaks regex Java. Untuk deskripsi komprehensif tentang sintaks regex Java dan konstruksinya, lihat java.util.Regex.Pattern.

  Parameter CLI: `MATCHES`
- Tidak sama - Pilih operator ini untuk memeriksa apakah variabel tidak sama dengan nilai string.

  Parameter CLI: `NE`

Variabel

Wajib: Ya

Variabel pipeline untuk diperiksa.

Nilai

Wajib: Ya

Nilai untuk ekspresi untuk memeriksa terhadap.

Dalam contoh berikut, variabel output untuk nama repositori akan diperiksa apakah sama dengan. `MyDemoRepo`

```
        "configuration": {
            "Variable": "#{SourceVariables.RepositoryName}",
            "Value": "MyDemoRepo",
            "Operator": "EQ"
        },
```

Dalam contoh JSON berikut, dua aturan terpisah didefinisikan, satu untuk pernyataan `EQ` (sama) yang memeriksa repositori dan nama cabang diformat sebagai # {. SourceVariables RepositoryName} dan satu untuk `CONTAINS` itu memeriksa variabel keluaran pesan komit yang diformat sebagai # {SourceVariables. CommitMessage} terhadap nilai yang diberikan “pembaruan.”

```
"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"
                            }
                        ]
                    }
                ]
            }
        }
    ],
```

Contoh konfigurasi aturan

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"
    }
]
```

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan aturan ini.

- Tutorial: Buat aturan pemeriksaan variabel untuk pipeline sebagai kondisi entri- Bagian ini menyediakan tutorial dengan langkah-langkah untuk membuat kondisi On Entry dengan aturan pemeriksaan variabel.
- Referensi variabel— Bagian ini memberikan informasi referensi dan contoh untuk variabel pipeline.
- Untuk informasi selengkapnya tentang aturan dan ketentuan, lihat Kondisi RuleTypeId, dan RuleExecutiondi Panduan CodePipeline API.

Referensi model integrasi

Ada beberapa integrasi pra-bangun untuk layanan pihak ketiga untuk membantu membangun alat pelanggan yang ada ke dalam proses rilis pipeline. Mitra, atau penyedia layanan pihak ketiga, menggunakan model integrasi untuk mengimplementasikan jenis tindakan untuk digunakan CodePipeline.

Gunakan referensi ini saat Anda merencanakan atau bekerja dengan tipe tindakan yang dikelola dengan model integrasi yang didukung di CodePipeline.

Untuk mengesahkan jenis tindakan pihak ketiga Anda sebagai integrasi mitra dengan CodePipeline, rujuk Jaringan AWS Mitra (APN). Informasi ini adalah suplemen untuk AWS CLI Referensi.

Topik

- Cara kerja tipe tindakan pihak ketiga dengan integrator
- Konsep
- Model integrasi yang didukung
- Model integrasi Lambda
- Model integrasi pekerja Job

Cara kerja tipe tindakan pihak ketiga dengan integrator

Anda dapat menambahkan jenis tindakan pihak ketiga ke pipeline pelanggan untuk menyelesaikan tugas pada sumber daya pelanggan. Integrator mengelola permintaan pekerjaan dan menjalankan tindakan dengan CodePipeline. Diagram berikut menunjukkan jenis tindakan pihak ketiga yang dibuat untuk digunakan pelanggan dalam pipeline mereka. Setelah pelanggan mengonfigurasi tindakan, tindakan berjalan dan membuat permintaan pekerjaan yang ditangani oleh mesin tindakan integrator.

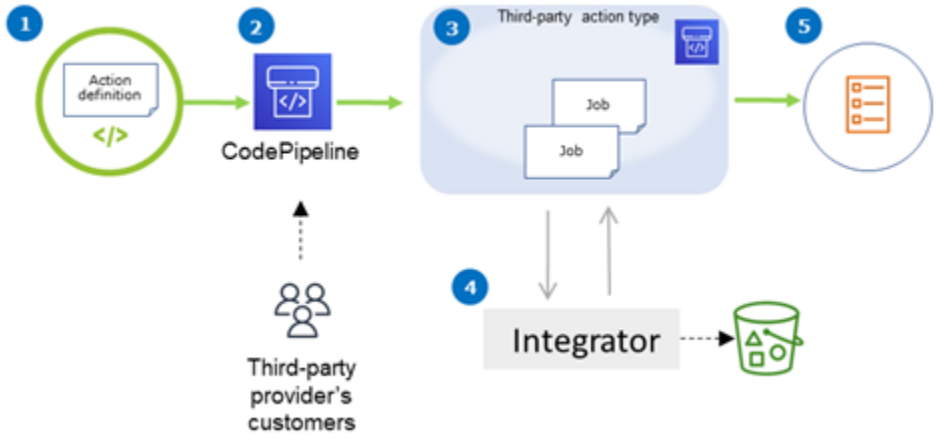


Diagram menunjukkan langkah-langkah berikut:

1. Definisi tindakan terdaftar dan tersedia di CodePipeline. Tindakan pihak ketiga tersedia untuk pelanggan penyedia pihak ketiga.
2. Pelanggan penyedia memilih dan mengonfigurasi tindakan di. CodePipeline
3. Aksi berjalan dan pekerjaan diantri. CodePipeline Ketika pekerjaan sudah siap CodePipeline, ia mengirimkan permintaan pekerjaan.
4. Integrator (pekerja pekerjaan untuk polling pihak ketiga APIs atau fungsi Lambda) mengambil permintaan pekerjaan, mengembalikan konfirmasi, dan mengerjakan artefak untuk tindakan tersebut.
5. Integrator mengembalikan success/failure output (pekerja pekerjaan menggunakan success/failure APIs or the Lambda function sends success/failure output) dengan hasil pekerjaan dan token lanjutan.

Untuk informasi tentang langkah-langkah yang dapat Anda gunakan untuk meminta, melihat, dan memperbarui jenis tindakan, lihatBekerja dengan tipe tindakan.

Konsep

Bagian ini menggunakan ketentuan berikut untuk jenis tindakan pihak ketiga:

Tipe tindakan

> Proses berulang yang dapat digunakan kembali dalam jaringan pipa yang melakukan beban kerja pengiriman berkelanjutan yang sama. Jenis tindakan diidentifikasi oleh`Owner`,, `CategoryProvider`, dan`Version`. Contoh:

```
        {

            "Category": "Deploy",
            "Owner": "AWS",
            "Provider": "CodeDeploy",
            "Version": "1"
        },
```

Semua tindakan dari jenis yang sama berbagi implementasi yang sama.

Tindakan

Contoh tunggal dari tipe tindakan, salah satu proses diskrit yang terjadi di dalam tahap pipa. Ini biasanya mencakup nilai pengguna khusus untuk pipeline tempat tindakan ini berjalan.

Definisi tindakan

Skema untuk tipe tindakan yang mendefinisikan properti yang diperlukan untuk mengkonfigurasi tindakan dan input/output artefak.

Eksekusi tindakan

Kumpulan pekerjaan yang telah dijalankan untuk menentukan apakah tindakan pada pipa pelanggan berhasil atau tidak.

Mesin eksekusi aksi

Properti konfigurasi eksekusi tindakan yang mendefinisikan tipe integrasi yang digunakan oleh tipe tindakan. Nilai yang valid adalah `JobWorker` dan `Lambda`.

Integrasi

Menjelaskan perangkat lunak yang dijalankan oleh integrator untuk mengimplementasikan tipe tindakan. CodePipeline mendukung dua jenis integrasi yang sesuai dengan dua mesin aksi yang didukung `JobWorker` dan`Lambda`.

Integrator

Orang yang memiliki implementasi tipe tindakan.

Pekerjaan

Sepotong pekerjaan dengan pipeline dan konteks pelanggan untuk melaksanakan integrasi. Eksekusi tindakan terdiri dari satu atau lebih pekerjaan.

Pekerja Job

Layanan yang memproses input pelanggan dan menjalankan pekerjaan.

Model integrasi yang didukung

CodePipeline memiliki dua model integrasi:

- Model integrasi Lambda: Model integrasi ini adalah cara yang lebih disukai untuk bekerja dengan tipe tindakan di. CodePipeline Model integrasi Lambda menggunakan fungsi Lambda untuk memproses permintaan pekerjaan saat tindakan Anda berjalan.
- Model integrasi pekerja pekerjaan: Model integrasi pekerja kerja adalah model yang sebelumnya digunakan untuk integrasi pihak ketiga. Model integrasi pekerja pekerjaan menggunakan pekerja pekerjaan yang dikonfigurasi untuk menghubungi CodePipeline APIs untuk memproses permintaan pekerjaan saat tindakan Anda berjalan.

Sebagai perbandingan, tabel berikut menjelaskan fitur dari dua model:

	Model integrasi Lambda	Model integrasi pekerja Job
Deskripsi	Integrator menulis integrasi sebagai fungsi Lambda, yang dipanggil CodePipeline oleh setiap kali ada pekerjaan yang tersedia untuk tindakan tersebut. Fungsi Lambda tidak melakukan polling untuk pekerjaan yang tersedia melainkan menunggu sampai permintaan pekerjaan berikutnya diterima.	Integrator menulis integrasi sebagai pekerja kerja yang terus-menerus melakukan polling untuk pekerjaan yang tersedia di jaringan pipa pelanggan. Pekerja pekerjaan kemudian mengeksekusi pekerjaan dan menyerahkan hasil pekerjaan kembali CodePipeline dengan menggunakan. CodePipeline APIs
Infrastruktur	AWS Lambda	Terapkan kode pekerja kerja ke infrastruktur integrator, seperti instans Amazon EC2.

	Model integrasi Lambda	Model integrasi pekerja Job
Upaya pengembangan	Integrasi hanya berisi logika bisnis.	Integrasi perlu berinteraksi dengan CodePipeline APIs selain mengandung logika bisnis.
Upaya operasi	Upaya operasi yang lebih rendah karena infrastruktur hanyalah AWS sumber daya.	Upaya operasi yang lebih tinggi karena pekerja pekerjaan membutuhkan perangkat keras mandiri.
Waktu Jalankan Pekerjaan Maks	Jika integrasi perlu dijalanka n secara aktif selama lebih dari 15 menit, model ini tidak dapat digunakan. Tindakan ini untuk integrator yang perlu memulai proses (misalnya, memulai pembuatan artefak kode pelanggan) dan mengembalikan hasilnya ketika selesai. Kami tidak menyarankan integrator terus menunggu build selesai. Sebalikny a, kembalikan kelanjutan. CodePipelinemembuat pekerjaan baru dalam 30 detik lagi jika kelanjutan diterima dari kode integrator untuk memeriksa pekerjaan sampai selesai.	Pekerjaan berjalan sangat lama (jam/hari) dapat dipertahankan menggunakan model ini.

Model integrasi Lambda

Model integrasi Lambda yang didukung mencakup pembuatan fungsi Lambda dan menentukan output untuk tipe tindakan pihak ketiga.

Perbarui fungsi Lambda Anda untuk menangani input dari CodePipeline

Anda dapat membuat fungsi Lambda baru. Anda dapat menambahkan logika bisnis ke fungsi Lambda yang dijalankan setiap kali ada pekerjaan yang tersedia di pipeline untuk jenis tindakan Anda. Misalnya, mengingat konteks pelanggan dan pipeline, Anda mungkin ingin memulai membangun layanan Anda untuk pelanggan.

Gunakan parameter berikut untuk memperbarui fungsi Lambda Anda untuk menangani input dari. CodePipeline

Format:

- `jobId`:
  - ID unik yang dihasilkan sistem dari pekerjaan.
  - Tipe: String
  - Pola: [0-9a-f] {8} - [0-9a-f] {4} - [0-9a-f] {4} - [0-9a-f] {4} - [0-9a-f] {12}
- `accountId`:
  - ID AWS akun pelanggan untuk digunakan saat melakukan pekerjaan.
  - Tipe: String
  - Pola: [0-9] {12}
- `data`:
  - Informasi lain tentang pekerjaan yang digunakan integrasi untuk menyelesaikan pekerjaan.
  - Berisi peta berikut ini:
    - `actionConfiguration`:
      - Data konfigurasi untuk tindakan. Bidang konfigurasi tindakan adalah pemetaan pasangan nilai kunci bagi pelanggan Anda untuk memasukkan nilai. Kunci ditentukan oleh parameter kunci dalam file definisi tipe tindakan saat Anda mengatur tindakan. Dalam contoh ini, nilai ditentukan oleh pengguna tindakan yang menentukan informasi di `Password` bidang `Username` dan.
      - Jenis: String ke peta string, secara opsional hadir

        Contoh:

```
"configuration": {
    "Username": "MyUser",
```

```
        "Password": "MyPassword"
    },
```

- `encryptionKey`:
  - Merupakan informasi tentang kunci yang digunakan untuk mengenkripsi data di toko artefak, seperti kunci. AWS KMS
  - Isi: Jenis tipe data`encryptionKey`, opsional hadir
- `inputArtifacts`:
  - Daftar informasi tentang artefak yang akan dikerjakan, seperti menguji atau membangun artefak.
  - Isi: Daftar tipe data`Artifact`, opsional hadir
- `outputArtifacts`:
  - Daftar informasi tentang output dari suatu tindakan.
  - Isi: Daftar tipe data`Artifact`, opsional hadir
- `actionCredentials`:
  - Merupakan objek kredensial AWS sesi. Kredensil ini adalah kredensil sementara yang dikeluarkan oleh. AWS STS Mereka dapat digunakan untuk mengakses artefak input dan output di bucket S3 yang digunakan untuk menyimpan artefak untuk pipa. CodePipeline

    Kredensyal ini juga memiliki izin yang sama dengan templat pernyataan kebijakan yang ditentukan dalam file definisi tipe tindakan.
  - Isi: Jenis tipe data`AWSSessionCredentials`, opsional hadir
- `actionExecutionId`:
  - ID eksternal dari menjalankan tindakan.
  - Tipe: String
- `continuationToken`:
  - Token yang dihasilkan sistem, seperti ID penerapan, diperlukan oleh pekerjaan untuk melanjutkan pekerjaan secara asinkron.
  - Jenis: String, opsional hadir

Tipe Data:

- `encryptionKey`:
  - `id`:

    - ID yang digunakan untuk mengidentifikasi kunci. Untuk AWS KMS kunci, Anda dapat menggunakan ID kunci, kunci ARN, atau alias ARN.
    - Tipe: String
  - `type`:
    - Jenis kunci enkripsi, seperti AWS KMS kunci.
    - Tipe: String
    - Nilai yang valid: `KMS`
- `Artifact`:
  - `name`:
    - Nama artefak.
    - Jenis: String, opsional hadir
  - `revision`:
    - ID revisi artefak. Bergantung pada jenis objek, ini bisa berupa ID komit (GitHub) atau ID revisi (Amazon S3).
    - Jenis: String, opsional hadir
  - `location`:
    - Lokasi artefak.
    - Isi: Jenis tipe data`ArtifactLocation`, opsional hadir
- `ArtifactLocation`:
  - `type`:
    - Jenis artefak di lokasi.
    - Jenis: String, opsional hadir
    - Nilai yang valid: `S3`
  - `s3Location`:
    - Lokasi bucket S3 yang berisi revisi.
    - Isi: Jenis tipe data`S3Location`, opsional hadir
- `S3Location`:
  - `bucketName`:
    - Nama bucket S3.
    - Tipe: String
  - `objectKey`:

    - Kunci objek dalam ember S3, yang secara unik mengidentifikasi objek dalam ember.
    - Tipe: String
- `AWSSessionCredentials`:
  - `accessKeyId`:
    - Kunci akses untuk sesi.
    - Tipe: String
  - `secretAccessKey`:
    - Kunci akses rahasia untuk sesi tersebut.
    - Tipe: String
  - `sessionToken`:
    - Token untuk sesi.
    - Tipe: String

Contoh:

```
{
    "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"
    }
}
```

Kembalikan hasil dari fungsi Lambda Anda ke CodePipeline

Sumber daya pekerja kerja integrator harus mengembalikan muatan yang valid dalam kasus keberhasilan, kegagalan, atau kelanjutan.

Format:

- `result`: Hasil pekerjaan.
  - Diperlukan
  - Nilai yang valid (tidak peka huruf besar/kecil):
    - `Success`: Menunjukkan pekerjaan berhasil dan terminal.
    - `Continue`: Menunjukkan pekerjaan berhasil dan harus dilanjutkan, misalnya jika pekerja pekerjaan dipanggil kembali untuk eksekusi tindakan yang sama.
    - `Fail`: Menunjukkan pekerjaan telah gagal dan terminal.
- `failureType`: Jenis kegagalan untuk dikaitkan dengan pekerjaan yang gagal.

`failureType`Kategori untuk tindakan mitra menggambarkan jenis kegagalan yang ditemui saat menjalankan pekerjaan. Integrator mengatur tipe bersama dengan pesan kegagalan saat mengembalikan hasil kegagalan pekerjaan kembali ke CodePipeline.

- Tidak wajib. Diperlukan jika hasilnya`Fail`.
- Harus null jika `result` ada `Success` atau `Continue`
- Nilai valid:
  - ConfigurationError
  - JobFailed
  - PermissionsError
  - RevisionOutOfSync
  - RevisionUnavailable
  - SystemUnavailable

- `continuation`: Status kelanjutan yang akan diteruskan ke pekerjaan berikutnya dalam eksekusi tindakan saat ini.
  - Tidak wajib. Diperlukan jika hasilnya`Continue`.
  - Harus null jika `result` ada `Success` atau`Fail`.
  - Properti:
    - `State`: Sebuah hash negara yang akan dilewati.
- `status`: Status eksekusi tindakan.
  - Tidak wajib.
  - Properti:
    - `ExternalExecutionId`: ID eksekusi eksternal opsional atau ID komit untuk dikaitkan dengan pekerjaan.
    - `Summary`: Ringkasan opsional tentang apa yang terjadi. Dalam skenario kegagalan, ini menjadi pesan kegagalan yang dilihat pengguna.
- `outputVariables`: Satu set key/value pasangan yang akan diteruskan ke eksekusi tindakan berikutnya.
  - Tidak wajib.
  - Harus null jika `result` ada `Continue` atau`Fail`.

Contoh:

```
{
    "result": "success",
    "failureType": null,
    "continuation": null,
    "status": {
        "externalExecutionId": "my-commit-id-123",
        "summary": "everything is dandy"
    },
    "outputVariables": {
        "FirstOne": "Nice",
        "SecondOne": "Nicest",
        ...
    }
}
```

Gunakan token lanjutan untuk menunggu hasil dari proses asinkron

`continuation`Token adalah bagian dari payload dan hasil dari fungsi Lambda Anda. Ini adalah cara untuk meneruskan status pekerjaan ke CodePipeline dan menunjukkan bahwa pekerjaan itu perlu dilanjutkan. Misalnya, setelah integrator memulai build untuk pelanggan di sumber daya mereka, integrator tidak menunggu build selesai, tetapi menunjukkan CodePipeline bahwa integrator tidak memiliki hasil terminal dengan mengembalikan `result` as `continue` dan mengembalikan ID unik build ke CodePipeline as token. `continuation`

> **Note**
>
> Fungsi Lambda hanya dapat berjalan hingga 15 menit. Jika pekerjaan perlu berjalan lebih lama, Anda dapat menggunakan token lanjutan.

CodePipeline Tim memanggil integrator setelah 30 detik dengan `continuation` token yang sama di muatannya sehingga dapat memeriksanya untuk penyelesaiannya. Jika build selesai, integrator mengembalikan success/fail hasil terminal, yang lain berlanjut.

Berikan CodePipeline izin untuk menjalankan fungsi Lambda integrator saat runtime

Anda menambahkan izin ke fungsi Lambda integrator Anda untuk menyediakan CodePipeline layanan dengan izin untuk memanggilnya menggunakan prinsip layanan:. CodePipeline

`codepipeline.amazonaws.com` Anda dapat menambahkan izin dengan menggunakan CloudFormation atau baris perintah. Sebagai contoh, lihat Bekerja dengan tipe tindakan.

Model integrasi pekerja Job

Setelah Anda merancang alur kerja tingkat tinggi Anda, Anda dapat membuat pekerja kerja Anda. Meskipun spesifik tindakan pihak ketiga menentukan apa yang dibutuhkan untuk pekerja kerja, sebagian besar pekerja kerja untuk tindakan pihak ketiga mencakup fungsionalitas berikut:

- Polling untuk pekerjaan dari CodePipeline penggunaan`PollForThirdPartyJobs`.
- Mengakui pekerjaan dan mengembalikan hasil untuk CodePipeline menggunakan`AcknowledgeThirdPartyJob`,`PutThirdPartyJobSuccessResult`, dan. `PutThirdPartyJobFailureResult`
- Mengambil artefak dari and/or memasukkan artefak ke dalam ember Amazon S3 untuk pipa. Untuk mengunduh artefak dari bucket Amazon S3, Anda harus membuat klien Amazon S3 yang menggunakan penandatanganan Signature Version 4 (Sig V4). Sig V4 diperlukan untuk. AWS KMS

  Untuk mengunggah artefak ke bucket Amazon S3, Anda juga harus mengonfigurasi permintaan Amazon `PutObject` S3 untuk menggunakan enkripsi melalui (). AWS Key Management Service AWS KMS AWS KMS menggunakan AWS KMS keys. Untuk mengetahui apakah akan menggunakan Kunci yang dikelola AWS atau kunci yang dikelola pelanggan untuk mengunggah artefak, pekerja kerja Anda harus melihat data pekerjaan dan memeriksa properti kunci enkripsi. Jika properti disetel, Anda harus menggunakan ID kunci terkelola pelanggan tersebut saat mengonfigurasi. AWS KMS Jika properti kunci adalah null, Anda menggunakan. Kunci yang dikelola AWS CodePipeline menggunakan Kunci yang dikelola AWS kecuali dikonfigurasi lain.

  Untuk contoh yang menunjukkan cara membuat AWS KMS parameter di Java atau .NET, lihat Menentukan AWS Key Management Service di Amazon S3 Menggunakan. AWS SDKs Untuk informasi selengkapnya tentang bucket Amazon S3 CodePipeline, lihat. CodePipeline konsep

Pilih dan konfigurasikan strategi manajemen izin untuk pekerja kerja Anda

Untuk mengembangkan pekerja kerja untuk tindakan pihak ketiga Anda di CodePipeline, Anda memerlukan strategi untuk integrasi pengguna dan manajemen izin.

Strategi paling sederhana adalah menambahkan infrastruktur yang Anda butuhkan untuk pekerja pekerjaan Anda dengan membuat instans Amazon EC2 dengan peran instans AWS Identity and

Access Management (IAM), yang memungkinkan Anda meningkatkan sumber daya yang Anda butuhkan untuk integrasi dengan mudah. Anda dapat menggunakan integrasi bawaan AWS untuk menyederhanakan interaksi antara pekerja kerja Anda dan CodePipeline.

Pelajari lebih lanjut tentang Amazon EC2 dan tentukan apakah itu pilihan yang tepat untuk integrasi Anda. Untuk selengkapnya, lihat Amazon EC2 - Hosting Server Virtual. Untuk informasi tentang cara menyiapkan instans Amazon EC2, lihat Memulai Instans Amazon EC2 Linux.

Strategi lain yang perlu dipertimbangkan adalah menggunakan federasi identitas dengan IAM untuk mengintegrasikan sistem dan sumber daya penyedia identitas yang ada. Strategi ini berguna jika Anda sudah memiliki penyedia identitas perusahaan atau sudah dikonfigurasi untuk mendukung pengguna menggunakan penyedia identitas web. Federasi identitas memungkinkan Anda memberikan akses aman ke AWS sumber daya, termasuk CodePipeline, tanpa harus membuat atau mengelola pengguna IAM. Anda dapat menggunakan fitur dan kebijakan untuk persyaratan keamanan kata sandi dan rotasi kredensi. Anda dapat menggunakan contoh aplikasi sebagai template untuk desain Anda sendiri. Untuk selengkapnya, lihat Mengelola Federasi.

Untuk memberikan akses dan menambahkan izin bagi pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti instruksi di Buat rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center .
- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti instruksi dalam Buat peran untuk penyedia identitas pihak ketiga (federasi) dalam Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Buat peran untuk pengguna IAM dalam Panduan Pengguna IAM.
  - (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

Berikut ini adalah contoh kebijakan yang mungkin Anda buat untuk digunakan dengan pekerja pekerjaan pihak ketiga Anda. Kebijakan ini dimaksudkan sebagai contoh saja dan disediakan apa adanya.

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/"

      ]
    }
  ]
}
```

Referensi file definisi gambar

Bagian ini hanya referensi. Untuk informasi tentang membuat pipeline dengan sumber atau tindakan penerapan untuk kontainer, lihatBuat pipeline, tahapan, dan tindakan.

AWS CodePipeline pekerja pekerjaan untuk tindakan penampung, seperti tindakan sumber Amazon ECR atau tindakan penerapan Amazon ECS, menggunakan file definisi untuk memetakan URI gambar dan nama kontainer ke definisi tugas. Setiap file definisi adalah file berformat JSON yang digunakan oleh penyedia tindakan sebagai berikut:

- Penerapan standar Amazon ECS memerlukan `imagedefinitions.json` file sebagai masukan untuk tindakan penerapan. Untuk tutorial yang menggunakan tindakan penerapan standar Amazon ECS CodePipeline, lihat. Tutorial: Penerapan Standar Amazon ECS dengan CodePipeline Untuk contoh tutorial lain yang menggunakan tindakan penerapan standar Amazon ECS CodePipeline bersama dengan ECRBuild AndPublish tindakan, lihat. Tutorial: Buat dan dorong gambar Docker ke Amazon ECR dengan CodePipeline (tipe V2)
- blue/green Penerapan Amazon ECS memerlukan `imageDetail.json` file sebagai masukan untuk tindakan penerapan. Untuk tutorial dengan contoh blue/green penerapan, lihatTutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR.
  - Tindakan sumber Amazon ECR menghasilkan `imageDetail.json` file yang disediakan sebagai output dari tindakan sumber.

Topik

- file imagedefinitions.json untuk tindakan penerapan standar Amazon ECS
- File ImageDetail.json untuk tindakan penerapan Amazon ECS blue/green

file imagedefinitions.json untuk tindakan penerapan standar Amazon ECS

Dokumen definisi gambar adalah file JSON yang menjelaskan nama wadah Amazon ECS Anda serta gambar dan tag. Jika Anda menerapkan aplikasi berbasis kontainer, Anda harus membuat file definisi gambar untuk menyediakan wadah Amazon ECS dan identifikasi gambar kepada pekerja CodePipeline pekerjaan untuk diambil dari repositori gambar, seperti Amazon ECR.

> **Note**
>
> Nama file default untuk file tersebut adalah`imagedefinitions.json`. Jika Anda memilih untuk menggunakan nama file yang berbeda, Anda harus menyediakannya saat membuat tahap penyebaran pipeline.

Buat `imagedefinitions.json` file dengan pertimbangan berikut:

- File harus menggunakan pengkodean UTF-8.
- Batas ukuran file maksimum untuk file definisi gambar adalah 100 KB.
- Anda harus membuat file sebagai sumber atau membangun artefak sehingga merupakan artefak input untuk tindakan penerapan. Dengan kata lain, pastikan file tersebut diunggah ke lokasi sumber Anda, seperti CodeCommit repositori Anda, atau dihasilkan sebagai artefak keluaran bawaan.

`imagedefinitions.json`File ini menyediakan nama kontainer dan URI gambar. Itu harus dibangun dengan set pasangan kunci-nilai berikut.

Key	Nilai
name	*container_name*
imageUri	*imageUri*

> **Note**
>
> Bidang nama digunakan untuk nama gambar kontainer, yang berarti nama untuk gambar Docker.

Berikut adalah struktur JSON, di mana nama kontainer adalah`sample-app`, URI gambar`ecs-repo`, dan tag adalah`latest`:

```
[
  {
    "name": "sample-app",
    "imageUri": "11111EXAMPLE.dkr.ecr.us-west-2.amazonaws.com/ecs-repo:latest"
```

```
  }
]
```

Anda juga dapat membuat file untuk mencantumkan beberapa pasangan gambar kontainer.

Struktur JSON:

```
[
  {
    "name": "simple-app",
    "imageUri": "httpd:2.4"
  },
  {
    "name": "simple-app-1",
    "imageUri": "mysql"
  },
  {
    "name": "simple-app-2",
    "imageUri": "java1.8"
  }
]
```

Sebelum membuat pipeline, gunakan langkah-langkah berikut untuk menyiapkan `imagedefinitions.json` file.

1. Sebagai bagian dari perencanaan penerapan aplikasi berbasis container untuk pipeline Anda, rencanakan tahap sumber dan tahap pembuatan, jika berlaku.
2. Pilih salah satu cara berikut:

   a. Jika pipeline Anda dibuat sehingga melewati tahap pembuatan, Anda harus membuat file JSON secara manual dan mengunggahnya ke repositori sumber Anda sehingga tindakan sumber dapat menyediakan artefak. Buat file menggunakan editor teks, dan beri nama file atau gunakan nama `imagedefinitions.json` file default. Dorong file definisi gambar ke repositori sumber Anda.

   > **Note**
   >
   > Jika repositori sumber Anda adalah bucket Amazon S3, ingatlah untuk zip file JSON.

   b. Jika pipeline Anda memiliki tahap build, tambahkan perintah ke file spesifikasi build yang menampilkan file definisi gambar di repositori sumber selama fase build. Contoh berikut

menggunakan printf perintah untuk membuat `imagedefinitions.json` file. Buat daftar perintah ini di `post_build` bagian `buildspec.yml` file:

```
printf '[{"name":"container_name","imageUri":"image_URI"}]' >
imagedefinitions.json
```

Anda harus menyertakan file definisi gambar sebagai artefak keluaran dalam `buildspec.yml` file.

3. Saat Anda membuat pipeline di konsol, pada halaman Deploy wizard Create Pipeline, di Image Filename, masukkan nama file definisi gambar.

Untuk step-by-step tutorial membuat pipeline yang menggunakan Amazon ECS sebagai penyedia penyebaran, lihat Tutorial: Penerapan Berkelanjutan dengan. CodePipeline

File ImageDetail.json untuk tindakan penerapan Amazon ECS blue/green

`imageDetail.json`Dokumen adalah file JSON yang menjelaskan URI gambar Amazon ECS Anda. Jika Anda menerapkan aplikasi berbasis kontainer untuk blue/green penerapan, Anda harus membuat `imageDetail.json` file untuk menyediakan Amazon ECS dan CodeDeploy job worker dengan identifikasi gambar untuk diambil dari repositori gambar, seperti Amazon ECR.

> **Note**
>
> Nama file harus`imageDetail.json`.

Untuk deskripsi tindakan dan parameternya, lihatAmazon Elastic Container Service dan CodeDeploy referensi aksi penerapan biru-hijau.

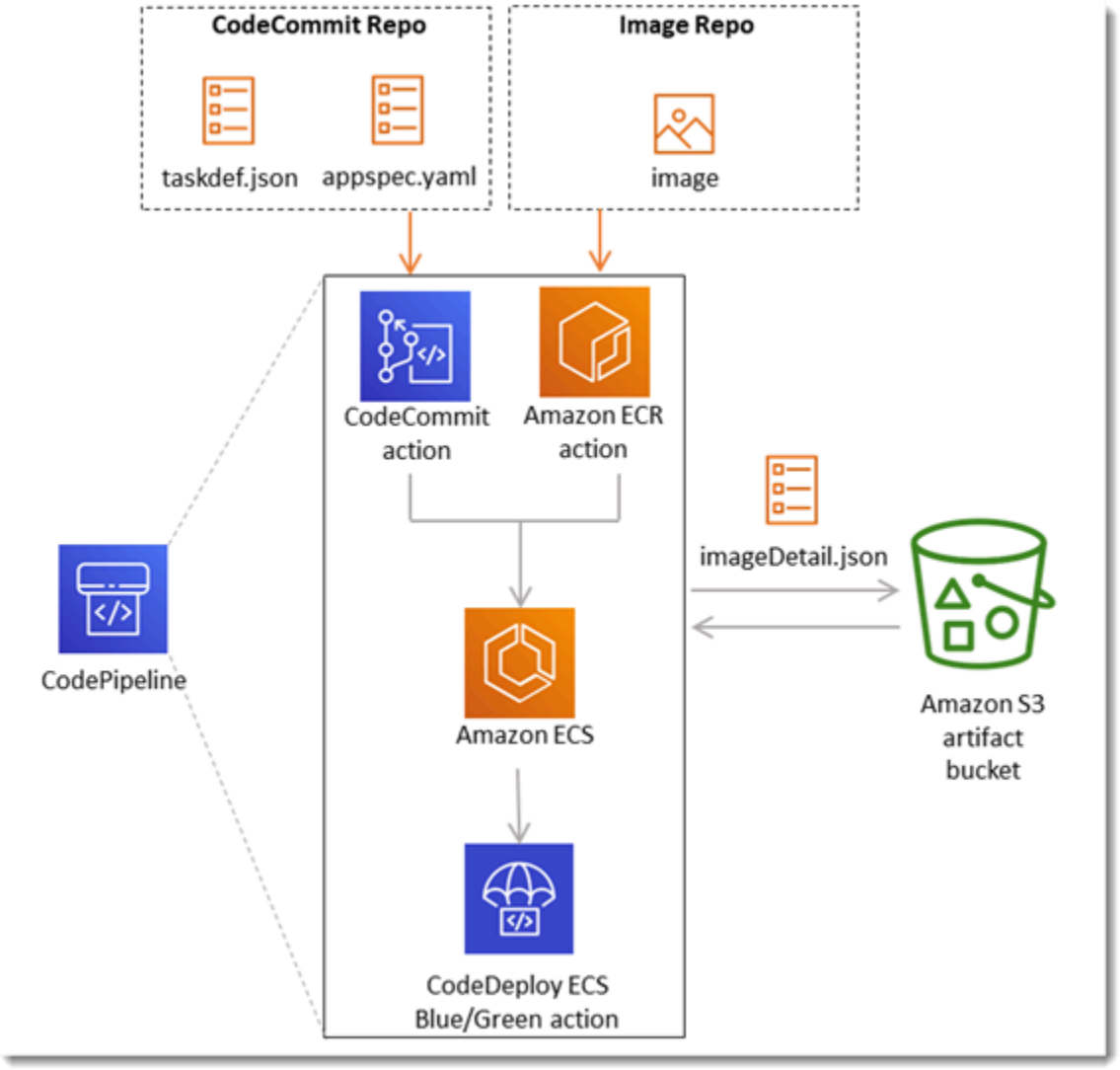
Anda harus membuat `imageDetail.json` file sebagai sumber atau membangun artefak sehingga merupakan artefak input untuk tindakan penerapan. Anda dapat menggunakan salah satu metode ini untuk menyediakan `imageDetail.json` file dalam pipeline:

- Sertakan `imageDetail.json` file di lokasi sumber Anda sehingga disediakan dalam pipeline sebagai masukan ke tindakan blue/green penerapan Amazon ECS Anda.

> **Note**
>
> Jika repositori sumber Anda adalah bucket Amazon S3, ingatlah untuk zip file JSON.

- Tindakan sumber Amazon ECR secara otomatis menghasilkan `imageDetail.json` file sebagai artefak masukan untuk tindakan selanjutnya.

> **Note**
>
> Karena tindakan sumber Amazon ECR membuat file ini, saluran pipa dengan tindakan sumber Amazon ECR tidak perlu menyediakan file secara manual. `imageDetail.json` Untuk tutorial tentang membuat pipeline yang menyertakan tahap sumber Amazon ECR, lihatTutorial: Membuat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR.



`imageDetail.json`File ini menyediakan URI gambar. Itu harus dibangun dengan pasangan kunci-nilai berikut.

Key	Nilai
ImageURI	*`image_URI`*

imageDetail.json

Berikut adalah struktur JSON, di mana URI gambar adalah`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)

`imageDetail.json`File dihasilkan secara otomatis oleh tindakan sumber Amazon ECR setiap kali perubahan didorong ke repositori gambar. Tindakan sumber Amazon ECR yang `imageDetail.json` dihasilkan disediakan sebagai artefak keluaran dari aksi sumber ke tindakan berikutnya dalam pipeline.

Berikut adalah struktur JSON, di mana nama repositori adalah`dk-image-repo`, URI gambar`ecs-repo`, dan tag gambar adalah: `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"
    ]
}
```

`imageDetail.json`File memetakan URI gambar dan nama kontainer ke definisi tugas Amazon ECS sebagai berikut:

- `ImageSizeInBytes`: Ukuran, dalam byte, dari gambar di repositori.
- `ImageDigest`: `sha256` Intisari manifes gambar.
- `Version`: Versi gambar.
- `ImagePushedAt`: Tanggal dan waktu ketika gambar terbaru didorong ke repositori.
- `RegistryId`: ID AWS akun yang terkait dengan registri yang berisi repositori.
- `RepositoryName`: Nama repositori Amazon ECR tempat gambar didorong.
- `ImageURI`: URI untuk gambar.

- `ImageTags`: Tag yang digunakan untuk gambar.

Sebelum membuat pipeline, gunakan langkah-langkah berikut untuk menyiapkan `imageDetail.json` file.

1. Sebagai bagian dari perencanaan blue/green penerapan aplikasi berbasis container untuk pipeline Anda, rencanakan tahap sumber dan tahap pembuatan, jika berlaku.
2. Pilih salah satu cara berikut:

   a. Jika pipeline Anda telah melewati tahap pembuatan, Anda harus membuat file JSON secara manual dan mengunggahnya ke repositori sumber Anda, seperti CodeCommit, sehingga tindakan sumber dapat menyediakan artefak. Buat file menggunakan editor teks, dan beri nama file atau gunakan nama `imageDetail.json` file default. Dorong `imageDetail.json` file ke repositori sumber Anda.

   b. Jika pipeline Anda memiliki tahap build, lakukan hal berikut:

      i. Tambahkan perintah ke file spesifikasi build Anda yang menampilkan file definisi gambar di repositori sumber Anda selama fase build. Contoh berikut menggunakan printf perintah untuk membuat `imageDetail.json` file. Buat daftar perintah ini di `post_build` bagian file buildspec.yl:

      ```
      printf '{"ImageURI":"image_URI"}' > imageDetail.json
      ```

      Anda harus menyertakan `imageDetail.json` file sebagai artefak keluaran dalam `buildspec.yml` file.

      ii. Tambahkan `imageDetail.json` sebagai file artefak dalam `buildspec.yml` file.

      ```
      artifacts:
        files:
          - imageDetail.json
      ```

Referensi variabel

Bagian ini hanya referensi. Untuk informasi tentang membuat variabel, lihatBekerja dengan variabel.

Variabel memungkinkan Anda mengonfigurasi tindakan pipeline dengan nilai yang ditentukan pada saat eksekusi pipeline atau eksekusi tindakan.

Beberapa penyedia tindakan menghasilkan serangkaian variabel yang ditentukan. Anda memilih dari kunci variabel default untuk penyedia tindakan tersebut, seperti ID komit.

> **Important**
>
> Saat melewati parameter rahasia, jangan masukkan nilai secara langsung. Nilai tersebut diberikan sebagai teks biasa dan karena itu dapat dibaca. Untuk alasan keamanan, jangan gunakan plaintext dengan rahasia. Kami sangat menyarankan Anda menggunakan AWS Secrets Manager untuk menyimpan rahasia.

Untuk melihat step-by-step contoh penggunaan variabel:

- Untuk tutorial dengan variabel tingkat pipa yang dilewatkan pada saat eksekusi pipeline, lihat. Tutorial: Gunakan variabel tingkat pipa
- Untuk tutorial dengan tindakan Lambda yang menggunakan variabel dari tindakan upstream (CodeCommit) dan menghasilkan variabel keluaran, lihat. Tutorial: Menggunakan variabel dengan tindakan panggilan Lambda
- Untuk tutorial dengan CloudFormation tindakan yang mereferensikan variabel keluaran tumpukan dari CloudFormation tindakan upstream, lihatTutorial: Buat pipeline yang menggunakan variabel dari AWS CloudFormation tindakan penerapan.
- Untuk contoh tindakan persetujuan manual dengan teks pesan yang mereferensikan variabel keluaran yang diselesaikan ke ID CodeCommit komit dan pesan komit, lihatContoh: Gunakan variabel dalam persetujuan manual.
- Untuk contoh CodeBuild tindakan dengan variabel lingkungan yang menyelesaikan nama GitHub cabang, lihat. Contoh: Gunakan BranchName variabel dengan variabel CodeBuild lingkungan
- CodeBuild tindakan menghasilkan sebagai variabel semua variabel lingkungan yang diekspor sebagai bagian dari build. Untuk informasi selengkapnya, lihat CodeBuild variabel keluaran aksi.

Batas Variabel

Untuk informasi batas, lihatKuota di AWS CodePipeline.

> **Note**
>
> Saat Anda memasukkan sintaks variabel di bidang konfigurasi tindakan, jangan melebihi batas 1000 karakter untuk bidang konfigurasi. Kesalahan validasi dikembalikan ketika batas ini terlampaui.

Topik

- Konsep
- Gunakan kasus untuk variabel
- Mengkonfigurasi variabel
- Resolusi variabel
- Aturan untuk variabel
- Variabel tersedia untuk tindakan pipeline

Konsep

Bagian ini mencantumkan istilah dan konsep utama yang terkait dengan variabel dan ruang nama.

Variabel

Variabel adalah pasangan nilai kunci yang dapat digunakan untuk mengonfigurasi tindakan secara dinamis di pipeline Anda. Saat ini ada tiga cara variabel-variabel ini tersedia:

- Ada satu set variabel yang secara implisit tersedia pada awal setiap eksekusi pipeline. Set ini saat ini termasuk`PipelineExecutionId`, ID dari eksekusi pipeline saat ini.
- Variabel pada tingkat pipa didefinisikan ketika pipa dibuat dan diselesaikan pada waktu proses pipa.

  Anda menentukan variabel tingkat pipeline saat pipeline dibuat, dan Anda dapat memberikan nilai pada saat eksekusi pipeline.
- Ada jenis tindakan yang menghasilkan set variabel ketika mereka dieksekusi. Anda dapat melihat variabel yang dihasilkan oleh tindakan dengan memeriksa `outputVariables` bidang yang

merupakan bagian dari ListActionExecutionsAPI. Untuk daftar nama kunci yang tersedia menurut penyedia tindakan, lihatVariabel tersedia untuk tindakan pipeline. Untuk melihat variabel mana yang dihasilkan setiap jenis tindakan, lihat CodePipeline Referensi struktur aksi.

Untuk mereferensikan variabel-variabel ini dalam konfigurasi tindakan Anda, Anda harus menggunakan sintaks referensi variabel dengan namespace yang benar.

Untuk contoh alur kerja variabel, lihatMengkonfigurasi variabel.

Namespace

Untuk memastikan bahwa variabel dapat direferensikan secara unik, mereka harus ditetapkan ke namespace. Setelah Anda memiliki satu set variabel yang ditetapkan ke namespace, mereka dapat direferensikan dalam konfigurasi tindakan dengan menggunakan namespace dan kunci variabel dengan sintaks berikut:

```
#{namespace.variable_key}
```

Ada tiga jenis ruang nama di mana variabel dapat ditetapkan:

- Namespace cadangan codepipeline

  Ini adalah namespace yang ditetapkan ke kumpulan variabel implisit yang tersedia di awal setiap eksekusi pipeline. Namespace ini adalah `codepipeline`. Contoh referensi variabel:

  ```
  #{codepipeline.PipelineExecutionId}
  ```

- Variabel namespace pada tingkat pipeline

  Ini adalah namespace yang ditugaskan ke variabel di tingkat pipeline. Namespace untuk semua variabel di tingkat pipeline adalah. `variables` Contoh referensi variabel:

  ```
  #{variables.variable_name}
  ```

- Tindakan yang ditetapkan namespace

  Ini adalah namespace yang Anda tetapkan untuk tindakan. Semua variabel yang dihasilkan oleh aksi berada di bawah namespace ini. Untuk membuat variabel yang dihasilkan oleh tindakan tersedia untuk digunakan dalam konfigurasi tindakan hilir, Anda harus mengonfigurasi tindakan produksi dengan namespace. Ruang nama harus unik di seluruh definisi pipeline dan tidak dapat

bertentangan dengan nama artefak apa pun. Berikut adalah contoh referensi variabel untuk tindakan yang dikonfigurasi dengan namespace dari. `SourceVariables`

```
#{SourceVariables.VersionId}
```

Gunakan kasus untuk variabel

Berikut ini adalah beberapa kasus penggunaan yang paling umum untuk variabel di tingkat pipeline, membantu Anda menentukan bagaimana Anda dapat menggunakan variabel untuk kebutuhan spesifik Anda.

- Variabel pada tingkat pipeline adalah untuk CodePipeline pelanggan yang ingin menggunakan pipeline yang sama setiap kali dengan variasi kecil dalam input ke konfigurasi tindakan. Setiap pengembang yang memulai pipeline menambahkan nilai variabel di UI saat pipeline dimulai. Dengan konfigurasi ini, Anda meneruskan parameter untuk eksekusi itu saja.
- Dengan variabel tingkat pipa, Anda dapat meneruskan input dinamis ke tindakan dalam pipeline. Anda dapat memigrasikan pipeline berparameter ke CodePipeline tanpa harus mempertahankan versi berbeda dari pipeline yang sama, atau membuat pipeline yang kompleks.
- Anda dapat menggunakan variabel tingkat pipeline untuk meneruskan parameter input yang memungkinkan Anda menggunakan kembali pipeline dengan setiap eksekusi, seperti saat Anda ingin menentukan versi mana yang ingin Anda terapkan ke lingkungan produksi, sehingga Anda tidak perlu menduplikasi pipeline.
- Anda dapat menggunakan satu pipeline untuk menyebarkan sumber daya ke beberapa lingkungan build dan deployment. Misalnya, untuk pipeline dengan CodeCommit repositori, penerapan dari cabang tertentu dan lingkungan penyebaran target dapat dilakukan dengan CodeBuild dan CodeDeploy parameter diteruskan pada tingkat pipa.

> **Note**
>
> Untuk Amazon ECR, Amazon S3, CodeCommit atau sumber, Anda juga dapat membuat penggantian sumber menggunakan entri transformasi input untuk menggunakan `revisionValue` peristiwa EventBridge in untuk pipeline Anda, yang berasal dari variabel peristiwa sumber untuk kunci objek, komit, atau ID gambar Anda. `revisionValue` Untuk informasi selengkapnya, lihat langkah opsional untuk entri transformasi masukan yang disertakan dalam prosedur di bawahTindakan sumber dan sumber daya Amazon ECR

EventBridge,Menyambungkan ke tindakan sumber Amazon S3 dengan sumber yang diaktifkan untuk acara, atauCodeCommit tindakan sumber dan EventBridge.

Mengkonfigurasi variabel

Anda dapat mengonfigurasi variabel di tingkat pipa atau tingkat tindakan dalam struktur pipa.

Mengkonfigurasi variabel di tingkat pipa

Anda dapat menambahkan satu atau lebih variabel di tingkat pipa. Anda dapat mereferensikan nilai ini dalam konfigurasi CodePipeline tindakan. Anda dapat menambahkan nama variabel, nilai default, dan deskripsi saat membuat pipeline. Variabel diselesaikan pada saat eksekusi.

> **Note**
>
> Jika nilai default tidak didefinisikan untuk variabel pada tingkat pipa, variabel dianggap sebagai diperlukan. Anda harus menentukan penggantian untuk semua variabel yang diperlukan saat Anda memulai pipeline, jika tidak, eksekusi pipeline akan gagal dengan kesalahan validasi.

Anda menyediakan variabel pada tingkat pipeline menggunakan atribut variabel dalam struktur pipa. Dalam contoh berikut, variabel `Variable1` memiliki nilai`Value1`.

```
"variables": [
     {
         "name": "Variable1",
         "defaultValue": "Value1",
         "description": "description"
     }
 ]
```

Untuk contoh dalam struktur pipa JSON, lihatBuat pipeline, tahapan, dan tindakan.

Untuk tutorial dengan variabel tingkat pipa yang dilewatkan pada saat eksekusi pipeline, lihat. Tutorial: Gunakan variabel tingkat pipa

Perhatikan bahwa menggunakan variabel tingkat pipa dalam segala jenis tindakan Sumber tidak didukung.

> **Note**
>
> Jika `variables` namespace sudah digunakan dalam beberapa tindakan dalam pipeline, Anda harus memperbarui definisi tindakan dan memilih namespace lain untuk tindakan yang bertentangan.

Mengkonfigurasi variabel pada tingkat tindakan

Anda mengonfigurasi tindakan untuk menghasilkan variabel dengan mendeklarasikan namespace untuk tindakan tersebut. Tindakan harus sudah menjadi salah satu penyedia tindakan yang menghasilkan variabel. Jika tidak, variabel yang tersedia adalah variabel tingkat pipa.

Anda mendeklarasikan namespace baik dengan:

- Pada halaman Edit tindakan konsol, masukkan namespace di Namespace Variable.
- Memasukkan namespace di bidang `namespace` parameter dalam struktur pipa JSON.

Dalam contoh ini, Anda menambahkan `namespace` parameter ke tindakan CodeCommit sumber dengan nama`SourceVariables`. Ini mengonfigurasi tindakan untuk menghasilkan variabel yang tersedia untuk penyedia tindakan tersebut, seperti`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
        }
    ]
},
```

Selanjutnya, Anda mengonfigurasi tindakan hilir untuk menggunakan variabel yang dihasilkan oleh tindakan sebelumnya. Anda melakukan ini dengan:

- Pada halaman Edit tindakan konsol, masukkan sintaks variabel (untuk tindakan hilir) di bidang konfigurasi tindakan.
- Memasukkan sintaks variabel (untuk tindakan hilir) di bidang konfigurasi tindakan dalam struktur pipa JSON

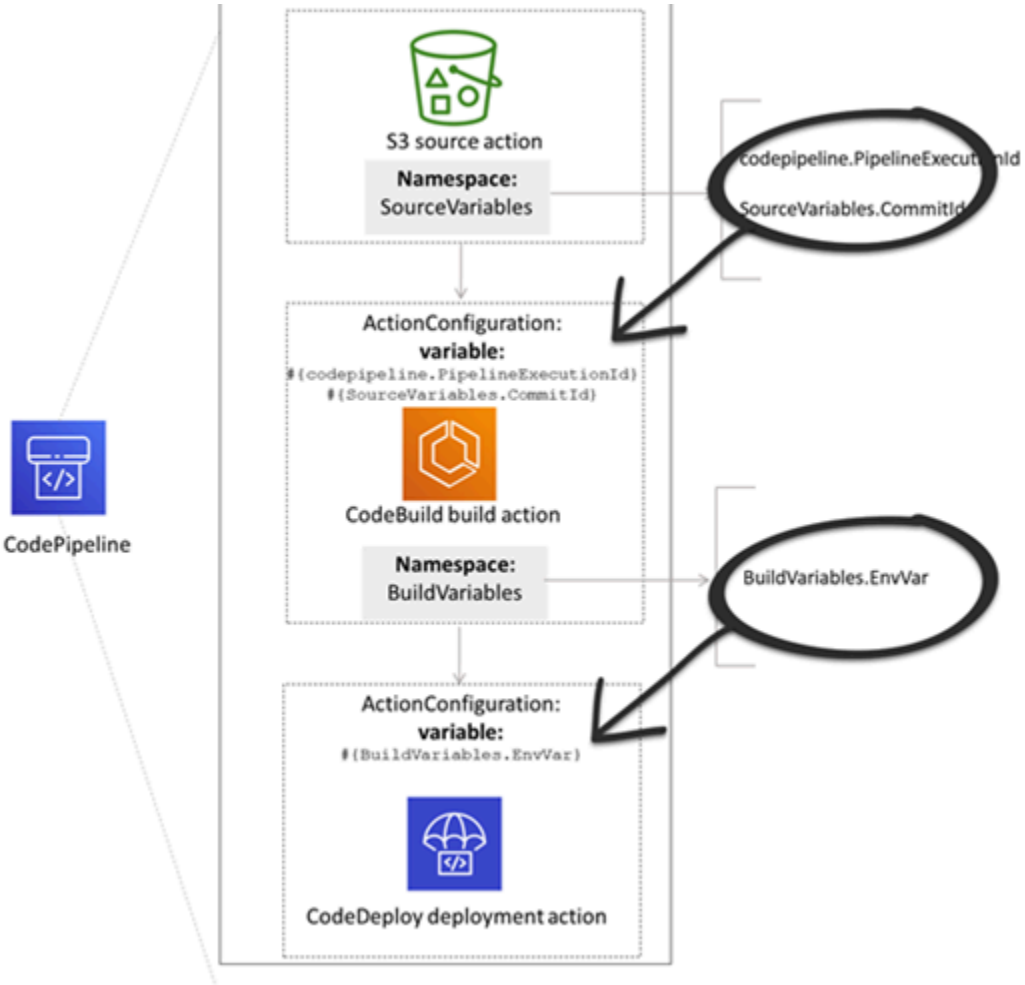
Dalam contoh ini, bidang konfigurasi tindakan build menunjukkan variabel lingkungan yang diperbarui pada eksekusi tindakan. Contoh menentukan namespace dan variabel untuk eksekusi ID dengan dan namespace `#{codepipeline.PipelineExecutionId}` dan variabel untuk komit ID dengan. **`#{SourceVariables.CommitId}`**

```
{
    "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
        }
    ]
},
```

Resolusi variabel

Setiap kali tindakan dijalankan sebagai bagian dari eksekusi pipeline, variabel yang dihasilkannya tersedia untuk digunakan dalam tindakan apa pun yang dijamin terjadi setelah tindakan produksi. Untuk menggunakan variabel-variabel ini dalam tindakan konsumsi, Anda dapat menambahkannya ke konfigurasi tindakan konsumsi menggunakan sintaks yang ditunjukkan pada contoh sebelumnya. Sebelum melakukan tindakan yang memakan, CodePipeline menyelesaikan semua referensi variabel yang ada dalam konfigurasi sebelum memulai eksekusi tindakan.



Aturan untuk variabel

Aturan berikut membantu Anda dengan konfigurasi variabel:

- Anda menentukan namespace dan variabel untuk tindakan melalui properti tindakan baru atau dengan mengedit tindakan.
- Saat Anda menggunakan panduan pembuatan pipeline, konsol akan menghasilkan namespace untuk setiap tindakan yang dibuat dengan wizard.
- Jika namespace tidak ditentukan, variabel yang dihasilkan oleh tindakan itu tidak dapat direferensikan dalam konfigurasi tindakan apa pun.
- Untuk referensi variabel yang dihasilkan oleh suatu tindakan, tindakan referensi harus terjadi setelah tindakan yang menghasilkan variabel. Ini berarti berada di tahap selanjutnya dari tindakan yang menghasilkan variabel, atau pada tahap yang sama tetapi pada urutan lari yang lebih tinggi.

Variabel tersedia untuk tindakan pipeline

Penyedia tindakan menentukan variabel mana yang dapat dihasilkan oleh tindakan.

Untuk step-by-step prosedur untuk mengelola variabel, lihatBekerja dengan variabel.

Tindakan dengan kunci variabel yang ditentukan

Tidak seperti namespace yang dapat Anda pilih, tindakan berikut menggunakan kunci variabel yang tidak dapat diedit. Misalnya, untuk penyedia tindakan Amazon S3, hanya kunci `VersionId` variabel `ETag` dan yang tersedia.

Setiap eksekusi juga memiliki satu set variabel pipeline CodePipeline yang dihasilkan yang berisi data tentang eksekusi, seperti ID rilis pipeline. Variabel-variabel ini dapat dikonsumsi oleh tindakan apa pun dalam pipeline.

Topik

- CodePipeline variabel ID eksekusi
- Variabel keluaran tindakan Amazon ECR
- CloudFormation StackSets variabel keluaran aksi
- CodeCommit variabel keluaran aksi
- CodeStarSourceConnection variabel keluaran aksi
- GitHub variabel keluaran tindakan (GitHub (melalui OAuth aplikasi) tindakan)
- Variabel keluaran aksi S3

CodePipeline variabel ID eksekusi

CodePipeline variabel ID eksekusi

Penyedia	Kunci variabel	Nilai contoh	Contoh sintaks variabel
codepipeline	`PipelineExecutionId`	8abc75f0-fbf8-4f4c-bfcontoh	`#{codepipeline.PipelineExecutionId}`

Variabel keluaran tindakan Amazon ECR

Variabel Amazon ECR

Kunci variabel	Nilai contoh	Contoh sintaks variabel
`ImageDigest`	sha256: EXAMPLE1122334455	`#{SourceVariables.ImageDigest}`
`ImageTag`	terbaru	`#{SourceVariables.ImageTag}`
`ImageURI`	11111Example.dkr. ecr.us-west-2.amazonaws.com /ecs-repo :latest	`#{SourceVariables.ImageURI}`
`RegistryId`	EXAMPLE12233	`#{SourceVariables.RegistryId}`
`RepositoryName`	my-image-repo	`#{SourceVariables.RepositoryName}`

CloudFormation StackSets variabel keluaran aksi

CloudFormation StackSets variabel

Kunci variabel	Nilai contoh	Contoh sintaks variabel
`OperationId`	11111111-2bbb-111-2bbb-11111contoh	`#{DeployVariables.OperationId}`
`StackSetId`	set tumpukan saya:1111 aaaa-1111-2222-2bbb-11111contoh	`#{DeployVariables.StackSetId}`

CodeCommit variabel keluaran aksi

CodeCommit variabel

Kunci variabel	Nilai contoh	Contoh sintaks variabel
`AuthorDate`	2019-10-29T 03:32:21 Z	`#{SourceVariables.AuthorDate}`
`BranchName`	pengembangan	`#{SourceVariables.BranchName}`
`CommitId`	contoh01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Memperbaiki bug (ukuran maksimum 100 KB)	`#{SourceVariables.CommitMessage}`
`CommitterDate`	2019-10-29T 03:32:21 Z	`#{SourceVariables.CommitterDate}`
`RepositoryName`	myCodeCommitRepo	`#{SourceVariables.RepositoryName}`

CodeStarSourceConnection variabel keluaran aksi

`CodeStarSourceConnection`variabel (Bitbucket Cloud, GitHub, GitHub Enterprise Repository, dan .com) GitLab

Kunci variabel	Nilai contoh	Contoh sintaks variabel
`AuthorDate`	2019-10-29T 03:32:21 Z	`#{SourceVariables.AuthorDate}`
`BranchName`	pengembangan	`#{SourceVariables.BranchName}`
`CommitId`	contoh01f91b31	`#{SourceVariables.CommitId}`

Kunci variabel	Nilai contoh	Contoh sintaks variabel
`CommitMessage`	Memperbaiki bug (ukuran maksimum 100 KB)	`#{SourceVariables.CommitMessage}`
`ConnectionArn`	arn:aws:codestar-koneksi:wilayah ::koneksi/*account-id connection-id*	`#{SourceVariables.ConnectionArn}`
`FullRepositoryName`	nama pengguna/ GitHubRepo	`#{SourceVariables.FullRepositoryName}`

GitHub variabel keluaran tindakan (GitHub (melalui OAuth aplikasi) tindakan)

GitHub variabel (GitHub (melalui OAuth aplikasi) tindakan)

Kunci variabel	Nilai contoh	Contoh sintaks variabel
`AuthorDate`	2019-10-29T 03:32:21 Z	`#{SourceVariables.AuthorDate}`
`BranchName`	utama	`#{SourceVariables.BranchName}`
`CommitId`	contoh01f91b31	`#{SourceVariables.CommitId}`
`CommitMessage`	Memperbaiki bug (ukuran maksimum 100 KB)	`#{SourceVariables.CommitMessage}`
`CommitterDate`	2019-10-29T 03:32:21 Z	`#{SourceVariables.CommitterDate}`
`CommitUrl`		`#{SourceVariables.CommitUrl}`
`RepositoryName`	myGitHubRepo	`#{SourceVariables.RepositoryName}`

Variabel keluaran aksi S3

Variabel S3

Kunci variabel	Nilai contoh	Contoh sintaks variabel
`ETag`	contoh28be1c3	`#{SourceVariables.ETag}`
`VersionId`	exampleta_ IUQCv	`#{SourceVariables.VersionId}`

Tindakan dengan kunci variabel yang dikonfigurasi pengguna

Untuk CodeBuild, CloudFormation, dan tindakan Lambda, kunci variabel dikonfigurasi oleh pengguna.

Topik

- CloudFormation variabel keluaran aksi
- CodeBuild variabel keluaran aksi
- Variabel keluaran aksi Lambda

CloudFormation variabel keluaran aksi

CloudFormation variabel

Kunci variabel	Contoh sintaks variabel
Untuk CloudFormation tindakan, variabel dihasilkan dari nilai apa pun yang ditunjuk di `Outputs` bagian template tumpukan. Perhatikan bahwa satu-satunya mode CloudFormation tindakan yang menghasilkan output adalah mode yang menghasilkan atau memperbarui tumpukan, seperti pembuatan tumpukan, pembaruan tumpukan, dan eksekusi set perubahan. Mode tindakan terkait yang menghasilkan variabel adalah: • CREATE_UPDATE	`#{DeployVariables.StackName}`

Kunci variabel	Contoh sintaks variabel
• CHANGE_SET_EXECUTE • CHANGE_SET_REPLACE • REPLACE_ON_FAILURE Untuk informasi selengkapnya tentang mode aksi ini, lihatCloudFormation menyebarkan referensi tindakan. Untuk tutorial yang menunjukkan cara membuat pipeline dengan tindakan CloudFormation penerapan dalam pipeline yang menggunakan variabel CloudFormation keluaran, lihatTutorial: Buat pipeline yang menggunakan variabel dari AWS CloudFormation tindakan penerapan.	

CodeBuild variabel keluaran aksi

CodeBuild variabel

Kunci variabel	Contoh sintaks variabel
Untuk CodeBuild tindakan, variabel dihasilkan dari nilai yang dihasilkan oleh variabel lingkungan yang diekspor. Siapkan variabel CodeBuild lingkungan dengan mengedit CodeBuild tindakan Anda di CodePipeline atau dengan menambahkan variabel lingkungan ke spesifikasi build. Tambahkan instruksi ke spesifikasi CodeBuild build Anda untuk menambahkan variabel lingkungan di bawah bagian variabel yang diekspor. Lihat env/exported-variable di Panduan Pengguna.AWS CodeBuild	`#{BuildVariables.EnvVar}`

Variabel keluaran aksi Lambda

Variabel Lambda

Kunci variabel	Contoh sintaks variabel
Tindakan Lambda akan menghasilkan sebagai variabel semua pasangan nilai kunci yang disertakan dalam `outputVariables` bagian permintaan API. PutJobSuccessResult Untuk tutorial dengan tindakan Lambda yang menggunakan variabel dari tindakan upstream (CodeCommit) dan menghasilkan variabel keluaran, lihat. Tutorial: Menggunakan variabel dengan tindakan panggilan Lambda	`#{TestVariables.testRunId}`

Bekerja dengan pola glob dalam sintaks

Bila Anda menentukan file atau jalur yang digunakan dalam artefak pipeline atau lokasi sumber, Anda dapat menentukan artefak tergantung pada jenis tindakan. Misalnya, untuk tindakan S3, Anda menentukan kunci objek S3.

Untuk pemicu, Anda dapat menentukan filter. Anda dapat menggunakan pola glob untuk menentukan filter. Berikut ini adalah beberapa contohnya.

Ketika sintaksnya “glob” maka representasi String dari jalur dicocokkan menggunakan bahasa pola terbatas dengan sintaks yang menyerupai ekspresi reguler. Sebagai contoh:

- `*.java`Menentukan path yang mewakili nama file yang berakhir di.java
- `*.*`Menentukan nama file yang berisi titik
- `*.{java,class}`Menentukan nama file yang diakhiri dengan.java atau.class
- `foo.?`Menentukan nama file yang dimulai dengan foo. dan ekstensi karakter tunggal

Aturan berikut digunakan untuk menafsirkan pola glob:

- Untuk menentukan nol atau lebih karakter dari komponen nama dalam batas direktori, gunakan`*`.
- Untuk menentukan nol atau lebih karakter dari komponen nama melintasi batas direktori, gunakan`**`.
- Untuk menentukan satu karakter dari komponen nama, gunakan`?`.
- Untuk menghindari karakter yang seharusnya ditafsirkan sebagai karakter khusus, gunakan karakter garis miring terbalik (). `\`
- Untuk menentukan satu karakter dari satu set karakter, gunakan`[ ]`.
- Untuk menentukan satu file yang ada di root lokasi build atau lokasi repositori sumber, gunakan. `my-file.jar`
- Untuk menentukan satu file dalam subdirektori, gunakan `directory/my-file.jar` atau`directory/subdirectory/my-file.jar`.
- Untuk menentukan semua file, gunakan`"**"`. Pola `**` glob menunjukkan untuk mencocokkan sejumlah subdirektori.
- Untuk menentukan semua file dan direktori dalam direktori bernama`directory`, gunakan`"directory/**"`. Pola `**` glob menunjukkan untuk mencocokkan sejumlah subdirektori.

- Untuk menentukan semua file dalam direktori bernama`directory`, tetapi tidak salah satu subdirektorinya, gunakan. `"directory/*"`
- Dalam ekspresi braket`*`, `?` dan `\` karakter cocok dengan dirinya sendiri. Karakter (-) cocok dengan dirinya sendiri jika itu adalah karakter pertama dalam tanda kurung, atau karakter pertama setelah `!` jika negasi.
- `{ }`Karakter adalah sekelompok subpola, di mana grup cocok jika ada subpola dalam grup yang cocok. `","`Karakter digunakan untuk memisahkan subpola. Grup tidak dapat disarangkan.

Perbarui jalur pemungutan suara ke metode deteksi perubahan yang direkomendasikan

Jika Anda memiliki pipeline yang menggunakan polling untuk bereaksi terhadap perubahan sumber, Anda dapat memperbaruinya untuk menggunakan metode deteksi yang disarankan. Untuk panduan migrasi dengan petunjuk untuk memperbarui pipline polling agar menggunakan metode deteksi perubahan berbasis peristiwa yang direkomendasikan, lihat. Migrasi jaringan pemungutan suara untuk menggunakan deteksi perubahan berbasis peristiwa

Perbarui tindakan sumber GitHub (melalui OAuth aplikasi) ke tindakan sumber GitHub (melalui GitHub Aplikasi)

Di AWS CodePipeline, ada dua versi aksi GitHub sumber yang didukung:

- Direkomendasikan: Tindakan GitHub (melalui GitHub Aplikasi) menggunakan autentikasi berbasis aplikasi Github yang didukung oleh sumber daya. CodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri Ini menginstal aplikasi AWS CodeStar Connections ke GitHub organisasi Anda sehingga Anda dapat mengelola akses di GitHub.
- Tidak disarankan: Tindakan GitHub (melalui OAuth aplikasi) menggunakan OAuth token untuk mengautentikasi GitHub dan menggunakan webhook terpisah untuk mendeteksi perubahan. Ini bukan lagi metode yang direkomendasikan.

> **Note**
>
> Koneksi tidak tersedia di Asia Pasifik (Hong Kong), Asia Pasifik (Hyderabad), Asia Pasifik (Jakarta), Asia Pasifik (Melbourne), Asia Pasifik (Osaka), Afrika (Cape Town), Timur Tengah (Bahrain), Timur Tengah (UEA), Eropa (Spanyol), Eropa (Zurich), Israel (Tel Aviv), atau (AS-Barat) Wilayah. AWS GovCloud Untuk mereferensikan tindakan lain yang tersedia, lihatIntegrasi produk dan layanan dengan CodePipeline. Untuk pertimbangan dengan tindakan ini di Wilayah Eropa (Milan), lihat catatan diCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.

Ada beberapa keuntungan penting menggunakan tindakan GitHub (melalui GitHub Aplikasi) alih-alih tindakan GitHub (melalui OAuth aplikasi):

- Dengan koneksi, CodePipeline tidak lagi memerlukan OAuth aplikasi atau token akses pribadi untuk mengakses repositori Anda. Saat membuat sambungan, Anda menginstal GitHub Aplikasi yang mengelola autentikasi ke GitHub repositori dan mengizinkan izin di tingkat organisasi. Anda harus mengotorisasi OAuth token sebagai pengguna untuk mengakses repositori. Untuk informasi selengkapnya tentang GitHub akses OAuth berbasis yang berbeda dengan GitHub akses berbasis

Aplikasi, lihat. https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps

- Ketika Anda mengelola GitHub (melalui GitHub Aplikasi) tindakan di CLI atau CloudFormation, Anda tidak lagi harus menyimpan token akses pribadi Anda sebagai rahasia di Secrets Manager. Anda tidak lagi harus mereferensikan rahasia yang disimpan secara dinamis dalam konfigurasi CodePipeline tindakan Anda. Anda malah menambahkan ARN koneksi ke konfigurasi tindakan Anda. Untuk contoh konfigurasi tindakan, lihatCodeStarSourceConnection untuk Bitbucket Cloud, GitHub Enterprise Server GitHub, GitLab .com, dan tindakan yang dikelola GitLab sendiri.
- Saat membuat sumber daya koneksi untuk digunakan dengan tindakan GitHub (melalui GitHub Aplikasi) CodePipeline, Anda dapat menggunakan sumber daya koneksi yang sama untuk mengaitkan layanan lain yang didukung, seperti CodeGuru Peninjau, dengan repositori Anda.
- Di Github (melalui GitHub Aplikasi), Anda dapat mengkloning repositori untuk mengakses metadata git dalam CodeBuild tindakan selanjutnya, sementara di Github (melalui OAuth aplikasi) Anda hanya dapat mengunduh sumbernya.
- Administrator menginstal aplikasi untuk repositori organisasi Anda. Anda tidak lagi harus melacak OAuth token yang bergantung pada individu yang membuat token.

Semua aplikasi yang diinstal ke organisasi memiliki akses ke kumpulan repositori yang sama. Untuk mengubah siapa yang dapat mengakses setiap repositori, ubah kebijakan IAM untuk setiap koneksi. Sebagai contoh, lihat Contoh: Kebijakan cakupan bawah untuk menggunakan koneksi dengan repositori tertentu.

Anda dapat menggunakan langkah-langkah dalam topik ini untuk menghapus tindakan sumber GitHub (melalui OAuth aplikasi) dan menambahkan tindakan sumber GitHub (melalui GitHub Aplikasi) dari CodePipeline konsol.

Topik

- Langkah 1: Ganti GitHub tindakan Anda (melalui OAuth aplikasi)
- Langkah 2: Buat koneksi ke GitHub
- Langkah 3: Simpan tindakan GitHub sumber Anda

Langkah 1: Ganti GitHub tindakan Anda (melalui OAuth aplikasi)

Gunakan halaman edit pipeline untuk mengganti tindakan (melalui OAuth aplikasi) Anda dengan GitHub tindakan GitHub (melalui GitHub Aplikasi).

Untuk mengganti GitHub tindakan Anda (melalui OAuth aplikasi)

1. Masuk ke CodePipeline konsol.
2. Pilih pipeline Anda, dan pilih Edit. Pilih Edit tahap pada tahap sumber Anda. Sebuah pesan menampilkan yang merekomendasikan Anda memperbarui tindakan Anda.
3. Di penyedia Tindakan, pilih GitHub (melalui GitHub Aplikasi).
4. Lakukan salah satu tindakan berikut:

   - Di bawah Koneksi, jika Anda belum membuat sambungan ke penyedia Anda, pilih Connect to GitHub. Lanjutkan ke Langkah 2: Buat koneksi ke GitHub.
   - Di bawah Koneksi, jika Anda telah membuat koneksi ke penyedia Anda, pilih koneksi. Lanjutkan ke Langkah 3: Simpan Tindakan Sumber untuk Koneksi Anda.

Langkah 2: Buat koneksi ke GitHub

Setelah Anda memilih untuk membuat koneksi, GitHub halaman Connect to ditampilkan.

Untuk membuat koneksi ke GitHub

1. Di bawah pengaturan GitHub koneksi, nama koneksi Anda ditampilkan di Nama koneksi.

   Di bawah GitHub Aplikasi, pilih penginstalan aplikasi atau pilih Instal aplikasi baru untuk membuatnya.

   > **Note**
   >
   > Anda menginstal satu aplikasi untuk semua koneksi Anda ke penyedia tertentu. Jika Anda telah menginstal GitHub aplikasi, pilih dan lewati langkah ini.

2. Jika halaman otorisasi untuk GitHub ditampilkan, masuk dengan kredensional Anda dan kemudian pilih untuk melanjutkan.
3. Di halaman penginstalan aplikasi, pesan menunjukkan bahwa AWS CodeStar aplikasi mencoba terhubung ke GitHub akun Anda.

   > **Note**
   >
   > Anda hanya menginstal aplikasi sekali untuk setiap GitHub akun. Jika sebelumnya Anda menginstal aplikasi, Anda dapat memilih Konfigurasi untuk melanjutkan ke halaman

modifikasi untuk instalasi aplikasi Anda, atau Anda dapat menggunakan tombol kembali untuk kembali ke konsol.

4. Pada AWS CodeStar halaman Instal, pilih Instal.
5. Pada GitHub halaman Connect to, ID koneksi untuk instalasi baru Anda ditampilkan. Pilih Hubungkan.

Langkah 3: Simpan tindakan GitHub sumber Anda

Selesaikan pembaruan Anda di halaman Edit tindakan untuk menyimpan tindakan sumber baru Anda.

Untuk menyimpan tindakan GitHub sumber Anda

1. Di Repositori, masukkan nama repositori pihak ketiga Anda. Di Branch, masukkan cabang tempat Anda ingin pipeline mendeteksi perubahan sumber.

   > **Note**
   >
   > Di Repository, ketik `owner-name/repository-name` seperti yang ditunjukkan dalam contoh ini:
   >
   > ```
   > my-account/my-repository
   > ```

2. Dalam format artefak Output, pilih format untuk artefak Anda.

   - Untuk menyimpan artefak keluaran dari GitHub tindakan menggunakan metode default, pilih CodePipeline default. Tindakan mengakses file dari GitHub repositori dan menyimpan artefak dalam file ZIP di toko artefak pipa.
   - Untuk menyimpan file JSON yang berisi referensi URL ke repositori sehingga tindakan hilir dapat melakukan perintah Git secara langsung, pilih klon Penuh. Opsi ini hanya dapat digunakan oleh tindakan CodeBuild hilir.

     Jika Anda memilih opsi ini, Anda perlu memperbarui izin untuk peran layanan CodeBuild proyek Anda seperti yang ditunjukkan pada[Tambahkan CodeBuild GitClone izin untuk koneksi ke Bitbucket,, GitHub Enterprise Server GitHub, atau .com GitLab](). Untuk tutorial

yang menunjukkan cara menggunakan opsi klon Penuh, lihatTutorial: Gunakan klon lengkap dengan sumber GitHub pipa.

3. Di artefak Output, Anda dapat mempertahankan nama artefak keluaran untuk tindakan ini, seperti. `SourceArtifact` Pilih Selesai untuk menutup halaman tindakan Edit.
4. Pilih Selesai untuk menutup halaman pengeditan panggung. Pilih Simpan untuk menutup halaman pengeditan pipeline.

Kuota di AWS CodePipeline

CodePipeline memiliki kuota untuk jumlah jaringan pipa, tahapan, tindakan, dan webhook yang dapat dimiliki AWS akun di setiap Wilayah. AWS

Kuota ini berlaku per Wilayah dan dapat ditingkatkan. Diperlukan waktu hingga dua minggu untuk memproses permintaan peningkatan kuota.

Sumber Daya	Default
Lama waktu sebelum waktu tindakan habis (Ini adalah batas waktu yang dapat dikonfigurasi. Lihat tabel berikut untuk batas waktu yang tidak dapat dikonfigurasi)	CloudFormation tindakan penyebaran: 3 hari CodeDeploy dan tindakan penyebaran CodeDeploy ECS (biru/hijau): 5 hari AWS Lambda memanggil tindakan: 24 jam **Note** Saat tindakan sedang berjalan, CodePipeline secara berkala menghubungi Lambda untuk status. Fungsi Lambda membalas dengan status di mana eksekusi tindakan berhasil, gagal, atau sedang berlangsung. Jika fungsi Lambda tidak mengirim balasan setelah 20 menit, waktu tindakan akan habis. Jika, selama 20 menit, fungsi Lambda telah menjawab bahwa tindakan masih berlangsung, CodePipeline restart timer 20 menit dan coba lagi. Jika tidak berhasil setelah 24 jam, CodePipeline atur status tindakan pemanggilan Lambda menjadi gagal.

Sumber Daya	Default
	Lambda memiliki batas waktu terpisah untuk fungsi Lambda yang tidak terkait dengan batas waktu tindakan. CodePipeline Tindakan penyebaran Amazon S3:90 menit **Note** Jika unggahan ke S3 habis selama penerapan file ZIP besar, tindakan gagal dengan kesalahan batas waktu. Coba pecahkan file ZIP menjadi file yang lebih kecil. Tindakan persetujuan manual batas waktu default tingkat akun: 7 hari **Note** Batas waktu default untuk tindakan persetujuan manual dapat diganti untuk tindakan tertentu dalam pipeline, dan dapat dikonfigurasi hingga 86400 menit (60 hari) dengan nilai minimum 5 menit. Untuk informasi selengkapnya, lihat ActionDeclaration di dalam Referensi API CodePipeline . Saat dikonfigurasi, batas waktu ini diterapkan untuk tindakan. Jika tidak, default level akun digunakan.

Sumber Daya	Default
	Semua tindakan lainnya: 1 jam **Note** Batas waktu tindakan penerapan Amazon ECS dapat dikonfigu rasi hingga satu jam (batas waktu default).
Jumlah maksimum total jaringan pipa per Wilayah dalam suatu akun AWS	1000 **Note** Pipeline yang dikonfigurasi untuk pemungutan suara atau deteksi perubahan berbasis peristiwa dihitung dalam kuota ini.
Jumlah maksimum jaringan pipa yang disetel ke polling untuk perubahan sumber, per Wilayah AWS	300 **Note** Kuota ini tetap dan tidak dapat diubah. Jika Anda mencapai batas untuk saluran pemungutan suara, Anda masih dapat mengonfig urasi saluran pipa tambahan yang menggunakan deteksi perubahan berbasis peristiwa. Untuk informasi lebih lanjut, lihatTindakan sumber dan metode deteksi perubahan. [1]
Jumlah maksimum webhook per Wilayah dalam akun AWS	300

Sumber Daya	Default
Jumlah tindakan kustom per Wilayah dalam AWS akun	50

[1] Berdasarkan penyedia sumber Anda, gunakan petunjuk berikut untuk memperbarui jalur pemungutan suara Anda agar menggunakan deteksi perubahan berbasis peristiwa:

- Untuk memperbarui tindakan CodeCommit sumber, lihatMigrasikan saluran pemungutan suara (atau sumber Amazon CodeCommit S3) (konsol).
- Untuk memperbarui tindakan sumber Amazon S3, lihat. Migrasikan saluran pemungutan suara (atau sumber Amazon CodeCommit S3) (konsol)
- Untuk memperbarui tindakan GitHub sumber, lihatMigrasikan pipeline polling ke webhooks ((melalui OAuth aplikasi) tindakan sumber) GitHub (konsol).

Kuota berikut AWS CodePipeline berlaku untuk ketersediaan Wilayah, batasan penamaan, dan ukuran artefak yang diizinkan. Kuota ini tetap dan tidak dapat diubah.

Untuk daftar titik akhir CodePipeline layanan untuk setiap Wilayah, lihat AWS CodePipeline titik akhir dan kuota di Referensi Umum.AWS

Untuk informasi tentang persyaratan struktural, lihatCodePipeline referensi struktur pipa.

AWS Wilayah tempat Anda dapat membuat pipeline	AS Timur (Ohio) AS Timur (Virginia Utara) AS Barat (California Utara) AS Barat (Oregon) Canada (Central) Eropa (Frankfurt) Eropa (Zürich) * Israel (Tel Aviv)

	Eropa (Irlandia)
	Eropa (London)
	Eropa (Milan) *
	Eropa (Paris)
	Eropa (Spanyol)
	Eropa (Stockholm)
	Afrika (Cape Town) *
	Asia Pasifik (Hong Kong) *
	Asia Pasifik (Hyderabad)
	Asia Pasifik (Mumbai)
	Asia Pacific (Tokyo)
	Asia Pasifik (Seoul)
	Asia Pasifik (Osaka)
	Asia Pasifik (Singapura)
	Asia Pasifik (Sydney)
	Asia Pasifik (Jakarta)
	Asia Pasifik (Melbourne)
	Amerika Selatan (Sao Paulo)
	Timur Tengah (Bahrain) *
	Timur Tengah (UEA)
	AWS GovCloud (AS-Barat)
	AWS GovCloud (AS-Timur)

Karakter diizinkan dalam nama tindakan	Nama aksi tidak boleh melebihi 100 karakter. Karakter yang diizinkan meliputi: Huruf kecil a melaluiz, inklusif. Huruf besar A melaluiZ, inklusif. Angka 0 melalui9, inklusif. Karakter khusus . (titik), @ (pada tanda), - (tanda minus), dan _ (garis bawah). Karakter lain, seperti spasi, tidak diperbolehkan.
Karakter diizinkan dalam tipe tindakan	Nama tipe tindakan tidak boleh melebihi 25 karakter. Karakter yang diizinkan meliputi: Huruf kecil a sampai z, inklusif. Huruf besar A sampai Z, inklusif. Angka 0 hingga 9, inklusif. Karakter khusus. (periode), @ (pada tanda), - (tanda minus), dan _ (garis bawah). Karakter lain, seperti spasi, tidak diperbolehkan.

Karakter diizinkan dalam nama artefak	Nama artifak tidak boleh melebihi 100 karakter. Karakter yang diizinkan meliputi: Huruf kecil a melaluiz, inklusif. Huruf besar A melaluiZ, inklusif. Angka 0 melalui9, inklusif. Karakter khusus - (tanda minus), dan _ (garis bawah). Karakter lain, seperti spasi, tidak diperbole hkan.
Karakter diizinkan dalam nama aksi mitra	Nama tindakan mitra harus mengikuti konvensi dan batasan penamaan yang sama dengan nama tindakan lainnya. CodePipeline Secara khusus, mereka tidak dapat melebihi 100 karakter. Karakter yang diizinkan meliputi: Huruf kecil a sampai z, inklusif. Huruf besar A sampai Z, inklusif. Angka 0 hingga 9, inklusif. Karakter khusus. (periode), @ (pada tanda), - (tanda minus), dan _ (garis bawah). Karakter lain, seperti spasi, tidak diperbole hkan.

Karakter diizinkan dalam nama pipeline	Nama pipa tidak boleh melebihi 100 karakter. Karakter yang diizinkan meliputi: Huruf kecil a sampai z, inklusif. Huruf besar A sampai Z, inklusif. Angka 0 hingga 9, inklusif. Karakter khusus. (periode), @ (pada tanda), - (tanda minus), dan _ (garis bawah). Karakter lain, seperti spasi, tidak diperbole hkan.
Karakter diizinkan dalam nama panggung	Nama panggung tidak boleh melebihi 100 karakter. Karakter yang diizinkan meliputi: Huruf kecil a sampai z, inklusif. Huruf besar A sampai Z, inklusif. Angka 0 hingga 9, inklusif. Karakter khusus. (periode), @ (pada tanda), - (tanda minus), dan _ (garis bawah). Karakter lain, seperti spasi, tidak diperbole hkan.
Lama waktu sebelum waktu tindakan habis	CodeBuild membangun aksi: 36 jam Tindakan uji: 8 jam Tindakan kustom: 24 jam Tindakan panggilan Step Functions: 7 hari Membangun batas waktu untuk tindakan Commands: 55 menit

Panjang maksimum tombol konfigurasi tindakan (misalnya, tombol CodeBuild konfigurasi adalah`ProjectName` ,`PrimarySource` , dan`EnvironmentVariables`)	50 karakter
Panjang maksimum nilai konfigurasi tindakan (misalnya, nilai `RepositoryName` konfigurasi dalam konfigurasi CodeCommit tindakan harus kurang dari 1000 karakter: `"RepositoryName": "my-repo-name-less-than-1000-characters"`)	1000 karakter
Jumlah maksimum tindakan per pipa	1000
Jumlah maksimum eksekusi pipa bersamaan per pipeline (ANTRIAN | mode PARALEL)	50
Jumlah maksimum eksekusi aksi bersamaan per eksekusi pipa mode PARALLEL	5
Jumlah maksimum file untuk objek Amazon S3	100.000
Jumlah maksimum aksi paralel dalam satu tahap	100
Jumlah maksimum tindakan berurutan dalam satu tahap	100

Ukuran maksimum artefak dalam tahap sumber	Artefak yang disimpan dalam ember Amazon S3:7 GB Artefak yang disimpan dalam CodeCommit atau GitHub repositori: 1 GB Pengecualian: Jika Anda menggunakan AWS Elastic Beanstalk untuk menyebarkan aplikasi, ukuran artefak maksimum selalu 512 MB. Pengecualian: Jika Anda menggunakan CloudFormation untuk menyebarkan aplikasi, ukuran artefak maksimum selalu 256 MB. Pengecualian: Jika Anda menggunak an `CodeDeployToECS` tindakan untuk menyebarkan aplikasi, ukuran artefak maksimum selalu 3 MB.
Ukuran maksimum file JSON definisi gambar yang digunakan dalam jaringan pipa yang menyebarkan wadah dan gambar Amazon ECS	100 KB
Ukuran maksimum artefak input untuk tindakan CloudFormation	270 MB
Ukuran maksimum artefak input untuk aksi `CodeDeployToECS`	3 MB
Ukuran maksimum artefak input untuk aksi `Step Functions`	Tindakan Step Functions berjalan di Lambda, dan karena itu memiliki kuota ukuran artefak yang sama dengan kuota ukuran artefak untuk fungsi Lambda. Untuk informasi selengkapnya, lihat Kuota Lambda di Panduan Pengembang Lambda.

Ukuran maksimum objek JSON yang dapat disimpan di properti `ParameterOverrides`	Untuk tindakan CodePipeline penerapan dengan CloudFormation sebagai penyedia, `ParameterOverrides` properti digunakan untuk menyimpan objek JSON yang menentukan nilai untuk file konfigurasi CloudFormation template. Ada batas ukuran maksimum 1 kilobyte untuk objek JSON yang dapat disimpan dalam properti `ParameterOverrides` .
Jumlah tindakan dalam satu tahap	Minimal 1, maksimal 50
Jumlah artefak yang diizinkan untuk setiap tindakan	Untuk jumlah artefak input dan output yang diizinkan untuk setiap tindakan, lihat Artefak input dan output yang valid untuk setiap jenis tindakan
Jumlah bulan dimana informasi riwayat eksekusi pipeline dipertahankan	12
Jumlah tahapan dalam pipa	Minimal 2, maksimal 50
Tag pipa	Tag peka terhadap huruf besar dan kecil. Maksimum 50 per sumber daya.

Nama kunci tag pipa	Setiap kombinasi huruf Unicode, angka, spasi, dan karakter diperbolehkan dalam UTF-8 dengan panjang antara 1 dan 128 karakter. Karakter yang diizinkan adalah + - =. _:/@ Nama kunci tag harus unik, dan setiap kunci hanya dapat memiliki satu nilai. Sebuah tanda tidak boleh: • Mulailah dengan AWS: • hanya terdiri dari spasi • diakhiri dengan spasi • berisi emoji atau salah satu karakter berikut:? ^ * [\ ~! # $% & * () > < | "'
Nilai tag pipa	Setiap kombinasi huruf Unicode, angka, spasi, dan karakter diizinkan dalam UTF-8 dengan panjang antara 1 dan 256 karakter. Karakter yang diizinkan adalah + - =. _:/@ Sebuah kunci hanya dapat memiliki satu nilai, tetapi banyak kunci dapat memiliki nilai yang sama. Sebuah tanda tidak boleh: • Mulailah dengan AWS: • hanya terdiri dari spasi • diakhiri dengan spasi • berisi emoji atau salah satu karakter berikut:? ^ * [\ ~! # $% & * () > < | "'

Pemicu	Ada maksimum 50 pemicu dalam definisi pipa di seluruh `push` dan `pull request` konfigura si. Ada maksimal tiga filter per pemicu dorong dan pemicu permintaan tarik. **Note** Duplikat untuk filter dalam array tipe acara yang sama tidak diperbolehkan. Anda dapat menambahkan hingga 8 termasuk dan 8 mengecualikan pola, cabang, dan jalur file untuk setiap jenis acara (push, pull request). Karakter yang diizinkan dalam nilai pola mencakup semua jenis karakter. Untuk pola include dan exclude, ada panjang maksimum 255 karakter. Untuk nama tag, ada panjang maksimum 255 karakter. Ukuran maksimum `triggers` array tidak boleh melebihi 200 KB

Filter pemicu	Jalur file: • Jumlah pola: Anda dapat menambahkan hingga 8 termasuk dan 8 mengecualikan pola. • Ukuran pola: Masing-masing menyertakan atau mengecualikan ukuran pola dapat mencapai 255 karakter. Cabang: • Jumlah pola: Anda dapat menambahkan hingga 8 termasuk dan 8 mengecualikan pola. • Ukuran pola: Masing-masing menyertakan atau mengecualikan ukuran pola dapat mencapai 255 karakter. Permintaan tarik: Cabang: • Jumlah pola: Anda dapat menambahkan hingga 8 termasuk dan 8 mengecualikan pola. • Ukuran pola: Masing-masing menyertakan atau mengecualikan ukuran pola dapat mencapai 255 karakter.

Keunikan nama	Dalam satu AWS akun, setiap pipeline yang Anda buat di AWS Wilayah harus memiliki nama yang unik. Anda dapat menggunakan kembali nama untuk saluran pipa di Wilayah yang berbeda AWS . Nama panggung harus unik di dalam pipa. Nama tindakan harus unik dalam satu panggung.
Kuota untuk variabel keluaran dan ruang nama	Ada batas ukuran maksimum 122880 byte untuk semua variabel keluaran yang digabungkan untuk tindakan tertentu. Ada batas ukuran maksimum 100 KB untuk total konfigurasi tindakan yang diselesaikan untuk tindakan tertentu. Nama variabel keluaran peka huruf besar/kecil. Ruang nama peka huruf besar/kecil. Karakter yang diizinkan meliputi: • Huruf kecil a sampai z, inklusif. • Huruf besar A sampai Z, inklusif. • Angka 0 hingga 9, inklusif. • Karakter khusus ^ (tanda sisipan), @ (di tanda), - (tanda minus), _ (garis bawah), [(braket kiri),] (braket kanan), * (tanda bintang), $ (tanda dolar). Karakter lain, seperti spasi, tidak diperbolehkan.

Kuota untuk variabel di tingkat pipa	Ada maksimum 50 variabel tingkat pipa per pipa. Nama variabel untuk variabel pada tingkat pipa harus: • Panjang maksimum 128 karakter • Huruf kecil a sampai z, inklusif. • Huruf besar A sampai Z, inklusif. • Angka 0 hingga 9, inklusif. • Karakter khusus `@\-_]+` Karakter lain, seperti spasi, tidak diperbolehkan. Untuk nilai variabel, ada panjang maksimum 1000 karakter Untuk nilai variabel, semua karakter diperbolehkan. Untuk deskripsi variabel, ada panjang maksimum 200 karakter.

* Anda harus mengaktifkan Wilayah ini sebelum Anda dapat menggunakannya.

Lampiran A: GitHub (melalui OAuth aplikasi) tindakan sumber

Lampiran ini memberikan informasi tentang (melalui OAuth aplikasi) GitHub tindakan di. CodePipeline

> **Note**
>
> Meskipun kami tidak merekomendasikan penggunaan tindakan GitHub (melalui OAuth aplikasi), saluran pipa yang ada dengan tindakan GitHub (melalui OAuth aplikasi) akan terus berfungsi tanpa dampak apa pun. Untuk pipeline dengan tindakan GitHub (melalui OAuth aplikasi), CodePipeline gunakan token OAuth berbasis untuk terhubung ke GitHub repositori Anda. Sebaliknya, GitHub tindakan (melalui GitHub Aplikasi) menggunakan sumber daya koneksi untuk mengaitkan sumber AWS daya ke GitHub repositori Anda. Sumber daya koneksi menggunakan token berbasis aplikasi untuk terhubung. Untuk informasi selengkapnya tentang memperbarui pipeline Anda ke GitHub tindakan yang disarankan yang menggunakan koneksi, lihatPerbarui tindakan sumber GitHub (melalui OAuth aplikasi) ke tindakan sumber GitHub (melalui GitHub Aplikasi). Untuk informasi selengkapnya tentang GitHub akses OAuth berbasis berbeda dengan GitHub akses berbasis aplikasi, lihat. https://docs.github.com/en/developers/apps/differences-between-github-apps-and-oauth-apps

Untuk berintegrasi dengan GitHub, CodePipeline gunakan GitHub OAuth aplikasi untuk pipeline Anda. CodePipelinemenggunakan webhook untuk mengelola deteksi perubahan untuk pipeline Anda dengan tindakan sumber GitHub (melalui OAuth aplikasi).

> **Note**
>
> Saat mengonfigurasi tindakan sumber GitHub (melalui GitHub Aplikasi) CloudFormation, Anda tidak menyertakan informasi GitHub token apa pun atau menambahkan sumber daya webhook. Anda mengonfigurasi sumber daya koneksi seperti yang ditunjukkan AWS::CodeStarConnections::Connectiondalam Panduan CloudFormation Pengguna.

Referensi ini berisi bagian berikut untuk tindakan GitHub (melalui OAuth aplikasi):

- Untuk informasi tentang cara menambahkan aksi sumber GitHub (melalui OAuth aplikasi) dan webhook ke pipeline, lihatMenambahkan tindakan sumber GitHub (melalui OAuth aplikasi).

- Untuk informasi tentang parameter konfigurasi dan contoh YAML/JSON cuplikan untuk tindakan sumber GitHub (melalui OAuth aplikasi), lihat. GitHub (melalui OAuth aplikasi) referensi tindakan sumber

> **Important**
>
> Saat membuat CodePipeline webhook, jangan gunakan kredensialnya sendiri atau gunakan kembali token rahasia yang sama di beberapa webhook. Untuk keamanan optimal, buat token rahasia unik untuk setiap webhook yang Anda buat. Token rahasia adalah string arbitrer yang Anda berikan, yang GitHub digunakan untuk menghitung dan menandatangani muatan webhook yang dikirim ke CodePipeline, untuk melindungi integritas dan keaslian muatan webhook. Menggunakan kredensi Anda sendiri atau menggunakan kembali token yang sama di beberapa webhook dapat menyebabkan kerentanan keamanan.

> **Note**
>
> Jika token rahasia diberikan, itu akan dihapus dalam tanggapan.

Topik

- Menambahkan tindakan sumber GitHub (melalui OAuth aplikasi)
- GitHub (melalui OAuth aplikasi) referensi tindakan sumber

Menambahkan tindakan sumber GitHub (melalui OAuth aplikasi)

Anda menambahkan GitHub (melalui OAuth aplikasi) tindakan sumber ke CodePipeline oleh:

- Menggunakan CodePipeline konsol Create pipeline wizard (Buat pipeline khusus (konsol)) atau Edit halaman tindakan untuk memilih opsi GitHubpenyedia. Konsol membuat webhook yang memulai pipeline Anda saat sumbernya berubah.
- Menggunakan CLI untuk menambahkan konfigurasi tindakan untuk `GitHub` tindakan dan membuat sumber daya tambahan sebagai berikut:
  - Menggunakan `GitHub` contoh konfigurasi tindakan GitHub (melalui OAuth aplikasi) referensi tindakan sumber untuk membuat tindakan seperti yang ditunjukkan padaBuat pipeline (CLI).

- Menonaktifkan pemeriksaan berkala dan membuat deteksi perubahan secara manual, karena metode deteksi perubahan default memulai pipeline dengan melakukan polling pada sumbernya. Anda memigrasikan pipeline polling Anda ke webhooks untuk tindakan GitHub (melalui OAuth aplikasi).

GitHub (melalui OAuth aplikasi) referensi tindakan sumber

> **Note**
>
> Meskipun kami tidak merekomendasikan penggunaan tindakan GitHub (melalui OAuth aplikasi), saluran pipa yang ada dengan tindakan GitHub (melalui OAuth aplikasi) akan terus berfungsi tanpa dampak apa pun. Untuk pipeline dengan aksi sumber GitHub GitHub (melalui OAuth aplikasi), CodePipeline gunakan token OAuth berbasis untuk terhubung ke GitHub repositori Anda. Sebaliknya, GitHub tindakan baru (melalui GitHub Aplikasi) menggunakan sumber daya koneksi untuk mengaitkan sumber AWS daya ke GitHub repositori Anda. Sumber daya koneksi menggunakan token berbasis aplikasi untuk terhubung. Untuk informasi selengkapnya tentang memperbarui pipeline Anda ke GitHub tindakan yang disarankan yang menggunakan koneksi, lihatPerbarui tindakan sumber GitHub (melalui OAuth aplikasi) ke tindakan sumber GitHub (melalui GitHub Aplikasi).

Memicu pipeline saat komit baru dibuat pada GitHub repositori dan cabang yang dikonfigurasi.

Untuk berintegrasi GitHub, CodePipeline gunakan OAuth aplikasi atau token akses pribadi untuk pipeline Anda. Jika Anda menggunakan konsol untuk membuat atau mengedit pipeline, CodePipeline buat GitHub webhook yang memulai pipeline saat terjadi perubahan di repositori.

Anda harus sudah membuat GitHub akun dan repositori sebelum menghubungkan pipeline melalui tindakan. GitHub

Jika Anda ingin membatasi akses CodePipeline ke repositori, buat GitHub akun dan berikan akses akun hanya ke repositori yang ingin Anda integrasikan. CodePipeline Gunakan akun tersebut saat Anda mengonfigurasi CodePipeline untuk menggunakan GitHub repositori untuk tahapan sumber di pipeline.

Untuk informasi selengkapnya, lihat dokumentasi GitHub pengembang di GitHub situs web.

Topik

- Tipe tindakan
- Parameter konfigurasi
- Artefak masukan
- Artefak keluaran
- Variabel keluaran
- Deklarasi tindakan (GitHub contoh)
- Menghubungkan ke GitHub (OAuth)
- Lihat juga

Tipe tindakan

- Kategori: `Source`
- Pemilik: `ThirdParty`
- Penyedia: `GitHub`
- Versi: `1`

Parameter konfigurasi

Pemilik

Wajib: Ya

Nama GitHub pengguna atau organisasi yang memiliki GitHub repositori.

Repo

Wajib: Ya

Nama repositori tempat perubahan sumber akan dideteksi.

Cabang

Wajib: Ya

Nama cabang tempat perubahan sumber harus dideteksi.

OAuthToken

Wajib: Ya

Merupakan token GitHub otentikasi yang memungkinkan CodePipeline untuk melakukan operasi pada GitHub repositori Anda. Entri selalu ditampilkan sebagai topeng dari empat tanda bintang. Ini merupakan salah satu dari nilai-nilai berikut:

- Saat Anda menggunakan konsol untuk membuat pipeline, CodePipeline gunakan OAuth token untuk mendaftarkan GitHub koneksi.
- Ketika Anda menggunakan AWS CLI untuk membuat pipeline, Anda dapat meneruskan token akses GitHub pribadi Anda di bidang ini. Ganti tanda bintang (****) dengan token akses pribadi Anda yang disalin dari. GitHub Saat Anda menjalankan `get-pipeline` untuk melihat konfigurasi tindakan, topeng empat tanda bintang ditampilkan untuk nilai ini.
- Saat Anda menggunakan CloudFormation template untuk membuat pipeline, Anda harus terlebih dahulu menyimpan token sebagai rahasia AWS Secrets Manager. Anda menyertakan nilai untuk bidang ini sebagai referensi dinamis ke rahasia yang disimpan di Secrets Manager, seperti`{{resolve:secretsmanager:MyGitHubSecret:SecretString:token}}`.

Untuk informasi selengkapnya tentang GitHub cakupan, lihat Referensi API GitHub Pengembang di GitHub situs web.

PollForSourceChanges

Wajib: Tidak

`PollForSourceChanges`mengontrol apakah CodePipeline polling GitHub repositori untuk perubahan sumber. Kami menyarankan Anda menggunakan webhooks untuk mendeteksi perubahan sumber sebagai gantinya. Untuk informasi selengkapnya tentang mengonfigurasi webhook, lihat atau. Migrasikan jalur pemungutan suara ke tindakan sumber webhook (GitHub (melalui OAuth aplikasi)) (CLI) Perbarui saluran pipa untuk acara push (GitHub (melalui OAuth aplikasi) tindakan sumber) (CloudFormation templat)

> **Important**
>
> Jika Anda bermaksud mengonfigurasi webhook, Anda harus mengatur `PollForSourceChanges false` untuk menghindari eksekusi pipeline duplikat.

Nilai yang valid untuk parameter ini:

- `True`: Jika disetel, CodePipeline polling repositori Anda untuk perubahan sumber.

> **Note**
>
> Jika Anda menghilangkan`PollForSourceChanges`, CodePipeline default untuk polling repositori Anda untuk perubahan sumber. Perilaku ini sama seperti jika `PollForSourceChanges` diatur ke`true`.

- `False`: Jika disetel, CodePipeline tidak melakukan polling repositori Anda untuk perubahan sumber. Gunakan pengaturan ini jika Anda bermaksud mengonfigurasi webhook untuk mendeteksi perubahan sumber.

Artefak masukan

- Jumlah artefak: `0`
- Deskripsi: Artefak masukan tidak berlaku untuk jenis tindakan ini.

Artefak keluaran

- Jumlah artefak: `1`
- Deskripsi: Artefak keluaran dari tindakan ini adalah file ZIP yang berisi konten repositori dan cabang yang dikonfigurasi pada komit yang ditentukan sebagai revisi sumber untuk eksekusi pipeline. Artefak yang dihasilkan dari repositori adalah artefak keluaran untuk tindakan tersebut. GitHub ID komit kode sumber ditampilkan CodePipeline sebagai revisi sumber untuk eksekusi pipeline yang dipicu.

Variabel keluaran

Ketika dikonfigurasi, tindakan ini menghasilkan variabel yang dapat direferensikan oleh konfigurasi tindakan tindakan hilir dalam pipeline. Tindakan ini menghasilkan variabel yang dapat dilihat sebagai variabel keluaran, bahkan jika tindakan tidak memiliki namespace. Anda mengonfigurasi tindakan dengan namespace untuk membuat variabel-variabel tersebut tersedia untuk konfigurasi tindakan hilir.

Untuk informasi lebih lanjut tentang variabel di CodePipeline, lihatReferensi variabel.

CommitId

ID GitHub komit yang memicu eksekusi pipeline. Komit IDs adalah SHA penuh dari komit.

CommitMessage

Pesan deskripsi, jika ada, terkait dengan komit yang memicu eksekusi pipeline.

CommitUrl

Alamat URL untuk komit yang memicu pipeline.

RepositoryName

Nama GitHub repositori tempat komit yang memicu pipeline dibuat.

BranchName

Nama cabang untuk GitHub repositori tempat perubahan sumber dilakukan.

AuthorDate

Tanggal ketika komit ditulis, dalam format stempel waktu.

CommitterDate

Tanggal ketika komit dilakukan, dalam format stempel waktu.

Deklarasi tindakan (GitHub contoh)

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"
        }
    ]
},
```

Menghubungkan ke GitHub (OAuth)

Pertama kali Anda menggunakan konsol untuk menambahkan GitHub repositori ke pipeline, Anda diminta untuk mengotorisasi CodePipeline akses ke repositori Anda. Token membutuhkan GitHub cakupan berikut:

- Ruang `repo` lingkup, yang digunakan untuk kontrol penuh untuk membaca dan menarik artefak dari repositori publik dan swasta ke dalam pipa.
- Ruang `admin:repo_hook` lingkup, yang digunakan untuk kontrol penuh kait repositori.

Ketika Anda menggunakan CLI atau CloudFormation template, Anda harus memberikan nilai untuk token akses pribadi yang telah Anda buat. GitHub

Lihat juga

Sumber daya terkait berikut dapat membantu Anda saat Anda bekerja dengan tindakan ini.

- Referensi sumber daya untuk Panduan AWS CloudFormation Pengguna AWS::CodePipeline::Webhook — Ini termasuk definisi bidang, contoh, dan cuplikan untuk sumber daya di. CloudFormation
- Referensi sumber daya untuk Panduan AWS CloudFormation Pengguna AWS::CodeStar::GitHubRepository — Ini termasuk definisi bidang, contoh, dan cuplikan untuk sumber daya di. CloudFormation
- Tutorial: Membuat pipeline yang membangun dan menguji aplikasi Android Anda AWS Device Farm— Tutorial ini menyediakan contoh file build spec dan contoh aplikasi untuk membuat pipeline dengan GitHub sumber. Ini membangun dan menguji aplikasi Android dengan CodeBuild dan AWS Device Farm.

AWS CodePipeline Riwayat dokumen Panduan Pengguna

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Panduan CodePipeline Pengguna. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

- Versi API: 2015-07-09
- Pembaruan dokumentasi terbaru: 23 Mei 2025

Perubahan	Deskripsi	Tanggal
Support ditambahkan untuk Azure DevOps	Support ditambahkan untuk mengonfigurasi koneksi untuk AWS sumber daya untuk berinteraksi dengan DevOps Azure. Untuk informasi selengkapnya, lihat Koneksi untuk Azure DevOps.	Agustus 5, 2025
Dokumentasi yang diperbarui untuk file spesifikasi penerapan tindakan penerapan EC2	Menambahkan informasi baru tentang penggunaan file spesifikasi penerapan untuk tindakan penerapan EC2 dengan contoh dan opsi konfigurasi yang diperbarui. Lihat halaman referensi tindakan di referensi tindakan Amazon EC2.	16 Mei 2025
Menambahkan dokumentasi untuk tindakan penerapan Lambda	Menambahkan dukungan untuk tindakan penerapan Lambda baru dengan contoh dan opsi konfigurasi yang diperbarui. Lihat halaman referensi tindakan di referensi tindakan AWS Lambda penerapan.	16 Mei 2025

Tambahkan bidang ke tindakan Perintah untuk jenis komputasi, tipe lingkungan, dan tipe variabel lingkungan	Menambahkan informasi tentang bidang baru yang memungkinkan Anda menentukan lingkungan build dan jenis komputasi untuk tindakan Command. Bidang tipe baru untuk variabel lingkungan juga telah ditambahkan untuk tindakan Command. Lihat halaman referensi tindakan di aksi CodePipeline Comands.	8 Mei 2025
Topik baru untuk mendokumentasikan kebijakan peran layanan default	Menambahkan informasi tentang kebijakan peran layanan minimum dan memperbarui tabel tautan ke izin peran layanan tambahan untuk setiap tindakan. CodePipeline Lihat halaman referensi aturan baru di kebijakan peran CodePipeline layanan.	Maret 26, 2025
Tindakan `CodePipeline` pemanggilan baru	Menambahkan informasi tentang tindakan `CodePipeline` pemanggilan baru. Lihat halaman referensi tindakan di referensi tindakan CodePipeline pemanggilan.	Maret 14, 2025

`CodeBuild` Aturan baru	Menambahkan informasi tentang `CodeBuild` aturan baru yang tersedia untuk menjalankan proyek build sebagai aturan untuk kondisi tahap. Lihat halaman referensi aturan baru di referensi CodeBuild aturan.	Maret 14, 2025
Berbagi koneksi di AWS CodeConnections	Untuk saluran pipa dengan koneksi menggunakan AWS CodeConnections, Anda dapat menggunakan koneksi yang dikonfigurasi untuk berbagi antara Akun AWS. Anda dapat mengonfigurasi koneksi bersama di AWS Resource Access Manager. Untuk informasi selengkapnya, lihat Menggunakan koneksi yang dibagikan dengan yang lain Akun AWS.	Maret 6, 2025

EventBridge masukan entri transformasi untuk revisi sumber	Untuk pipeline dengan CodeCommit, Amazon ECR, dan sumber Amazon S3, Anda dapat EventBridge menggunakan entri transformasi masukan untuk revisi sumber, yang berasal dari variabel peristiwa sumber untuk kunci objek (S3), commit (), atau ID gambar CodeCommit (ECR) Anda. `revisionValue` Lihat tindakan sumber dan sumber EventBridge daya Amazon ECR, Menyambungkan ke tindakan sumber Amazon S3 dengan sumber yang diaktifkan untuk peristiwa, CodeCommit serta tindakan sumber, dan. EventBridge	Maret 3, 2025
Build spec override untuk tindakan AWS CodeBuild	Anda dapat memilih untuk mengganti spesifikasi build untuk CodeBuild tindakan sambil memasukkan perintah secara langsung. Lihat halaman referensi tindakan di AWS CodeBuild build dan test action reference. Lihat juga Membuat pipeline, tahapan, dan tindakan.	Maret 3, 2025

Tindakan EC2 penerapan baru	Menambahkan informasi tentang tindakan EC2 penerapan baru. Lihat halaman referensi tindakan di referensi tindakan Amazon EC2. Untuk tutorial, lihat Tutorial: Menyebarkan ke instans EC2 dengan. CodePipeline	Februari 21, 2025
Tindakan EKS penerapan baru	Menambahkan informasi tentang tindakan EKS penerapan baru. Lihat halaman referensi tindakan di tindakan penerapan EKS. Untuk tutorial, lihat Tutorial: Menyebarkan ke Amazon EKS dengan CodePipeline.	Februari 20, 2025
CloudWatch Metrik dan dimensi baru untuk CodePipeline	Menambahkan metrik untuk pipeline dalam CloudWatch metrik. Lihat CodePipeline CloudWatch metrik.	Februari 13, 2025
Commands Aturan baru	Menambahkan informasi tentang Commands aturan baru yang tersedia untuk menjalankan perintah shell sebagai aturan untuk kondisi tahap. Lihat halaman referensi aturan baru di Referensi aturan perintah.	Desember 17, 2024

Contoh yang diperluas dan informasi referensi untuk pemicu	Menambahkan deskripsi untuk filter peristiwa permintaan tarik oleh penyedia dalam peristiwa permintaan Tarik untuk pemicu oleh penyedia. Menambahkan contoh yang diperluas untuk pemicu dengan informasi lebih rinci tentang termasuk dan pengecualian dalam peristiwa push dan peristiwa permintaan tarik dalam peristiwa permintaan Tarik untuk pemicu oleh penyedia. Menambahkan informasi referensi JSON tambahan tentang termasuk dan tidak termasuk dalam Pemicu.	Desember 17, 2024
Tindakan baru dalam katalog tindakan	Anda sekarang dapat menggunakan `ECRBuildAndPublish` dan `InspectorScan` tindakan. Untuk informasi selengkapnya, lihat halaman referensi ECRBuildAndPublishdan InspectorScantindakan.	November 22, 2024
Konfigurasi otomatis baru untuk percobaan ulang tahap pada kegagalan	Anda dapat mengonfigurasi tahapan untuk secara otomatis mencoba kembali tahap yang gagal atau tindakan yang gagal di panggung. Untuk informasi selengkapnya, lihat Mengkonfigurasi tahapan untuk mencoba ulang otomatis pada kegagalan.	Oktober 15, 2024

`Skip`Hasil baru untuk kondisi Entri	Menambahkan informasi tentang `Skip` hasil yang tersedia untuk kondisi masuk. Anda dapat menggunakan `LambdaInvoke` aturan `VariableCheck` dan dengan konfigurasi ini. Lihat langkah-langkah di Membuat kondisi Entri dengan hasil Lewati. Untuk daftar pertimbangan kondisi dengan hasil Lewati, lihat Pertimbangan untuk hasil yang dikonfigurasi untuk kondisi tahap.	Oktober 15, 2024
Langkah konsol baru untuk membuat pipeline dari template statis	Di CodePipeline konsol, Anda dapat menggunakan wizard pembuatan pipeline baru untuk memilih dari beberapa templat statis untuk menghasilkan sumber daya pipeline CloudFormation. Untuk informasi selengkapnya, lihat Membuat pipeline dari templat statis.	Oktober 9, 2024

CommandsAksi baru	Menambahkan informasi tentang Commands tindakan baru yang tersedia untuk menjalankan perintah shell sebagai tindakan di pipeline Anda. Lihat halaman referensi tindakan baru di Tindakan Perintah dan izin peran layanan di Tambahkan izin ke peran CodePipeline layanan. Untuk tutorial, lihat Tutorial: Buat pipeline yang menjalankan perintah dengan komputasi.	Oktober 3, 2024
VariableCheck Aturan baru untuk kondisi	Menambahkan informasi tentang VariableCheck aturan untuk kondisi panggung. Lihat halaman referensi aturan baru di VariableCheck. Untuk tutorial, lihat Tutorial: Membuat aturan pemeriksaan variabel untuk pipeline sebagai kondisi entri.	September 27, 2024
Pembaruan koneksi untuk GitHub	Menambahkan informasi tentang penggunaan token akses GitHub pengguna dengan koneksi ke GitHub (tindakan GitHub V2). Token akses pengguna digunakan dengan CodeBuild proyek. Lihat GitHub koneksi dan Tutorial: Gunakan klon lengkap dengan sumber GitHub pipa.	September 16, 2024

Pembaruan untuk merestrukturisasi referensi JSON pipa dan daftar isi panduan	Panduan ini telah direstrukturisasi, termasuk beberapa perubahan judul bagian, untuk meningkatkan kegunaan bagian referensi dan tugas.	Agustus 16, 2024
Perbarui ke bidang token rahasia dalam respons `PutWebhook` dan `ListWebhooks` tindakan	Bidang token rahasia untuk `ListWebhooks` tindakan `PutWebhook` dan telah diperbarui. Jika token rahasia diberikan, itu akan dihapus dalam tanggapan. Lihat catatan yang ditambahkan ke Lampiran A: tindakan sumber GitHub versi 1. Untuk pembaruan terkait di Panduan CodePipeline API, lihat PutWebhookdan ListWebhooks.	Agustus 6, 2024
Menambahkan konten baru untuk kondisi dan aturan panggung	Anda sekarang dapat mengonfigurasi kondisi panggung dan aturan untuk saluran pipa tipe V2. Lihat Konsep, Bagaimana cara kerja kondisi panggung? , dan Konfigurasikan kondisi untuk sebuah panggung. Bab referensi untuk aturan telah ditambahkan yang menyediakan informasi referensi. Lihat referensi CodePipeline aturan.	Juli 30, 2024

Menambahkan informasi referensi baru untuk jenis pipa dan fitur terkait	Skrip analisis baru tersedia untuk mengevaluasi biaya untuk pindah ke pipa tipe V2. Lihat Jenis pipa apa yang tepat untuk saya? . Tabel referensi telah ditambahk an yang menyediakan tautan di semua dokumenta si CodePipeline layanan berdasarkan fitur. Lihat referensi CodePipeline fitur.	Juli 11, 2024
Pembaruan untuk tindakan S3 sumber untuk menambahkan opsi baru untuk penggantian sumber	Opsi baru untuk penggantian sumber bernama `S3_OBJECT_KEY` tersedia untuk tindakan S3 sumber. `AllowOverrideForS3ObjectKey` Parameter baru telah ditambahkan untuk aksi sumber S3. Lihat halaman referensi tindakan sumber Amazon S3 dan Mulai pipeline dengan penggantian revisi sumber.	Juni 7, 2024
Pembaruan untuk tindakan S3 sumber untuk menambahkan variabel keluaran baru	Variabel keluaran baru bernama `BucketName` dan `ObjectKey` tersedia untuk aksi `S3` sumber. Lihat halaman referensi tindakan sumber Amazon S3.	Juni 5, 2024

Pembaruan untuk `CloudFormationStackSet` dan `CloudFormationStackInstances` tindakan	`CallAsParameter` ditambahkan untuk `CloudFormationStackInstances` tindakan `CloudFormationStackSet` dan. Lihat halaman referensi tindakan.	2 Mei 2024
Support untuk rollback tingkat tahap	Anda dapat secara manual atau otomatis memutar kembali tahap ke eksekusi pipeline sebelumnya yang berhasil untuk tahap tersebut. Lihat Mengkonfigurasi rollback panggung dan Konsep.	April 26, 2024
Pembaruan ketersediaan Wilayah untuk StackSets tindakan Step Functions	Tindakan StackSets dan Step Functions sekarang tersedia di semua Wilayah CodePipeline jika tersedia. Lihat referensi CloudFormation StackSets tindakan dan referensi tindakan AWS Step Functions.	Maret 27, 2024
Pembaruan kebijakan terkelola	Kebijakan AWS terkelola `AWSCodePipeline_FullAccess` telah diperbarui. Lihat kebijakan AWS terkelola untuk AWS CodePipeline.	Maret 15, 2024
Support untuk batas waktu yang dapat dikonfigurasi untuk tindakan persetujuan manual	Informasi kuota ditambahkan untuk bidang batas waktu baru yang dapat dikonfigurasi untuk tindakan persetujuan manual. Untuk informasi lebih lanjut, lihat Kuota .	Februari 15, 2024

Support untuk pemfilteran pemicu berdasarkan cabang dan jalur file	Support ditambahkan untuk konfigurasi pemicu yang memungkinkan pemfilter an pada status permintaan tarik, cabang, dan jalur file untuk pipeline tipe V2. Untuk informasi selengkapnya, lihat Memfilter pemicu pada permintaan push atau pull kode, Pemicu, dan Filter pada cabang fitur untuk memulai pipeline, dan Kuota.	Februari 8, 2024
Support untuk mode eksekusi pipeline baru	Support ditambahkan untuk mode eksekusi pipa PARALLEL dan QUEUED. Untuk informasi selengkap nya, lihat Mengatur mode eksekusi pipeline, Cara eksekusi diproses dalam mode ANTRIAN, Cara eksekusi diproses dalam modePARALLEL, dan Kuota.	Februari 8, 2024

Pembaruan ke halaman konsol untuk melihat detail tindakan, meninjau tindakan persetujuan manual, dan halaman pipeline daftar	Pembaruan konsol didokumentasikan untuk tombol Lihat detail baru dan kotak dialog, dialog persetujuan manual baru, dan kolom baru untuk eksekusi terbaru di halaman pipeline daftar. Untuk informasi selengkapnya, lihat Melihat pipeline (konsol), Melihat detail tindakan dalam pipeline, dan Mengelola tindakan persetujuan di pipeline.	10 Januari 2024
Support untuk GitLab self-managed	Support ditambahkan untuk mengkonfigurasi koneksi untuk AWS sumber daya untuk berinteraksi dengan GitLab self-managed. Untuk informasi selengkapnya, lihat Koneksi untuk GitLab dikelola sendiri.	28 Desember 2023
Pembaruan untuk `CloudFormationStackSet` dan `CloudFormationStackInstances` tindakan	`ConcurrencyMode` Parameter ditambahkan untuk `CloudFormationStackInstances` tindakan `CloudFormationStackSet` dan. Lihat halaman referensi tindakan.	Desember 19, 2023
Pembaruan untuk parameter AWS Device Farm tindakan di CodePipeline	Parameter untuk AWS Device Farm tindakan di CodePipeline telah diperbarui. Untuk informasi selengkapnya, lihat referensi AWS Device Farm tindakan.	18 Desember 2023

Support ditambahkan untuk pesan kesalahan rinci untuk CloudFormation tindakan di CodePipeline	CloudFormation pesan kesalahan tindakan sekarang dapat memunculkan detail tentang sumber daya yang gagal. Untuk informasi selengkapnya, lihat referensi CloudFormation tindakan.	15 Desember 2023
Pembaruan untuk memulai pipeline dengan penggantian revisi sumber di CodePipeline	Anda sekarang dapat memulai pipeline dengan revisi sumber tertentu. Untuk informasi selengkapnya, lihat Memulai pipeline dengan penggantian revisi sumber.	17 November 2023
Wilayah baru yang didukung	CodePipeline Sekarang tersedia di Asia Pasifik (Hyderabad), Asia Pasifik (Jakarta), Asia Pasifik (Melbourne), Asia Pasifik (Osaka), Timur Tengah (UEA), Eropa (Spanyol), dan Israel (Tel Aviv). Topik referensi bucket placeholder Acara dan topik Layanan AWS titik akhir telah diperbarui.	13 November 2023
Pembaruan untuk bidang acara di Amazon EventBridge	Anda sekarang dapat melihat bidang acara yang diperbaru i di Amazon EventBridge. Untuk informasi selengkapnya, lihat Memantau CodePipeline peristiwa.	9 November 2023

Pembaruan untuk pipeline tipe V2 baru, pemicu pada tag Git, dan variabel pipeline di CodePipeline	Anda sekarang dapat memilih jenis pipa di CodePipeline. Untuk pipeline tipe V2, Anda sekarang dapat menggunakan konfigurasi pemicu untuk memulai pipeline Anda pada tag Git. Dengan pipeline tipe V2, Anda juga dapat menggunakan variabel pada level pipeline untuk meneruskan parameter input untuk eksekusi pipeline. Untuk informasi selengkapnya, lihat Variabel, Tutorial: Menggunakan variabel tingkat pipeline, dan Tutorial: Gunakan tag Git untuk memulai pipeline Anda. Untuk informasi selengkapnya tentang jenis pipa, lihat Jenis pipa.	24 Oktober 2023
CodePipeline memungkinkan mencoba kembali semua tindakan dalam tahap yang gagal	Untuk tahap yang gagal CodePipeline, Anda dapat mencoba lagi panggung tanpa menjalankan kembali pipa. Anda melakukan ini baik dengan mencoba kembali tindakan yang gagal dalam satu tahap atau dengan mencoba kembali semua tindakan di tahap mulai dari tindakan pertama di panggung. Untuk informasi selengkapnya, lihat.	17 Oktober 2023

Support untuk GitLab grup	Support ditambahkan untuk mengkonfigurasi koneksi untuk AWS sumber daya untuk berinteraksi dengan GitLab grup. Untuk informasi selengkapnya, lihat GitLab koneksi.	15 September 2023
CodePipeline mendukung koneksi GitLab ke.com	Anda dapat menggunakan koneksi untuk mengonfig urasi AWS sumber daya untuk berinteraksi GitLab dengan.co m. Anda juga dapat memilih opsi klon lengkap untuk menggunakan perintah Git dan metadata untuk tindakan hilir. Untuk informasi selengkapnya, lihat GitLab koneksi dan topik referensi struktur CodeStarS ourceConnection tindakan.	10 Agustus 2023
Perbarui ke `CloudFormationStackInstances` tindakan	`RegionConcurrencyType` Parameter ditambahkan untuk `CloudFormationStackInstances` tindakan. Lihat halaman referensi tindakan untuk `CloudFormationStackInstances` tindakan.	8 Agustus 2023
Perbarui ke `CloudFormationStackSet` tindakan	`RegionConcurrencyType` Parameter ditambahkan untuk `CloudFormationStackSet` tindakan. Lihat halaman referensi tindakan untuk `CloudFormationStackSet` tindakan.	Juli 24, 2023

Pembaruan kebijakan terkelola	Kebijakan AWS terkelola `AWSCodePipeline_FullAccess` telah diperbarui. Lihat kebijakan AWS terkelola untuk AWS CodePipeline.	Juni 21, 2023
Pembaruan prosedur migrasi untuk jaringan pemungutan suara	Prosedur untuk memigrasi (memperbarui) jalur pemungutan suara untuk menggunakan deteksi perubahan berbasis peristiwa telah diperbarui dengan langkah-langkah untuk saluran pipa yang menggunakan bucket Amazon S3 yang diaktifkan untuk notifikasi. EventBridge Untuk informasi selengkapnya, lihat Memigrasi jalur pemungutan suara untuk menggunakan deteksi perubahan berbasis peristiwa.	12 Juni 2023
Pembaruan kebijakan terkelola	Kebijakan AWS terkelola `AWSCodePipeline_FullAccess` dan `AWSCodePipeline_ReadOnlyAccess` telah diperbarui dengan izin tambahan. Untuk informasi selengkapnya, lihat AWS CodePipeline pembaruan kebijakan AWS terkelola.	16 Mei 2023

Pembaruan kebijakan terkelola	Kebijakan AWS terkelola `AWSCodePipelineFullAccess` dan `AWSCodePipelineReadOnlyAccess` tidak digunakan lagi. Gunakan `AWSCodePipeline_FullAccess` dan `AWSCodePipeline_ReadOnlyAccess` kebijakan. Lihat AWS CodePipeline pembaruan kebijakan AWS terkelola.	17 November 2022
Pembaruan untuk prosedur yang menggunakan CloudTrail	Semua prosedur konsol, contoh perintah CLI, dan CloudFormation cuplikan sampel dan templat untuk pipeline dengan sumber S3 telah diperbarui dengan opsi untuk memilih Tulis dan memilih false untuk peristiwa Manajemen di. CloudTrail Lihat sampel yang diperbarui di Memulai pipeline, Tutorial: Buat pipeline dengan CloudFormation, Edit pipeline untuk menggunakan peristiwa push, dan Perbarui pipeline polling.	27 April 2022

Integrasi baru yang didukung dengan Snyk	Anda dapat menggunakan tindakan pemanggilan Snyk CodePipeline untuk mengotomatiskan pemindaian keamanan untuk kode sumber terbuka Anda. Untuk informasi lebih lanjut, lihat referensi tindakan Snyk dan Integrasi.	10 Juni 2021
Wilayah Eropa baru yang didukung (Milan)	CodePipeline sekarang tersedia di Eropa (Milan). Topik Limits dan topik Layanan AWS endpoint telah diperbarui.	27 Januari 2021
Deteksi perubahan dapat dimatikan untuk tindakan sumber dengan koneksi	Anda dapat menggunakan CLI atau SDK untuk memperbarui tindakan `CodeStarSourceConnection` sumber untuk mematikan deteksi perubahan otomatis untuk repositori sumber. Topik referensi struktur CodeStarSourceConnection tindakan telah diperbarui dengan deskripsi untuk `DetectChanges` parameter.	8 Januari 2021

CodePipeline sekarang mendukung CloudFormation StackSets tindakan penerapan	Tutorial baru, Tutorial: Buat pipeline yang digunakan CloudFormation StackSets sebagai penyedia penyebaran, menyediakan langkah-langkah yang harus digunakan CloudFormation StackSets untuk membuat dan memperbarui kumpulan tumpukan dan instance tumpukan dengan pipeline Anda. Topik referensi struktur CloudFormation StackSets tindakan juga telah ditambahkan.	30 Desember 2020
Wilayah baru yang didukung Asia Pasifik (Hong Kong)	CodePipeline sekarang tersedia di Asia Pasifik (Hong Kong). Topik Limits dan topik Layanan AWS endpoint telah diperbarui.	22 Desember 2020
Lihat pola EventBridge acara yang diperbarui di CodePipeline	Pola dan status peristiwa yang diperbarui untuk peristiwa pipeline, panggung, dan tingkat tindakan telah ditambahkan ke CodePipeline acara Pemantauan.	21 Desember 2020
Lihat eksekusi pipa masuk di CodePipeline	Anda dapat menggunakan konsol atau CLI untuk melihat eksekusi masuk. Untuk informasi selengkapnya, lihat Melihat eksekusi masuk (konsol) dan Melihat status eksekusi masuk (CLI).	16 November 2020

Tindakan CodeCommit sumber di CodePipeline mendukung opsi klon lengkap	Saat Anda menggunakan tindakan CodeCommit sumber, Anda dapat memilih opsi klon lengkap untuk menggunakan perintah Git dan metadata untuk tindakan hilir. CodeBuild Untuk informasi selengkapnya, lihat referensi CodeCommit tindakan dan Tutorial: Gunakan klon lengkap dengan sumber CodeCommit pipeline.	11 November 2020
CodePipeline mendukung koneksi ke GitHub dan GitHub Enterprise Server	Anda dapat menggunakan koneksi untuk mengonfigurasi AWS sumber daya untuk berinteraksi GitHub, GitHub Enterprise Cloud, dan GitHub Enterprise Server. Anda juga dapat memilih opsi klon lengkap untuk menggunakan perintah Git dan metadata untuk tindakan hilir. Untuk informasi selengkapnya, lihat GitHub koneksi, koneksi Server GitHub Perusahaan, dan Tutorial: Gunakan klon lengkap dengan sumber GitHub pipeline. Jika Anda memiliki pipeline yang sudah ada dengan tindakan GitHub sumber, lihat Memperbarui tindakan sumber GitHub (melalui OAuth aplikasi) ke tindakan sumber GitHub (melalui GitHub Aplikasi).	30 September 2020

CodeBuild Tindakan ini mendukung pengaktifan batch build di AWS CodePipeline	Untuk CodeBuild tindakan dalam pipeline, Anda dapat mengaktifkan build batch untuk menjalankan beberapa build dalam satu eksekusi. Untuk informasi selengkap nya, lihat referensi struktur CodeBuild tindakan dan Membuat pipeline (konsol).	30 Juli 2020
AWS CodePipeline sekarang mendukung AWS AppConfig tindakan penerapan	Tutorial baru, Tutorial: Buat pipeline yang digunakan AWS AppConfig sebagai penyedia penyebaran, menyediak an langkah-langkah untuk digunakan AWS AppConfig untuk menyebarkan file konfigurasi dengan pipeline Anda. Topik referensi struktur AWS AppConfig tindakan juga telah ditambahkan.	25 Juni 2020
AWS CodePipeline sekarang mendukung Amazon VPC di AWS GovCloud (AS-Barat)	Anda sekarang dapat terhubung langsung ke AWS CodePipeline melalui titik akhir VPC Amazon pribadi di AWS GovCloud (AS-Barat). Untuk informasi selengkapnya, lihat Menggunakan CodePipel ine dengan Amazon Virtual Private Cloud.	2 Juni 2020

AWS CodePipeline sekarang mendukung tindakan AWS Step Functions pemanggilan	Anda sekarang dapat membuat pipeline CodePipeline yang digunakan AWS Step Functions sebagai penyedia tindakan pemanggilan. Tutorial baru, Tutorial: AWS Step Functions Gunakan tindakan pemanggilan dalam pipeline, menyediakan langkah-langkah untuk memulai eksekusi mesin status dari pipeline Anda. Topik Referensi Struktur AWS Step Functions Aksi juga telah ditambahkan.	28 Mei 2020
Lihat, daftar, dan perbarui koneksi	Anda dapat membuat daftar, menghapus, dan memperbarui koneksi di konsol. Lihat Daftar koneksi di CodePipeline.	21 Mei 2020
Koneksi mendukung penandaan sumber daya koneksi di CLI	Sumber daya koneksi sekarang mendukung penandaan di AWS CLI. Koneksi sekarang terintegrasi dengan AWS CodeGuru. Lihat Referensi Izin IAM untuk Koneksi.	6 Mei 2020
CodePipeline sekarang tersedia di AWS GovCloud (AS-Barat)	Anda sekarang dapat menggunakan CodePipeline di AWS GovCloud (AS-Barat). Untuk informasi lebih lanjut, lihat Kuota .	8 April 2020

Topik kuota menunjukkan kuota CodePipeline layanan mana yang dapat dikonfigurasi	Topik CodePipeline kuota telah diformat ulang. Dokumentasi menunjukkan kuota layanan mana yang dapat dikonfigurasi dan kuota mana yang tidak dapat dikonfigurasi. Lihat Kuota di AWS CodePipeline.	12 Maret 2020
Batas waktu tindakan penerapan Amazon ECS dapat dikonfigurasi	Batas waktu tindakan penerapan Amazon ECS dapat dikonfigurasi hingga satu jam (batas waktu default). Lihat Kuota di AWS CodePipeline.	5 Februari 2020
Topik baru menjelaskan bagaimana Anda dapat menghentikan eksekusi pipeline	Anda dapat menghentikan eksekusi pipeline di CodePipeline. Anda dapat menentukan bahwa eksekusi berhenti setelah tindakan yang sedang berlangsung diizinkan untuk diselesaikan, atau Anda dapat menentukan untuk menghentikan eksekusi segera dan meninggalkan tindakan yang sedang berlangsung. Lihat Bagaimana eksekusi pipeline dihentikan dan Menghentikan eksekusi pipeline di CodePipeline.	21 Januari 2020

CodePipeline mendukung koneksi	Anda dapat menggunakan koneksi untuk mengkonfigurasi AWS sumber daya untuk berinteraksi dengan repositori kode eksternal. Setiap koneksi adalah sumber daya yang dapat digunakan oleh layanan seperti CodePipeline untuk terhubung ke repositori pihak ketiga, seperti Bitbucket Cloud. Untuk informasi selengkapnya, lihat Bekerja dengan koneksi di CodePipeline.	18 Desember 2019
Topik keamanan, otentikasi, dan kontrol akses yang diperbarui	Informasi keamanan, otentikasi, dan kontrol akses untuk CodePipeline telah diatur ke dalam babak Keamanan baru. Untuk informasi selengkapnya, lihat Keamanan.	17 Desember 2019
Topik baru menjelaskan bagaimana Anda dapat menggunakan variabel di pipeline Anda	Anda sekarang dapat mengonfigurasi ruang nama untuk tindakan dan menghasilkan variabel setiap kali eksekusi tindakan selesai. Anda dapat mengatur tindakan hilir untuk mereferensikan namespace dan variabel ini. Lihat Bekerja dengan variabel dan Variabel.	14 November 2019

Topik baru menjelaskan cara kerja eksekusi pipeline, mengapa tahapan dikunci selama eksekusi, dan kapan eksekusi pipeline digantikan	Sejumlah topik telah ditambahkan ke bagian Selamat Datang untuk menjelaskan cara kerja eksekusi pipeline, termasuk mengapa tahapan dikunci selama eksekusi dan apa yang terjadi ketika eksekusi pipeline digantikan. Topik-topik ini mencakup daftar konsep, contoh DevOps alur kerja, dan rekomendasi tentang bagaimana pipeline harus disusun. Topik berikut telah ditambahkan: Istilah saluran pipa, contoh DevOps pipa, dan Cara kerja eksekusi pipeline.	11 November 2019
CodePipeline mendukung aturan pemberitahuan	Anda sekarang dapat menggunakan aturan notifikasi untuk memberi tahu pengguna tentang perubahan penting dalam saluran pipa. Untuk informasi selengkapnya, lihat Membuat aturan notifikasi.	5 November 2019

CodeBuild variabel lingkungan tersedia di CodePipeline	Anda dapat menyetel variabel CodeBuild lingkungan dalam tindakan CodeBuild build untuk pipeline Anda. Anda dapat menggunakan konsol atau CLI untuk menambahkan `EnvironmentVariables` parameter ke struktur pipa. Topik Buat pipeline (konsol) telah diperbarui. Contoh konfigurasi tindakan dalam referensi tindakan untuk juga CodeBuildtelah diperbarui.	14 Oktober 2019
Wilayah Baru	CodePipeline sekarang tersedia di Eropa (Stockholm). Topik Limits dan topik Layanan AWS endpoint telah diperbarui.	Selasa, 05 September 2019
Tentukan kontrol kalengan ACLs dan cache untuk tindakan penerapan Amazon S3	Sekarang Anda dapat menentukan opsi kontrol ACL dan cache yang dikalengkan saat membuat tindakan penerapan Amazon S3. CodePipeline Topik berikut telah diperbarui: Buat pipeline (konsol), referensi struktur CodePipeline Pipeline, dan Tutorial: Buat pipeline yang menggunakan Amazon S3 sebagai penyedia penerapan.	27 Juni 2019

Anda sekarang dapat menambahkan tag ke sumber daya di AWS CodePipeline	Anda sekarang dapat menggunakan penandaan untuk melacak dan mengelola AWS CodePipeline sumber daya seperti pipeline, tindakan kustom, dan webhook. Topik baru berikut telah ditambahkan: Menandai sumber daya, Menggunakan tag untuk mengontrol akses ke CodePipeline sumber daya, Menandai pipeline CodePipeline, Menandai tindakan kustom CodePipeline, dan Menandai webhook. CodePipeline Topik berikut telah diperbarui untuk menunjukkan cara menggunakan CLI untuk menandai sumber daya: Buat pipeline (CLI), Buat tindakan khusus (CLI), dan Buat webhook untuk sumber. GitHub	15 Mei 2019

Anda sekarang dapat melihat riwayat eksekusi tindakan di AWS CodePipeline	Anda sekarang dapat melihat detail tentang eksekusi sebelumnya dari semua tindakan dalam pipeline. Detail ini mencakup waktu mulai dan akhir, durasi, ID eksekusi tindakan, status, detail lokasi artefak input dan keluaran, dan detail sumber daya eksternal. Detail pipa Lihat dan topik riwayat telah diperbarui untuk mencermin kan dukungan ini.	20 Maret 2019
AWS CodePipeline sekarang mendukung penerbitan aplikasi ke AWS Serverless Application Repository	Anda sekarang dapat membuat pipeline CodePipel ine yang menerbitkan aplikasi tanpa server Anda ke file. AWS Serverless Applicati on Repository Tutorial baru, Tutorial: Publikasikan aplikasi ke AWS Serverless Applicati on Repository, menyediak an langkah-langkah untuk membuat dan mengonfigurasi pipeline untuk terus mengirimk an aplikasi tanpa server Anda ke file. AWS Serverless Application Repository	8 Maret 2019

AWS CodePipeline sekarang mendukung tindakan lintas wilayah di konsol	Anda sekarang dapat mengelola tindakan lintas wilayah di AWS CodePipeline konsol. Menambahkan tindakan Lintas wilayah telah diperbarui dengan langkah-langkah untuk menambah, mengedit, atau menghapus tindakan yang ada di AWS Wilayah berbeda dari pipeline Anda. Topik referensi Create a pipeline, Edit CodePipeline pipeline, dan pipeline structure telah diperbarui.	14 Februari 2019
AWS CodePipeline sekarang mendukung penerapan Amazon S3	Sekarang Anda dapat membuat pipeline CodePipeline yang menggunakan Amazon S3 sebagai penyedia tindakan penerapan. Tutorial baru, Tutorial: Buat pipeline yang menggunakan Amazon S3 sebagai penyedia penerapan, menyediakan langkah-langkah untuk menerapkan file sampel ke bucket Amazon S3 Anda. CodePipeline Topik referensi struktur CodePipeline pipa juga telah diperbarui.	16 Januari 2019

AWS CodePipeline sekarang mendukung penerapan Alexa Skills Kit	Anda sekarang dapat menggunakan CodePipeline dan Alexa Skills Kit untuk penyebaran keterampilan Alexa yang berkelanjutan. Tutorial baru, Tutorial: Buat pipeline yang menyebarkan keterampilan Amazon Alexa, berisi langkah-langkah untuk membuat kredensil yang memungkinkan AWS CodePipeline untuk terhubung ke akun pengembang Alexa Skills Kit Anda dan kemudian membuat pipeline yang menyebarkan keterampilan sampel. Topik referensi struktur CodePipeline pipa telah diperbarui.	19 Desember 2018
AWS CodePipeline sekarang mendukung titik akhir Amazon VPC yang didukung oleh AWS PrivateLink	Anda sekarang dapat terhubung langsung AWS CodePipeline melalui titik akhir pribadi di VPC Anda, menjaga semua lalu lintas di dalam VPC dan jaringan Anda. AWS Untuk informasi selengkapnya, lihat Menggunakan CodePipeline dengan Amazon Virtual Private Cloud.	6 Desember 2018

AWS CodePipeline sekarang mendukung tindakan sumber Amazon ECR dan tindakan ECS-to-CodeDeploy penerapan	Anda sekarang dapat menggunakan CodePipeline dan CodeDeploy dengan Amazon ECR dan Amazon ECS untuk penyebaran berkelanjutan aplikasi berbasis kontainer. Tutorial baru, Buat pipeline dengan sumber dan ECS-to-CodeDeploy penerapan Amazon ECR, berisi langkah-langkah untuk menggunakan konsol untuk membuat pipeline yang menyebarkan aplikasi kontainer yang disimpan dalam repositori gambar ke cluster Amazon ECS dengan perutean lalu lintas. CodeDeploy Topik referensi Buat CodePipeline pipa dan struktur pipa telah diperbarui.	27 November 2018
AWS CodePipeline sekarang mendukung tindakan lintas wilayah dalam sebuah pipeline	Topik baru, Tambahkan Tindakan Lintas Wilayah, berisi langkah-langkah untuk menggunakan AWS CLI atau CloudFormation menambahkan tindakan yang ada di wilayah berbeda dari pipeline Anda. Topik referensi Create a pipeline, Edit CodePipeline pipeline, dan pipeline structure telah diperbarui.	12 November 2018

AWS CodePipeline sekarang terintegrasi dengan Service Catalog	Sekarang Anda dapat menambahkan Service Catalog sebagai tindakan penerapan ke pipeline Anda. Ini memungkinkan Anda menyiapkan pipeline untuk mempublikasikan pembaruan produk ke Service Catalog saat Anda membuat perubahan di repositori sumber. Topik Integrasi telah diperbarui untuk mencerminkan dukungan ini untuk Service Catalog. Dua tutorial Service Catalog telah ditambahkan ke bagian AWS CodePipeline tutorial.	16 Oktober 2018
AWS CodePipeline Sekarang terintegrasi dengan AWS Device Farm	Anda sekarang dapat menambahkan AWS Device Farm sebagai tindakan pengujian ke pipeline Anda. Ini memungkinkan Anda mengatur pipeline untuk menguji aplikasi seluler. Topik Integrasi telah diperbarui untuk mencerminkan dukungan ini. AWS Device Farm Dua AWS Device Farm tutorial telah ditambahkan ke bagian AWS CodePipeline tutorial.	19 Juli 2018

AWS CodePipeline Pemberitahuan pembaruan Panduan Pengguna sekarang tersedia melalui RSS	Versi HTML Panduan CodePipeline Pengguna sekarang mendukung umpan RSS pembaruan yang didokumentasikan di halaman Riwayat Pembaruan Dokumentasi. Umpan RSS mencakup pembaruan yang dilakukan setelah 30 Juni 2018 dan yang lebih baru. Pembaruan yang diumumkan sebelumnya masih tersedia di halaman Riwayat Pembaruan Dokumentasi. Gunakan tombol RSS di panel menu atas untuk berlangganan feed.	30 Juni 2018

Pembaruan lebih awal

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Panduan CodePipeline Pengguna pada 30 Juni 2018 dan sebelumnya.

Ubah	Deskripsi	Tanggal diubah
Gunakan webhooks untuk mendeteksi perubahan sumber di saluran pipa GitHub	Saat Anda membuat atau mengedit pipeline di konsol, CodePipeline sekarang buat webhook yang mendeteksi perubahan pada repositori GitHub sumber Anda dan kemudian memulai pipeline Anda. Untuk informasi tentang memigrasi pipeline, lihat Mengonfigurasi GitHub Pipeline Anda untuk Menggunakan Webhook untuk Deteksi Perubahan. Untuk informasi selengkapnya, lihat Memulai Eksekusi Pipeline di CodePipeline.	1 Mei 2018
Topik yang diperbarui	Saat Anda membuat atau mengedit pipeline di konsol, CodePipeline sekarang buat aturan CloudWatch Acara	22 Maret 2018

Ubah	Deskripsi	Tanggal diubah
	Amazon dan AWS CloudTrail jejak yang mendeteksi perubahan pada bucket sumber Amazon S3, lalu memulai pipeline Anda. Untuk informasi tentang memigrasi pipeline Anda, lihatTindakan sumber dan metode deteksi perubahan. Tutorial: Buat pipeline sederhana (ember S3)Telah diperbarui untuk menunjukkan bagaimana aturan dan jejak CloudWatch Acara Amazon dibuat saat Anda memilih sumber Amazon S3. Buat pipeline, tahapan, dan tindakandan juga Edit pipa di CodePipeline telah diperbarui. Untuk informasi selengkapnya, lihat Mulai pipa di CodePipeline.	
Topik yang diperbarui	CodePipeline sekarang tersedia di Eropa (Paris). Topik Kuota di AWS CodePipeline telah diperbarui.	21 Februari 2018
Topik yang diperbarui	Anda sekarang dapat menggunakan CodePipeline dan Amazon ECS untuk penyebaran berkelanjutan aplikasi berbasis kontainer. Saat membuat pipeline, Anda dapat memilih Amazon ECS sebagai penyedia penerapan. Perubahan kode di repositori kontrol sumber memicu pipeline Anda untuk membuat image Docker baru, mendorongnya ke registri penampung, lalu menerapkan gambar yang diperbarui ke layanan Amazon ECS. TopikIntegrasi produk dan layanan dengan CodePipeline,Buat pipeline, tahapan, dan tindakan, dan CodePipeline referensi struktur pipa telah diperbarui untuk mencerminkan dukungan ini untuk Amazon ECS.	12 Desember 2017

Ubah	Deskripsi	Tanggal diubah
Topik yang diperbarui	Saat Anda membuat atau mengedit pipeline di konsol, CodePipeline sekarang buat aturan Amazon CloudWatch Events yang mendeteksi perubahan pada CodeCommit repositori Anda dan kemudian secara otomatis memulai pipeline Anda. Untuk informasi tentang memigrasi pipeline yang ada, lihatTindakan sumber dan metode deteksi perubahan. Tutorial: Buat pipeline sederhana (CodeCommit repositori)Telah diperbarui untuk menunjukkan bagaimana aturan dan peran CloudWatch Acara Amazon dibuat saat Anda memilih CodeCommit repositori dan cabang. Buat pipeline, tahapan, dan tindakandan juga Edit pipa di CodePipeline telah diperbarui. Untuk informasi selengkapnya, lihat Mulai pipa di CodePipeline.	11 Oktober 2017
Topik baru dan diperbarui	CodePipeline sekarang menyediakan dukungan bawaan untuk pemberitahuan perubahan status pipeline melalui Amazon CloudWatch Events dan Amazon Simple Notification Service (Amazon SNS). Tutorial baru Tutorial: Menyiapkan aturan CloudWatch Acara untuk menerima pemberitahuan email untuk perubahan status pipeline telah ditambahkan. Untuk informasi selengkapnya, lihat Memantau CodePipeline peristiwa.	8 September 2017

Ubah	Deskripsi	Tanggal diubah
Topik baru dan diperbarui	Anda sekarang dapat menambahkan CodePipeline sebagai target untuk tindakan Amazon CloudWatch Events. Aturan Amazon CloudWatch Events dapat diatur untuk mendeteksi perubahan sumber sehingga pipeline dimulai segera setelah perubahan tersebut terjadi, atau aturan tersebut dapat diatur untuk menjalankan eksekusi pipeline terjadwal. Informasi telah ditambahkan untuk opsi konfigurasi tindakan PollForSourceChanges sumber. Untuk informasi selengkapnya, lihat Mulai pipa di CodePipeline.	5 September 2017
Daerah Baru	CodePipeline Sekarang tersedia di Asia Pasifik (Seoul) dan Asia Pasifik (Mumbai). Kuota di AWS CodePipelineTopik dan topik Wilayah dan Titik Akhir telah diperbarui.	27 Juli 2017
Daerah Baru	CodePipeline sekarang tersedia di AS Barat (California N.), Kanada (Tengah), dan Eropa (London). Kuota di AWS CodePipelineTopik dan topik Wilayah dan Titik Akhir telah diperbarui.	29 Juni 2017
Topik yang diperbarui	Anda sekarang dapat melihat detail tentang eksekusi pipeline sebelumnya, bukan hanya eksekusi terbaru. Detail ini mencakup waktu mulai dan berakhir, durasi, dan ID eksekusi. Rincian tersedia untuk maksimum 100 eksekusi pipa selama periode 12 bulan terakhir. TopikLihat saluran pipa dan detailnya di CodePipeline,CodePipeline referensi izin, dan Kuota di AWS CodePipeline telah diperbarui untuk mencerminkan dukungan ini.	22 Juni 2017
Topik yang diperbarui	Nouvola telah ditambahkan ke daftar tindakan yang tersedia di. Integrasi tindakan uji	18 Mei 2017

Ubah	Deskripsi	Tanggal diubah
Topik yang diperbarui	Di AWS CodePipeline wizard, halaman Langkah 4: Beta telah diganti namanya Langkah 4: Menyebarkan. Nama default tahap yang dibuat oleh langkah ini telah diubah dari “Beta” menjadi “Pementasan”. Banyak topik dan tangkapan layar telah diperbarui untuk mencerminkan perubahan ini.	7 April 2017
Topik yang diperbarui	Anda sekarang dapat menambahkan AWS CodeBuild sebagai tindakan uji ke setiap tahap pipa. Ini memungkinkan Anda untuk lebih mudah menggunakan AWS CodeBuild untuk menjalankan pengujian unit terhadap kode Anda. Sebelum rilis ini, Anda dapat menggunakan AWS CodeBuild untuk menjalankan pengujian unit hanya sebagai bagian dari tindakan build. Tindakan build memerlukan artefak keluaran build, yang biasanya tidak dihasilkan oleh pengujian unit. Topik Integrasi produk dan layanan dengan CodePipelineEdit pipa di CodePipeline,, dan CodePipeline referensi struktur pipa telah diperbarui untuk mencerminkan dukungan ini AWS CodeBuild.	8 Maret 2017

Ubah	Deskripsi	Tanggal diubah
Topik baru dan diperbarui	Daftar isi telah direorganisasi untuk memasukkan bagian untuk saluran pipa, tindakan, dan transisi tahap. Bagian baru telah ditambahkan untuk CodePipeline tutorial. Untuk kegunaan yang lebih baik, Integrasi produk dan layanan dengan CodePipeline telah dibagi menjadi topik yang lebih pendek. Bagian baru, Otorisasi dan Kontrol Akses, memberikan informasi komprehensif tentang penggunaan AWS Identity and Access Management (IAM) dan CodePipeline untuk membantu mengamankan akses ke sumber daya Anda melalui penggunaan kredensil. Kredensi ini memberikan izin yang diperlukan untuk mengakses AWS sumber daya, seperti menempatkan dan mengambil artefak dari bucket Amazon S3 dan mengintegrasikan tumpukan ke dalam saluran pipa Anda. OpsWorks	8 Februari 2017
Wilayah Baru	CodePipeline sekarang tersedia di Asia Pasifik (Tokyo). Kuota di AWS CodePipelineTopik dan topik Wilayah dan Titik Akhir telah diperbarui.	14 Desember 2016
Wilayah Baru	CodePipeline sekarang tersedia di Amerika Selatan (São Paulo). Kuota di AWS CodePipelineTopik dan topik Wilayah dan Titik Akhir telah diperbarui.	Desember 7, 2016

Ubah	Deskripsi	Tanggal diubah
Topik yang diperbarui	Anda sekarang dapat menambahkan AWS CodeBuild sebagai tindakan build ke setiap tahap pipeline. AWS CodeBuild adalah layanan build yang dikelola sepenuhnya di cloud yang mengkompilasi kode sumber Anda, menjalankan pengujian unit, dan menghasilkan artefak yang siap diterapkan. Anda dapat menggunakan proyek build yang sudah ada atau membuatnya di CodePipeline konsol. Output dari proyek build kemudian dapat digunakan sebagai bagian dari pipeline. TopikIntegrasi produk dan layanan dengan CodePipeline,Buat pipeline, tahapan, dan tindakan, Otentikasi dan Kontrol Akses, dan CodePipeline referensi struktur pipa telah diperbarui untuk mencerminkan dukungan ini. AWS CodeBuild Anda sekarang dapat menggunakan CodePipeline dengan CloudFormation dan Model Aplikasi AWS Tanpa Server untuk terus mengirimkan aplikasi tanpa server Anda. Topik Integrasi produk dan layanan dengan CodePipeline telah diperbarui untuk mencerminkan dukungan ini. Integrasi produk dan layanan dengan CodePipelinetelah direorganisasi ke penawaran kelompok AWS dan mitra berdasarkan jenis tindakan.	1 Desember 2016
Wilayah Baru	CodePipeline sekarang tersedia di Eropa (Frankfurt). Kuota di AWS CodePipelineTopik dan topik Wilayah dan Titik Akhir telah diperbarui.	16 Nopember 2016

Ubah	Deskripsi	Tanggal diubah
Topik yang diperbarui	AWS CloudFormation sekarang dapat dipilih sebagai penyedia penerapan di pipeline, memungkinkan Anda untuk mengambil tindakan pada CloudFormation tumpukan dan mengubah set sebagai bagian dari eksekusi pipeline. TopikIntegrasi produk dan layanan dengan CodePipeline,Buat pipeline, tahapan, dan tindakan, Otentikasi dan Kontrol Akses, dan CodePipeline referensi struktur pipa telah diperbarui untuk mencerminkan dukungan ini. AWS CloudFormation	Selasa, 3 November 2016
Wilayah Baru	CodePipeline sekarang tersedia di Wilayah Asia Pasifik (Sydney). Kuota di AWS CodePipelineTopik dan topik Wilayah dan Titik Akhir telah diperbarui.	26 Oktober 2016
Wilayah Baru	CodePipeline sekarang tersedia di Asia Pasifik (Singapura). Kuota di AWS CodePipelineTopik dan topik Wilayah dan Titik Akhir telah diperbarui.	20 Oktober 2016
Wilayah Baru	CodePipeline sekarang tersedia di Wilayah AS Timur (Ohio). Kuota di AWS CodePipelineTopik dan topik Wilayah dan Titik Akhir telah diperbarui.	17 Oktober 2016
Topik yang diperbarui	Buat pipeline, tahapan, dan tindakantelah diperbarui untuk mencerminkan dukungan untuk menampilkan pengenal versi tindakan kustom di penyedia Sumber dan daftar penyedia Build.	September 22, 2016
Topik yang diperbarui	Menambahkan tindakan persetujuan manual ke panggungBagian ini telah diperbarui untuk mencerminkan peningkatan yang memungkinkan pengulas tindakan Persetujuan membuka Menyetujui atau menolak formulir revisi langsung dari pemberitahuan email.	14 September 2016

Ubah	Deskripsi	Tanggal diubah
Topik baru dan diperbarui	Topik baru yang menjelaskan cara melihat detail tentang perubahan kode yang saat ini mengalir melalui pipeline rilis perangkat lunak Anda. Akses cepat ke informasi ini dapat berguna saat meninjau tindakan persetujuan manual atau kegagalan pemecahan masalah dalam pipeline Anda. Bagian baru,Memantau pipeline, menyediakan lokasi pusat untuk semua topik yang terkait dengan pemantauan status dan kemajuan jaringan pipa Anda.	September 08, 2016
Topik baru dan diperbarui	Bagian baru,Menambahkan tindakan persetujuan manual ke panggung, memberikan informasi tentang mengkonfi gurasi dan menggunakan tindakan persetujuan manual di saluran pipa. Topik di bagian ini memberikan informasi konseptual tentang proses persetujuan; instruksi untuk menyiapkan izin IAM yang diperlukan, membuat tindakan persetujuan, dan menyetujui atau menolak tindakan persetujuan; dan sampel data JSON yang dihasilkan saat tindakan persetujuan dicapai dalam pipeline.	Juli 06, 2016
Wilayah Baru	CodePipeline sekarang tersedia di Wilayah Eropa (Irlandia). Kuota di AWS CodePipelineTopik dan topik Wilayah dan titik akhir telah diperbarui.	23 Juni 2016
Topik baru	Topik baru, telah ditambahkan untuk menjelaskan bagaimana mencoba lagi tindakan yang gagal atau sekelompok tindakan gagal paralel secara bertahap.	22 Juni 2016

Ubah	Deskripsi	Tanggal diubah
Topik yang diperbarui	Sejumlah topik, termasukBuat pipeline, tahapan, dan tindakan, Otentikasi dan Kontrol Akses,CodePipeline referensi struktur pipa, danIntegrasi produk dan layanan dengan CodePipeline, telah diperbarui untuk mencermin kan dukungan untuk mengonfigurasi pipeline untuk menyebarkan kode bersama dengan buku masak dan aplikasi Chef kustom yang dibuat. AWS OpsWorks CodePipeline dukungan untuk saat OpsWorks ini tersedia di Wilayah AS Timur (Virginia N.) (us-east-1) saja.	Juni 2, 2016
Topik baru dan diperbarui	Topik baruTutorial: Buat pipeline sederhana (CodeComm it repositori),, telah ditambahkan. Topik ini memberika n contoh panduan yang menunjukkan cara menggunak an CodeCommit repositori dan cabang sebagai lokasi sumber untuk tindakan sumber dalam pipeline. Beberapa topik lain telah diperbarui untuk mencerminkan integrasi ini dengan CodeCommit, termasuk Otentikasi dan Kontrol Akses,, Integrasi produk dan layanan dengan CodePipel ineTutorial: Buat pipeline empat tahap, danPemecahan masalah CodePipeline.	18 April 2016
Topik baru	Topik baruMemanggil AWS Lambda fungsi dalam pipeline di CodePipeline,, telah ditambahkan. Topik ini berisi contoh AWS Lambda fungsi dan langkah-langkah untuk menambahkan fungsi Lambda ke pipeline.	27 Januari 2016
Topik yang diperbarui	Bagian baru telah ditambahkan ke Autentikasi dan Kontrol Akses, Kebijakan Berbasis Sumber Daya.	Januari 22, 2016
Topik baru	Topik baruIntegrasi produk dan layanan dengan CodePipel ine,, telah ditambahkan. Informasi tentang integrasi dengan mitra dan lainnya Layanan AWS telah dipindahkan ke topik ini. Tautan ke blog dan video juga telah ditambahkan.	17 Desember 2015

Ubah	Deskripsi	Tanggal diubah
Topik yang diperbarui	Rincian integrasi dengan Solano CI telah ditambahkan keIntegrasi produk dan layanan dengan CodePipeline.	17 November 2015
Topik yang diperbarui	CodePipeline Plugin untuk Jenkins sekarang tersedia melalui Jenkins Plugin Manager sebagai bagian dari perpustakaan plugin untuk Jenkins. Langkah-langkah untuk menginstal plugin telah diperbarui diTutorial: Buat pipeline empat tahap.	November 9, 2015
Wilayah Baru	CodePipeline sekarang tersedia di Wilayah AS Barat (Oregon). Topik Kuota di AWS CodePipeline telah diperbarui. Tautan telah ditambahkan ke Wilayah dan Titik Akhir.	22 Oktober 2015
Topik baru	Dua topik baru, Konfigurasikan enkripsi sisi server untuk artefak yang disimpan di Amazon S3 CodePipeline danBuat pipeline CodePipeline yang menggunakan sumber daya dari AWS akun lain, telah ditambahkan. Bagian baru telah ditambahkan ke Otentikasi dan Kontrol Akses.	Agustus 25, 2015
Topik yang diperbarui	Buat dan tambahkan tindakan kustom di CodePipelineTopik telah diperbarui untuk mencerminkan perubahan dalam struktur, termasuk `inputArtifactDetails` dan`outputArtifactDetails` .	17 Agustus 2015
Topik yang diperbarui	Pemecahan masalah CodePipelineTopik telah diperbarui dengan langkah-langkah revisi untuk memecahkan masalah dengan peran layanan dan Elastic Beanstalk.	Agustus 11, 2015
Topik yang diperbarui	Topik Autentikasi dan Kontrol Akses telah diperbarui dengan perubahan terbaru pada peran layanan. CodePipeline	Agustus 6, 2015

Ubah	Deskripsi	Tanggal diubah
Topik baru	Sebuah Pemecahan masalah CodePipeline topik telah ditambahkan. Langkah-langkah yang diperbarui telah ditambahkan untuk peran IAM dan Jenkins di. Tutorial: Buat pipeline empat tahap	24 Juli 2015
Pembaruan topik	Langkah-langkah yang diperbarui telah ditambahkan untuk mengunduh file sampel di Tutorial: Buat pipeline sederhana (ember S3) danTutorial: Buat pipeline empat tahap.	22 Juli 2015
Pembaruan topik	Solusi sementara untuk masalah unduhan dengan file sampel telah ditambahkan. Tutorial: Buat pipeline sederhana (ember S3)	17 Juli 2015
Pembaruan topik	Tautan ditambahkan Kuota di AWS CodePipeline untuk menunjukkan informasi tentang batas mana yang dapat diubah.	Juli 15, 2015
Pembaruan topik	Bagian kebijakan terkelola di Otentikasi dan Kontrol Akses telah diperbarui.	10 Juli 2015
Rilis Publik Awal	Ini adalah rilis publik awal dari Panduan CodePipeline Pengguna.	9 Juli 2015

CodePipeline referensi fitur

Bagian ini adalah referensi tingkat tinggi untuk fitur dalam CodePipeline dokumentasi. Untuk gambaran konseptual dari struktur pipa, lihatCodePipeline referensi struktur pipa.

Tabel ini diperbarui secara berkala dengan informasi referensi fitur.

Tanggal pembaruan tabel	Fitur	CodePipeline Panduan Pengguna	CodePipeline Panduan API	CloudFormation Referensi	AWS CDK Referensi - konstruksi L1
Oktober 15, 2024	Coba lagi otomatis pada kegagalan panggung	Konfigurasikan tahapan untuk coba lagi otomatis pada kegagalan	RetryConfiguration		
Oktober 15, 2024	Lewati hasil untuk kondisi beforeEntry	Membuat kondisi Entri dengan Lewati hasil dan `VariableCheck` aturan (konsol)	FailureConditions		
Oktober 03, 2024	Perintah tindakan dengan kategori Compute baru	Referensi tindakan perintah	ActionDeclaration		
Agustus 28, 2024	Jenis beforeEntry	Konfigurasikan kondisi	BeforeEntryConditions	AWS::CodePipeline:	sebelumMasuk

Tanggal pembaruan tabel	Fitur	CodePipeline Panduan Pengguna	CodePipeline Panduan API	CloudFormation Referensi	AWS CDK Referensi - konstruksi L1
	untuk kondisi panggung	untuk sebuah panggung		:PipelineBeforeEntryConditions	
Agustus 28, 2024	Jenis onSuccess untuk kondisi panggung	Konfigurasikan kondisi untuk sebuah panggung	SuccessConditions	AWS::CodePipeline::Pipeline.SuccessKetentuan	OnSuccess
April 26, 2024	Stage rollback dan tipe OnFailure untuk kondisi panggung	Mengkonfigurasi rollback panggung	RollbackStage	OnFailure	OnFailure
Februari 08, 2024	Mode eksekusi PARALEL dan ANTRIAN	Mengatur atau mengubah mode eksekusi pipa	PipelineExecution	Execution Mode	Mode eksekusi
17 November 2023	Pengesampingan sumber	Mulai pipeline dengan penggantian revisi sumber	SourceRevisionOverride	N/A	N/A
24 Oktober 2023	Jenis pipa	Jenis pipa apa yang tepat untuk saya?	PipelineDeclaration	PipelineType	enum PipelineType

Tanggal pembaruan tabel	Fitur	CodePipeline Panduan Pengguna	CodePipeline Panduan API	CloudFormation Referensi	AWS CDK Referensi - konstruksi L1
24 Oktober 2023	Variabel tingkat pipa	Tutorial: Gunakan variabel tingkat pipa	PipelineVariable	AWS::CodePipeline::Pipeline VariableDeclaration	Variabel tingkat pipa
24 Oktober 2023	Memicu dan memfilter jalur file/cabang/permintaan tarik	Otomatiskan saluran pipa awal menggunakan pemicu dan penyaringan Tutorial: Filter nama cabang untuk permintaan tarik untuk memulai pipeline Anda (tipe V2)	ActionDeclaration	AWS::CodePipeline::Pipeline PipelineTriggerDeclaration	Pemicu
17 Oktober 2023	Coba lagi panggung	Mengkonfigurasi percobaan ulang tahap untuk tahap yang gagal atau tindakan yang gagal	ActionDeclaration	N/A	N/A

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.